(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3673714号
(P3673714)
(45)発行日　平成17年7月20日(2005.7.20)　(24)登録日　平成17年4月28日(2005.4.28)

(51)Int.Cl.[7]　F I
G06F 3/12　G06F 3/12　C
G06F 3/00　G06F 3/00　651E

請求項の数 23　(全 22 頁)

(21)出願番号　特願2000-364628(P2000-364628)
(22)出願日　平成12年11月30日(2000.11.30)
(65)公開番号　特開2002-169669(P2002-169669A)
(43)公開日　平成14年6月14日(2002.6.14)
審査請求日　平成14年12月19日(2002.12.19)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100101306
弁理士　丸山　幸雄
(74)代理人　100115071
弁理士　大塚　康弘
(72)発明者　有富　雅規
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

審査官　近藤　聡

最終頁に続く

(54)【発明の名称】ユーザインタフェース制御装置および方法ならびに記憶媒体

(57)【特許請求の範囲】
【請求項１】
ユーザインタフェースを介して入力される所定の制御対象に対する設定データの間に生じる不整合を回避するユーザインタフェース制御装置であって、
不整合回避記述を示すコンフリクト処理ルールを記憶する記憶手段と、
前記記憶手段に記憶された前記コンフリクト処理ルールに基づき、補完的な不整合回避記述を示す補完ルールを生成する補完ルール生成手段と、
前記コンフリクト処理ルールと前記補完ルールとに従い、入力された設定データを更新する更新手段と、
を有することを特徴とするユーザインタフェース制御装置。
【請求項２】
前記記憶手段は、
前記コンフリクト処理ルールを、コンフリクト処理ルール記述ファイルとして記憶することを特徴とする請求項１に記載のユーザインタフェース制御装置。
【請求項３】
前記補完ルール生成手段は、
前記コンフリクト処理ルール記述ファイルに、２状態（ONおよびOFF）を有する前記制御対象の１の機能について、一方の状態に対するコンフリクト処理ルールが複数記述され、他方の状態に対するコンフリクト処理ルールは記述されていないとき、前記一方の状態に対するコンフリクト処理ルールの逆論理を、他方の状態に対するコンフリクト処理ルール

に対する補完ルールとして生成することを特徴とする請求項2に記載のユーザインタフェース制御装置。
【請求項４】
前記コンフリクト処理ルール記述ファイルは、所定のマークアップ言語に従って記述されることを特徴とする請求項３に記載のユーザインタフェース制御装置。
【請求項５】
前記コンフリクト処理ルール記述ファイルには、特定の制御対象にのみ適用可能なローカルルールが記述され、複数の制御対象群に共通して適用可能な普遍ルールを記述したユニバーサルルール記述ファイルを外部参照とすることを特徴とする請求項４に記載のユーザインタフェース制御装置。
【請求項６】
前記コンフリクト処理ルール記述ファイルは、前記ユーザインタフェースの更新コマンドの記述を含むことを特徴とする請求項３ないし５のいずれか１項に記載のユーザインタフェース制御装置。
【請求項７】
前記補完ルール生成手段は、
前記生成した補完ルールを前記コンフリクト処理ルール記述ファイルに追記する手段を更に有することを特徴とする請求項３ないし６のいずれか１項に記載のユーザインタフェース制御装置。
【請求項８】
前記更新手段により前記コンフリクト処理ルールまたは前記補完ルールの適用を受けて設定データが更新されたことを通知する手段を更に有することを特徴とする請求項１ないし７のいずれか１項に記載のユーザインタフェース制御装置。
【請求項９】
前記制御対象は、画像形成装置であることを特徴とする請求項１ないし８のいずれか１項に記載のユーザインタフェース制御装置。
【請求項１０】
ユーザインタフェースを介して入力される所定の制御対象に対する設定データの間に生じる不整合を回避するユーザインタフェース制御方法であって、
不整合回避記述を示すコンフリクト処理ルールを記述したコンフリクト処理ルール記述ファイルを参照し、該コンフリクト処理ルールに基づいて、補完的な不整合回避記述を示す補完ルールを生成する補完ルール生成工程と、
前記コンフリクト処理ルールと前記補完ルールとに従い、入力された設定データを更新する更新工程と、
を有することを特徴とするユーザインタフェース制御方法。
【請求項１１】
前記補完ルール生成工程は、
前記コンフリクト処理ルール記述ファイルに、２状態（ONおよびOFF）を有する前記制御対象の１の機能について、一方の状態に対するコンフリクト処理ルールが複数記述され、他方の状態に対するコンフリクト処理ルールは記述されていないとき、前記一方の状態に対するコンフリクト処理ルールの逆論理を、他方の状態に対するコンフリクト処理ルールに対する補完ルールとして生成することを特徴とする請求項１０に記載のユーザインタフェース制御方法。
【請求項１２】
前記コンフリクト処理ルール記述ファイルは、所定のマークアップ言語に従って記述されることを特徴とする請求項１０に記載のユーザインタフェース制御方法。
【請求項１３】
前記コンフリクト処理ルール記述ファイルには、特定の制御対象にのみ適用可能なローカルルールが記述され、複数の制御対象群に共通して適用可能な普遍ルールを記述したユニバーサルルール記述ファイルを外部参照とすることを特徴とする請求項１２に記載のユー

ザインタフェース制御方法。
【請求項１４】
前記コンフリクト処理ルール記述ファイルは、前記ユーザインタフェースの更新コマンドの記述を含むことを特徴とする請求項１０ないし１３のいずれか１項に記載のユーザインタフェース制御方法。
【請求項１５】
前記補完ルール生成工程は、
前記生成した補完ルールを前記コンフリクト処理ルール記述ファイルに追記する工程を更に有することを特徴とする請求項１０ないし１４のいずれか１項に記載のユーザインタフェース制御方法。
【請求項１６】
前記更新工程により前記コンフリクト処理ルールまたは前記補完ルールの適用を受けて設定データが更新されたことを通知する工程を更に有することを特徴とする請求項１０ないし１５のいずれか１項に記載のユーザインタフェース制御方法。
【請求項１７】
ユーザインタフェースを介して入力される所定の制御対象に対する設定データの間に生じる不整合を回避するユーザインタフェース制御方法をコンピュータによって実現するためのプログラムを格納した記憶媒体であって、
不整合回避記述を示すコンフリクト処理ルールを記述したコンフリクト処理ルール記述ファイルと、
前記コンフリクト処理ルール記述ファイルを参照して、該コンフリクト処理ルールに基づき補完的な不整合回避記述を示す補完ルールを生成する補完ルール生成工程のプログラムコードと、
前記コンフリクト処理ルールと前記補完ルールとに従い、入力された設定データを更新する更新工程のプログラムコードと、
を格納することを特徴とする記憶媒体。
【請求項１８】
前記補完ルール生成工程のプログラムコードは、
前記コンフリクト処理ルール記述ファイルに、２状態（ONおよびOFF）を有する前記制御対象の１の機能について、一方の状態に対するコンフリクト処理ルールが複数記述され、他方の状態に対するコンフリクト処理ルールは記述されていないとき、前記一方の状態に対するコンフリクト処理ルールの逆論理を、他方の状態に対するコンフリクト処理ルールに対する補完ルールとして生成することを特徴とする請求項１７に記載の記憶媒体。
【請求項１９】
前記コンフリクト処理ルール記述ファイルは、所定のマークアップ言語に従って記述されることを特徴とする請求項１７に記載の記憶媒体。
【請求項２０】
前記コンフリクト処理ルール記述ファイルには、特定の制御対象にのみ適用可能なローカルルールが記述され、複数の制御対象群に共通して適用可能な普遍ルールを記述したユニバーサルルール記述ファイルを外部参照とすることを特徴とする請求項１９に記載の記憶媒体。
【請求項２１】
前記コンフリクト処理ルール記述ファイルは、前記ユーザインタフェースの更新コマンドの記述を含むことを特徴とする請求項１７ないし２０のいずれか１項に記載の記憶媒体。
【請求項２２】
前記補完ルール生成工程のプログラムコードは、
前記生成した補完ルールを前記コンフリクト処理ルール記述ファイルに追記する工程のプログラムコードを更に有することを特徴とする請求項１７ないし２１のいずれか１項に記載の記憶媒体。
【請求項２３】

前記更新工程により前記コンフリクト処理ルールまたは前記補完ルールの適用を受けて設定データが更新されたことを通知する工程のプログラムコードを更に有することを特徴とする請求項１７ないし２２のいずれか１項に記載の記憶媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ユーザインタフェースを介して入力される所定の制御対象に対する設定データの間に生じる不整合を回避するユーザインタフェース制御装置および方法ならびに記憶媒体に関する。

【０００２】

【従来の技術】

ユーザインタフェース（以下、「ＵＩ」ともいう。）を介してユーザから複数の設定値の入力を受け付け、それらの設定値に基づき制御される装置の一例として、画像形成装置（プリンタ装置）がある。プリンタ装置は一般に、印刷動作を制御するプリンタドライバを備え、プリンタドライバは、ユーザからの印刷設定等を受け付けるＵＩを含む。

【０００３】

プリンタドライバは、ＵＩを介してユーザから設定値の入力を受け付ける度に、それまでに設定された複数の設定値の中で関連する設定値の値との関係を評価し、設定値間の不整合（コンフリクト）がないかどうかを判別する。コンフリクトの例としては、記録媒体としてセットされたＯＨＰシートに対して両面印刷を行わせるような設定といったユーザにとって不都合と予想される設定や、プリンタに不可能な動作を行わせるような設定等がある。

【０００４】

コンフリクトがあった場合には、それを解消するためのコンフリクト処理を実施する必要がある。

【０００５】

従来は、コンフリクトの判別およびコンフリクト処理を設定値間の関係に依存したかたちで行う専用のコンフリクト処理プログラムを用いるのが一般的であった。あるいは、コンフリクト処理が必要となる複数の設定値の条件を一覧としてまとめてファイル等に保存しておき、このファイルをコンフリクト処理プログラムに読み込ませることで、コンフリクト処理プログラムが特定の設定値に依存することを排除し、コンフリクト処理プログラムが汎用的に利用できるようになっている場合もある。

【０００６】

【発明が解決しようとする課題】

しかしながら、いずれの場合もその実現のためには、プログラム開発者等がすべてのコンフリクト処理ルールを網羅的に記述する必要があった。そのため、設定値間の依存関係が複雑な場合には完全に条件を網羅することができずに、漏れが生じてしまう場合があるという問題がある。

【０００７】

また、従来は、ルール記述のベースは組み合わせであり、１対１の対象機能制御しかできなかった。ルールを追加する場合には、入力者が記述全体をチェックする必要があった。そして、すべての組み合わせを網羅するようデータを作成する必要があるので入力量が非常に多い。また、ルールを１箇所に記載するため反復記述が多く、誤入力の可能性が高いうえ、多大な訂正工数がかかるという問題があった。

【０００８】

本発明は上記の問題点に鑑みてなされたもので、漏れのない確実なコンフリクト処理を実現するとともに、コンフリクト処理ルールの記述方法を改善して、プログラム開発者等による入力工数や人為的ミスを低減することを可能とするユーザインタフェース制御装置および方法ならびに記憶媒体を提供することを目的とする。

【０００９】

【課題を解決するための手段】
上記目的を達成するため、例えば本発明のユーザインタフェース制御装置は、以下の構成を備える。すなわち、
ユーザインタフェースを介して入力される所定の制御対象に対する設定データの間に生じる不整合を回避するユーザインタフェース制御装置であって、
不整合回避記述を示すコンフリクト処理ルールを記憶する記憶手段と、
前記記憶手段に記憶された前記コンフリクト処理ルールに基づき、補完的な不整合回避記述を示す補完ルールを生成する補完ルール生成手段と、
前記コンフリクト処理ルールと前記補完ルールとに従い、入力された設定データを更新する更新手段と、
を有することを特徴とする。
【００１０】
【発明の実施の形態】
（ハードウェア構成）
図１は本発明の一実施形態を示す印刷処理システムのブロック構成図である。印刷処理システムは、ホストコンピュータ３０００と、プリンタ１５００より構成される。
【００１１】
ホストコンピュータ３０００において、１はシステムバス４に接続される各デバイスを総括的に制御するＣＰＵ、２はＣＰＵ１の主メモリ、ワークエリア等として機能するＲＡＭである。３は各種プログラム、データを格納するＲＯＭであって、各種フォントを記憶するフォントＲＯＭ３ａ、ブートプログラムやＢＩＯＳ等を記憶するプログラムＲＯＭ３ａ、および各種データを記憶するデータＲＯＭ３ｃに区分けして構成されている。
【００１２】
５はキーボードコントローラ（ＫＢＣ）で、キーボード９や不図示のポインティングデバイスからのキー入力を制御する。６はＣＲＴコントローラ（ＣＲＴＣ）であり、ＣＲＴディスプレイ（ＣＲＴ）１０の表示を制御する。
【００１３】
外部メモリ１１（ディスクコントローラ（ＤＫＣ）７によりアクセス制御される）は、ハードディスク（ＨＤ）やフロッピーディスク（ＦＤ）等であり、図示の如く、オペレーティングシステムプログラム（以下ＯＳ）２０５をはじめ各種アプリケーション（例えば、図形、イメージ、文字、表等が混在した文書処理を行う文書処理アプリケーションプログラム）２０１、印刷処理関連プログラム２０４を記憶する他、ユーザファイル、編集ファイル等も記憶する。印刷処理関連プログラム２０４は、プリンタ制御コマンド生成モジュール（以下「プリンタドライバ」という）２０４１、プリンタドライバＵＩ制御モジュール２０４２を含む。
【００１４】
８はプリンタコントローラ（ＰＲＴＣ）であり、双方向性インタフェース２１を介してプリンタ１５００に接続されて、プリンタ１５００との通信制御処理を行う。
【００１５】
外部メモリ１１に記憶されたアプリケーションは、ＲＡＭ２にロードされてＣＰＵ１により実行されることになる。また、ＣＰＵ１は、例えばＲＡＭ２へのアウトラインフォントの展開（ラスタライズ）処理を実行し、ＣＲＴ１０上でのWYSIWYG（What you see is What you get）を可能としている。さらに、ＣＰＵ１は、ＣＲＴ１０上の不図示のマウスカーソル等で指示されたコマンドに基づいて登録された種々のウインドウを開き、種々のデータ処理を実行する。ユーザは印刷を実行する際、印刷設定設定画面（プリンタドライバＵＩ制御モジュール２０４２により制御される）を開き、プリンタの設定や、印刷モードの選択を含むプリンタドライバ２０４１に対する印刷処理の設定を行うことができる。
【００１６】
プリンタ１５００において、１２はプリンタ１５００の全体を制御するＣＰＵである。１９はＣＰＵ１２の主メモリ、ワークエリア等として機能するとともに、出力情報展開領域

、環境データ格納領域、ＮＶＲＡＭ等に用いられるＲＡＭであり、図示しない増設ポートに接続されるオプションＲＡＭによりメモリ容量を拡張することができるように構成されている。１３はＲＯＭであり、各種フォントを記憶するフォントＲＯＭ１３ａ、制御プログラム等を記憶するプログラムＲＯＭ１３ｂ、および各種データを記憶するデータＲＯＭ１３ｃより構成される。

【００１７】

外部メモリ１４（メモリコントローラ（ＭＣ）２０によりアクセスを制御される）は、オプションとして接続されるハードディスク（ＨＤ）、フロッピーディスク（ＦＤ）、ＩＣカード等であり、フォントデータ、エミュレーションプログラム、フォームデータ等を記憶する。ハードディスク等の外部メモリ１４が接続されていない場合には、ＲＯＭ１３のデータＲＯＭ１３ｃに、ホストコンピュータ３０００で利用される情報等を記憶することになる。なお、外部メモリ１４は１個に限らず、複数備えるものであってもよく、例えば、内蔵フォントに加えてオプションフォントカード、言語系の異なるプリンタ制御言語を解釈するプログラム等を格納した外部メモリを複数接続できるように構成されていてもよい。

【００１８】

操作部１５０１にはユーザからの操作を受け付ける操作パネルが設けられ、その操作パネルには操作のためのスイッチおよびＬＥＤ表示器等が配されている（図示は省略）。また、図示しないＮＶＲＡＭを有し、操作パネル１５０１からのプリンタモード設定情報を記憶するようにしてもよい。

【００１９】

プリンタＣＰＵ１２は、ＲＯＭ１３のプログラムＲＯＭ１３ｂに記憶された制御プログラム等に基づき、印刷部インタフェース１６を介してシステムバス１５に接続される印刷部（プリンタエンジン）１７に出力情報としての画像信号を出力する。また、ＣＰＵ１２は入力部１８を介してホストコンピュータ３０００との通信処理が可能となっており、プリンタ１５００内の情報等をホストコンピュータ３０００に通知可能に構成されている。

【００２０】

（ソフトウェア構成）

図２は、所定のアプリケーションおよび印刷処理関連プログラムを起動して、ホストコンピュータ３０００上のＲＡＭ２にロードされた状態のＲＡＭ２のメモリマップを示している。ＲＡＭ２には、図示の如く、ＢＩＯＳ２０６、ＯＳ２０５をはじめ、アプリケーション２０１、印刷処理関連プログラム２０４、および関連データ２０３がロードされているとともに、空きメモリ領域２０２も確保されている。これにより、アプリケーション２０１および印刷処理関連プログラム２０４は実行可能状態にある。

【００２１】

印刷処理関連プログラム２０４におけるプリンタドライバＵＩ制御モジュール２０４２は、ユーザによる印刷設定指令に応じてＣＲＴ１０にプリンタドライバＵＩとしての印刷設定画面を表示しユーザからの設定を可能にする。

【００２２】

図８に、印刷設定画面の表示例を示す。同図において、[Print Style]欄８０は、印刷レイアウトを指定する欄であり、ユーザは例えば、片面印刷（1-Sided Printing）８０１、両面印刷（2-Sided Printing）８０２、および製本印刷（Booklet Printing）８０３のうちのいずれかを指定することができる。

【００２３】

[Finishing]欄８１は、印刷済み記録媒体の出力順序および仕上げについて指定する欄であり、ユーザは例えば、以下のいずれかから指定可能となっている。

【００２４】

[Collate] ８１１

部単位印刷。Ｎページの文書をＭ部印刷する場合に、第１ページ、第２ページ、…、第Ｎページの順に１枚ずつ出力し、これをＭ回繰り返す。

【００２５】
[Group] ８１２
ページ単位印刷。Ｎページの文書をＭ部印刷する場合に、第１ページをＭ枚、第２ページをＭ枚、…、第ＮページをＭ枚、の順に出力する。
【００２６】
[Staple] ８１３
ステープル仕上げ。上記[Collate] ８１１と同様に部単位で出力し、仕上げとして各部ごとにステープラで止める。
【００２７】
なお、本明細書では、上記したようなユーザ設定可能項目を「プリンタ機能」または単に「機能」とよぶ。この他にも多くのプリンタ機能を有するが、説明を簡単にするため省略する。
【００２８】
ここで、ユーザにとって不都合と思われる設定の組み合わせ、意味のない設定の組み合わせ、すなわち設定値間の不整合（コンフリクト）は、プリンタドライバＵＩ制御モジュール２０４２により、以下詳細に説明するコンフリクト処理として回避されるように設計される。例えば、図示においては、印刷レイアウトとして片面印刷（1-Sided Printing）８０１が指定されているが、この場合には、[Finishing]欄８１の[Staple] ８１３は薄い灰色で表示され指定できないようにされる。また、図９に示すように、印刷レイアウトとして製本印刷（Booklet Printing）８０３が指定されたときは、[Finishing]欄８１のすべてが指定できないようにされる。上記した例はごく簡単な例示であって、実際に想定されるコンフリクトはかなりの数にのぼるであろう。以下、コンフリクト処理の詳細を説明する。
【００２９】
図３は、実施形態における印刷処理関連プログラム２０４のプリンタドライバＵＩ制御モジュール２０４２の概略構成を示している。３０３は、各モジュール間のデータの受け渡しやデータの更新等を管理してコンフリクト処理を統括するコンフリクトマネージャである。３０６が、上記した印刷設定画面表示としてのプリンタドライバＵＩである。３０１は、後述する記述形式で記述される不整合回避記述を示すコンフリクト処理ルールを列記したコンフリクト処理ルール記述ファイルである。３０２は、コンフリクト処理ルール記述ファイル３０１をロードして新たなコンフリクト処理ルールを生成する推論エンジン、３０４は、各プリンタ機能の状態をリスト形式で表示する状態変数リストであり、ユーザからの入力およびコンフリクト処理ルール記述ファイル３０１の内容に基づき更新されうる。３０５は、プリンタドライバＵＩ３０６が提供する画面表示の基になる帳票としての内部構造体であり、状態変数リスト３０４と連動して各プリンタ機能の状態を所定の形式で表示する。
【００３０】
コンフリクト処理ルール記述ファイル３０１には、骨格となる主たるルールが開発者によりあらかじめ列記される。推論エンジン３０２は、後ほど詳細に説明する方法で新たなコンフリクト処理ルールを自動生成し、コンフリクト処理ルール記述ファイル３０１に追記する。
【００３１】
その後に、プリンタドライバＵＩ３０６を介してユーザからの設定情報を受け取ったコンフリクトマネージャ３０３は、コンフリクト処理ルール記述ファイル３０１を参照する。このことは、図示の如くコンフリクト処理ルール記述ファイル３０１からコンフリクトマネージャ３０３に向かう矢印で、「R(Read)」として表示されている。参照の結果、設定情報がコンフリクト処理ルールに適合する場合、コンフリクト処理が適用される。そうして、コンフリクトマネージャ３０３は、状態変数リスト３０４および内部構造体３０５を更新してプリンタドライバＵＩ３０６に反映させる。この更新作業は、図示の如くコンフリクトマネージャ３０３は、状態変数リスト３０４および内部構造体３０５とは各々双方

向矢印で結ばれ、「R/W (Read/Write)」として表示されている。
【００３２】
図４は、図３に示した各モジュールで扱われるデータの関連を説明する模式図である。図４において、コンフリクト処理ルール記述ファイル３０１は、推論エンジン３０２にインクルード（ロード）されたかたちで参照され、さらに新たなルールが追記される。このコンフリクト処理ルール記述ファイル３０１は、コンフリクトマネージャ３０３にも参照され、それを受けて状態変数リスト３０４が変更されることになる。また、内部構造体３０５と状態変数リスト３０４とは先に述べたとおり連動表示されるものであるから、互いにマッピングされる関係にある。そしてこの状態がプリンタドライバＵＩ３０６によってユーザの目に見えるかたちで表現される。
【００３３】
内部構造体３０５は、プリンタ機能名A, B, C に対応するメンバをそれぞれ cA, cB, cC と表現する。
【００３４】
（コンフリクト処理ルールの記述形式）
次に、コンフリクト処理ルール記述ファイル３０１について説明する。
【００３５】
従来、機能名１対１の記述や機能名群を{ }でくくる記述方法は、いずれも組み合わせをベースとしていた。そのため網羅的な記述が必要であった。先述したとおり、この問題を解決するため、主たるルールを開発者が記述することとし、そこから類推される例えば逆のルール等は推論エンジン３０２によって自動生成させる（詳細は後述）。
【００３６】
ルールの記述形式の概要は、次のとおりである。
【００３７】
・宣言的知識をロジックで表現する。
・論理（ロジック）を用いてコンフリクト処理ルールを数学的に形式化する。
・知識は、普遍的な知識（例えば、複数の制御対象群に共通して適用可能な知識）とローカルな知識（例えば、特定の制御対象にのみ適用可能な知識）に分類できる。普遍的な知識はインクルード可能とする。
・論理のANDは記載する。ORは複数のルールに分割し排除する。NOTは使用可能とする。
・機能は引数を１つ持った述語の形で記述する。
・他の記述から導出できることについては重複記述しない。
【００３８】
この概要から各ルールの記述方法を具体化する。各ルールの記述の基本的な形式は次のとおりである。
【００３９】
・左辺には機能名(ON)、機能名(OFF)、機能名(値)を記入する。
・機能が成り立つ場合の論理を記入するときは、(ON)に対するすべてのルールを記述する。(OFF)に対するルールの記述は不要である（後述するように自動生成されるため）。
・機能が成り立たない場合の論理を記入するときは、(OFF)に対するすべてのルールを記述する。(ON)に対するルールの記述は不要である（自動生成されるため）。
・右辺には左辺が成り立つための論理を、機能名(ON)、機能名(OFF)、機能名(値)を記入する。複数の項を記述可能。また論理否定のNOTを使用可能である。
【００４０】
上記したように、各ルールは、論理（ロジック）を用いてコンフリクト処理ルールを数学的に形式化する。述語は、「プリンタ機能名（引数）」の形で記述される。引数としては、ON/OFFの他、数値が入る場合がある（例えば、印刷部数等）。左辺にはプリンタ機能名（引数）を、右辺には左辺が成り立つための論理を記述し、記号「←」または「＜－」で関係づける。例えば、
A(ON) ← B(ON).

は、「プリンタ機能Bの状態がONのときは、プリンタ機能Aの状態をONとする」という意味のルールになる。

【００４１】

また、式中の記号「,」は「かつ（AND）」の意味で用いられる。例えば、「プリンタ機能Bの状態がON、かつ、プリンタ機能Cの状態がOFFのとき、プリンタ機能Aの状態をONとする」というルールは、

A(ON) ←B(ON), C(OFF).

と記述される。

【００４２】

上記した論理式の具体的な表記法は、宣言/論理型言語に準拠する形式で記述することもできるが、一部の表記方法や右左辺を反転しても同形式とみなすことができる。機能名、ON/OFFや()の表記の方法は、適宜設計し、またネットワーク経由で交換することも考慮し、マークアップ言語内に記述できるよう定義することもできる（マークアップ言語内での記述例については後述する）。

【００４３】

図６は、以上の例に従って記述されている。同図において、プリンタ機能として、図８に示した[Collate] ８１１に対応する部単位印刷機能、[Group] ８１２に対応するページ単位印刷機能、および[Staple] ８１３に対応するステープル仕上げ機能はそれぞれ、Collate()、Group()、およびStaple()で示され、引数はONまたはOFFとなる。また、[PrintStyle]欄８０に対応する印刷レイアウト機能は、Layout()で示され、引数は、1-Sided、2-Sided、Bookletのいずれかである。

【００４４】

図６の(1)は、ユーザにより[Group] ８１２をチェックされたことでGroup(ON)となったときは、Collate(OFF)とするルールを表す。(2)は、Staple(ON)となったときは、同じくCollate(OFF)とするルールである。また、(3)は、Layout(Booklet)となったときは、Group(OFF)とするルールを示している。

【００４５】

（コンフリクト処理ルールの自動生成）

・開発者は、ある機能名に対して、ONとなるすべてのルールを記述し、上記したとおり、OFFとなるルールを省略することができる。OFFとなるルールは推論エンジン３０２により自動生成される。

・逆に、開発者は、ある機能名に対して、OFFとなるすべてのルールを記述し、上記したとおり、ONとなるルールを省略することができる。ONとなるルールは推論エンジン３０２により自動生成される。

・開発者は、ある機能名に対して、ONとなるルール、OFFとなるルールのすべてを記述することもできる。この場合ルールは自動生成されない。

・左辺に機能名(ON)、機能名(OFF)の形式で記述された項に対して、右辺に項目を記述する場合には、左辺と同一のON/OFF形式にする。この場合にルールが自動生成される。

【００４６】

以下、代表的な表記方法を用いて説明する。

【００４７】

ここで、論理式における左辺と右辺の必要十分条件的関係について補足説明しておく。

【００４８】

A(ON) ← B(ON). (a)

というルールが１行だけ記述されていた場合には、B(ON)はA(ON)に対して十分条件であり、逆にA(ON)はB(ON)に対して必要条件となる。したがって、下記のような逆の条件のルールは一般的には逆はまた真なりとはならない。

B(ON) ← A(ON). (b)

A(OFF) ← B(OFF). (c)

【００４９】

上記(a)、(b)、(c)が同時に成り立つ場合には、A(ON)とB(ON)は互いに必要十分条件の関係にある。(a)に対して論理的な対偶である下記のルール
B(OFF) ← A(OFF).　　(d)
は必ず真である。したがって、上記(a)、(b)、(c)が同時に成り立つ場合には、(a)か(d)のいずれか一方が開発者により記載されていれば、推論エンジン３０２により自動的に正しいコンフリクト条件（ルール）が生成されることになる。

【００５０】

次に、処理ルールの記述方法と自動生成されるロジックの関係について説明する。

【００５１】

２状態値(ON,OFF)を値として持つルールで、同じ機能名について複数行記述されている場合には、原則として左辺にONまたはOFFの一方のみを記述する。例えば、

【００５２】

A(ON) ← B(ON),C(OFF).
A(ON) ← D(V1).
B(OFF) ← E(OFF).

ただし、引数V1は、数値を表す。この場合のルールに対して推論エンジン３０２は、次に示すON/OFF逆側のルールを補完ルールとして自動生成する。

【００５３】

A(OFF) ← true.　　(e)
B(ON) ← true.

【００５４】

この補完ルールは下記のルールを最適化したものである。

【００５５】

A(OFF) ← not A(ON).
B(ON) ← not B(OFF).

【００５６】

これは論理としてはA(ON)とA(OFF)は完全に排他な関係にあることを意味している。つまりA(ON)とA(OFF)でAについての集合空間が100％埋め尽くされる。B(ON)とB(OFF)も同様である。結果、A(ON/OFF)が失敗することはなくなり、ON/OFFのいずれかが成立する。

【００５７】

ユーザがA(ON)とA(OFF)を混在して記述する場合には(e)の自動生成はされない。このような場合にはAの集合空間を埋め尽くすように記述する必要がある。

【００５８】

処理ルールには機能名( )以外にプライオリティやアクションを記述できる組み込み述語を用意する。以下、組み込み述語と関連記述方法について説明する。

【００５９】

（プライオリティおよびアクションの記述）
プライオリティの記述に用いる組み込み述語の代表例を挙げておく。

【００６０】

status(機能名、値)
引数として指定した機能名の現在値が指定した値のときはtrue、そうでないときはfalseを返す。

【００６１】

また、ルールに対してそのアクションを記述することができる。代表例を挙げておく。

【００６２】

右辺に、{ }で囲み、ルールが成り立ったときに実行されるアクションを記述する。{ }内の組み込み述語としてはメッセージを表示するMessage()や、コントロールを制御するEnable,Disable,Show,Hide等が使用可能である。プライオリティやアクションは、オプショナルの表現形式としてはルールの形式に合わせて設計すればよい。

【００６３】

（組み込み述語とデフォルト値）
組み込み述語status(A,_X)によって機能Aの状態変数値を変数_Xに受けることができる。推論エンジン３０２は、コンフリクト処理ルール記述ファイル３０１をロードした後、出現するすべてのルール名について次のルールを自動的に生成する。
【００６４】
A(_X) ← status(A,_X).
B(_X) ← status(B,_X).
C(_X) ← status(C,_X).
...
A(ON) ← B(ON),C(OFF)
【００６５】
Aの状態値は上記ルールを受けてONになる。Bがもし上記自動生成された以外のルールが存在しないならば、
B(_X) ← status(B,_X).
が適用される。この自動生成ルールは必ず成り立つので、Bの状態変数の値ONが_Xにユニファイされルールの状態値となる。
【００６６】
（状態変数の充足機構）
確認した状態変数に関連する全ルールに対して充足処理を行う。
【００６７】
例）
C(ON) ← A(ON).
B(OFF) ← A(ON).
A(ON).
Aの状態変数を確認した場合、AにONが充足され、それを参照しているB,CにそれぞれOFF,ONが充足される。確認した状態変数に関連する全ルールに対して充足処理をする。
【００６８】
（拘束理由の設定）
sreason(R)の使用によって、状態変数の結果がそうなった理由を設定することができる。
【００６９】
例）
B(OFF) ← A(OFF), { sreason(R) }.
AがOFFであるとBもOFFになる場合の理由をRに設定しておく。コンフリクト等が発生した場合の理由を後に取り出すことができる。例えば、コンフリクト処理ルール記述ファイル３０１に、
A(ON) ← B(ON), C(OFF).
が記述されている場合、図４の状態変数リスト３０４に示すように、コンフリクト処理ルール記述ファイル３０１の中に出現するプリンタ機能A，B，C のそれぞれについて同名の状態変数が存在している。
【００７０】
（プリンタドライバＵＩ制御モジュール２０４２の処理の内容）
以下、図５のフローチャートを用いて、コンフリクト処理を含むプリンタドライバＵＩ制御モジュール２０４２の処理について、詳しく説明する。
【００７１】
プリンタドライバＵＩ制御モジュール２０４２の処理は、ユーザがキーボードコントローラＫＢＣ５等を用いて、プリンタドライバＵＩを開く指示をすることで始まる。ユーザがプリンタドライバＵＩを開くよう指示すると、上記したとおり、ＯＳ２０５の管理の下、ＲＡＭ２に印刷処理関連プログラム２０４がロードされる。ここで印刷処理関連プログラム２０４は、ページ記述言語を用いて記述される印刷データを生成するプログラムであるため、同系列の複数のプリンタに対して共通に利用されるモジュールである。そのため、

印刷要求時には、本印刷処理関連プログラム２０４は、ユーザがプリンタドライバＵＩを開くよう指示したプリンタドライバＵＩを起動する必要がある。

【００７２】

印刷処理関連プログラム２０４がＲＡＭ２０５にロードされると、まず、プリンタドライバＵＩを開くための初期化処理として、推論エンジン３０２は、コンフリクト処理ルール記述ファイル３０１をコンフリクトマネージャ３０３を介して、ＲＡＭ2に読み込む（ステップＳ５０１）。

【００７３】

続いて、ステップ５０１で読み込んだコンフリクト処理ルールを元に、２状態値（ONとOFF）を値として持つルールについて、補完ルールを新たに生成する（ステップＳ５０２）。ここで例を示す。コンフリクト処理ルール記述ファイル３０１が、以下のように記述されていたとする。

【００７４】

A(ON) ←B(ON), C(OFF).　　... (f)
A(ON) ←D(V1).　　... (g)
B(OFF) ← E(OFF).　　... (h)

【００７５】

この場合、(f)と(g)はともに、同一のプリンタ機能Aについて、同一の状態ONについての記述がされており、他の式、ここでは(h)、においては、A(OFF)についての記述はない。このように、左辺に、同じプリンタ機能について、ONまたはOFFのどちらかだけが、コンフリクト処理ルール記述ファイル３０１に記述されている場合、上述したとおり、推論エンジン３０２は、当該式に対してON/OFF逆側のルールを自動生成する。したがって、この例の場合、(f)、(g) に対しては、

A(OFF) ←not A(ON).　　... (i)

が生成され、(h)に対しては、

B(ON) ← not B(OFF).　　... (j)

が生成される。この自動生成された(i)、(j)のルールは処理効率の観点から最適化されて以下の(i)',(j)'のように変形される。ただし、意味は全く同じである。

A(OFF) ← true.　　... (i)'
B(ON) ← true.　　... (j)'

【００７６】

上記した例のように逆側が自動生成された場合は、論理として考えた場合に、A(ON) とA(OFF) とは完全に排他な関係にあることを意味している。つまりA(ON)とA(OFF) とでプリンタ機能Aについての集合空間が100%埋め尽くされることを意味している。B(ON) と B(OFF) も同様である。

【００７７】

なお、 A(ON) とA(OFF) が混在して記述されている場合には、(i)および(i)' は自動生成されない。したがってこの場合には、すべてのケースを網羅的に記述してAについての集合空間を100%埋め尽くすようにする必要がある。

【００７８】

続いて、状態変数リスト３０４とコンフリクト処理ルール記述ファイル３０１で使用されるプリンタ機能名の値の初期化に関するコンフリクト処理ルールを、補完ルールとして自動生成する（ステップＳ５０３）。

【００７９】

コンフリクト処理ルール記述ファイル３０１に記述されるすべてのプリンタ機能名は、コンフリクトマネージャ３０３の内部にインクルードされた状態変数リスト３０４にそれぞれ、状態変数を持っている。この状態変数の値は、プリンタドライバＵＩ３０６で使用される内部構造体３０５の対応するメンバの値と連動している。各プリンタ機能名の状態変数の初期値は、その内部構造体３０５のメンバの値となる。

【００８０】

例えば、図４において、内部構造体３０５に記述されているint cAの初期値は０であるから、それに対応する状態変数リスト３０４におけるプリンタ機能Aの値はOFFとなっている。したがって、推論エンジン３０２内に記述される補完ルールの、プリンタ機能Aの状態の初期値はOFFとなる。

【００８１】

その後、推論エンジン３０２は、コンフリクト処理ルール記述ファイル３０１を参照することでコンフリクトチェックの推論を行う。例えば、図４に示すように、コンフリクト処理ルール記述ファイル３０１に記述されている、

A(ON) ← B(ON), C(OFF).

が成立した場合、推論エンジン３０２は、状態変数リスト３０４の、プリンタ機能Aの状態変数値を、初期値OFFからONに変更する。コンフリクトチェックの推論が終了した後、コンフリクトマネージャは、変更された状態変数の値を内部構造体３０５の対応するメンバ int cAに反映させる。すなわち、int cA は上記ルールが成立したことによって０から１に変更される。

【００８２】

推論エンジン３０２は、組込み関数 status(a, _X) によってプリンタ機能A の状態変数値を推論エンジン３０２で使用される変数 _X に受けることができる。推論エンジン３０２はコンフリクト処理ルール記述ファイル３０１をロードした後、そのコンフリクト処理ルール中に出現する全てのルール名について、次のルールを自動的に生成する。

【００８３】

A(_X) ← status(A,_X).
B(_X) ← status(B,_X).
C(_X) ← status(C,_X).
:

【００８４】

これは、他に適用するルールが存在しない場合には、内部構造体３０５の対応するメンバの値がそのプリンタ機能名の状態値となることを意味する。

【００８５】

A については、

A(ON) ← B(ON), C(OFF).

が成り立つので、Aの状態値はこれを受けてONとなる。Bがもし上記自動生成されたルール以外にルールが存在しないならば、

B(_X) ← status(B,_X).

が適用される。この自動生成ルールは必ず成り立つので、B の状態変数の値ONが_Xにユニファイされ、それがプリンタ機能Bの状態値となる。つまり、ユーザ定義ルールが存在しないか、または存在していても成り立つものがないプリンタ機能については、内部構造体３０５の対応するメンバに格納されている値がそのプリンタ機能の状態値ということになる。

【００８６】

図７は、ステップＳ５０２およびステップＳ５０３によって生成されたルールを、図６に示した元のコンフリクト処理ルール記述ファイル３０１に追記した例を示している。ステップＳ５０２では、(1)および(2)から生成される(4)と、(3)から生成される(5)とが補完ルールとして追記される。さらに、ステップＳ５０３では、(6)から(9)までが追記されることになる。

【００８７】

続いて、プリンタドライバＵＩ３０６のオープンのために必要とされるその他の初期化処理を行い、図８に例示したようなプリンタドライバＵＩをオープンする（ステップＳ５０４）。

【００８８】

プリンタドライバＵＩ３０６がオープンされた後は、ＯＳより送られてくるイベントを取

得し、そのイベントに対する処理を繰り返す（ステップＳ５０５）。
【００８９】
次に、ステップＳ５０５にて取得したイベントが、ユーザがプリンタドライバＵＩ３０６上の設定項目を変更したイベントであるかどうかの判別を行い（ステップ５０６）、そうでなかった場合には、ステップＳ５１２に進み、プリンタドライバＵＩ３０６のクローズ要求かどうかの判別を行う。クローズ要求であった場合には、ステップＳ５１３に進み、終了処理を行い、プリンタドライバＵＩ３０６をクローズして、すべての処理を終了する。一方、ステップＳ５１２で、クローズ要求ではなかった場合は、ステップＳ５０５に戻り、処理を繰り返す。
【００９０】
ステップＳ５０６で、ステップＳ５０５で取得したイベントがユーザによる設定変更要求であった場合には、ステップＳ５０７に進み、ステップＳ５０１からステップＳ５０３の処理により新たに生成したコンフリクト処理ルールを適用する。
【００９１】
取得したイベントがユーザによる設定変更要求であった場合の一例として、図８に示す[Print Style]欄８０における、片面印刷（1-Sided Printing）８０１から製本印刷（Booklet Printing）８０３に変更するものであった場合について説明する。このとき、内部構造体３０５のメンバとして存在するCollate、Group、Staple、Layoutの各メンバのコンフリクト処理ルール適用前、すなわち、設定変更要求前における値は、以下のようになっていた。
【００９２】
Collate　OFF
Group　ON
Staple　OFF
Layout　1-Sided
【００９３】
ユーザの変更要求がLayoutを１－SidedからBookletに変更するものであるので、Layoutのメンバの内容が変更されて、内部構造体３０５の各メンバの値は次のようになる。
【００９４】
Collate　OFF
Group　ON
Staple　OFF
Layout　Booklet
【００９５】
すると、プリンタドライバＵＩ３０６はコンフリクトマネージャ３０３をコールし、状態変数リスト３０４にあるLayoutの状態変数が更新され、続いて推論エンジン３０２がコールされて、コンフリクト処理ルールの適用が始まる。まず図７の(6)～(9)のルールが適用され、推論エンジン３０２内の各プリンタ機能名が状態変数リストの各メンバの持つ値で初期化される。続いて、図７の(3)が適用され、その結果、次のようにGroupの値はONからOFFへと変化する。
【００９６】
Collate　OFF
Group　OFF
Staple　OFF
Layout　Booklet
【００９７】
さらに、図７の(4)のルールが適用され、CollateがOFFからONへと変化する。
【００９８】
Collate　ON
Group　OFF

Staple OFF
Layout Booklet
【００９９】
この他に、適用されるルールが存在しなくなると、推論エンジン３０２でのコンフリクト処理ルールの適用が終了する。
【０１００】
コンフリクトマネージャ３０３は、上記ステップＳ５０７のコンフリクト処理ルールをの適用結果に基づき状態変数リストを更新し（ステップＳ５０８）、内部構造体３０５を更新する（ステップＳ５０９）。
【０１０１】
続いて、プリンタドライバＵＩ３０６が内部構造体３０５のメンバの値を参照して、ＵＩの更新が必要かどうかの判別を行う（ステップＳ５１０）。ＵＩの更新の必要がない場合には、ステップＳ５０５に戻り、処理を繰り返す。ＵＩの更新が必要な場合には、ＵＩの更新を行ったうえで（ステップＳ５１１）、ステップＳ５０５に戻り、処理を繰り返すことになる。上記の例では、Layoutが１－Sided PrintingからBooklet Printingに設定が変更されたことにより、CollateがOFFからONへ、GroupがONからOFFへと変化しているので、プリンタドライバＵＩは、図８から図９に示すとおりに更新される。
【０１０２】
以上の処理は、プリンタドライバＵＩ３０６がクローズされるまで、繰り返し実行される。プリンタドライバＵＩ３０６がクローズされると処理は終了し、印刷処理関連プログラム２０４の処理も終了し、ＲＡＭ２からはＯＳ２０５の機能により消去される。
【０１０３】
ところで、プリンタドライバＵＩの更新処理の実行は、コンフリクト処理ルール記述ファイル３０１の中に、プリンタドライバＵＩを更新するための処理を記述し、推論エンジン３０２がその記述を解釈した時点で、コンフリクトマネージャ３０３の状態変数リスト３０４を介して、プリンタドライバＵＩの更新処理を直接行うようにしても良い。
【０１０４】
図１０は、図７に示した追記後のコンフリクト処理ルールのうち、(3)の次の行に、ＵＩ更新の処理として{disable}の記述を追加したものである。この記述により、図９におけるGroupラジオボタンコントロールをdisableする処理（設定不可の状態にする処理）がコンフリクト処理ルールの適用内容の一部として実現されることになる。
【０１０５】
さらに、図１１に示すように、コンフリクト処理ルールの中に、ユーザに対する情報表示を可能にするメッセージボックスの表示処理を加えることも可能である。例えば、図１１の{disable}の次の行の記述{ Message(MSG001) }は、図１２に示すようなメッセージボックスを表示させることを示している。「MSG001」は、図１２に表示されているメッセージテキスト「Groupの設定はCollateに調整されました。」の文字列を指示するIDで、ID: MSG001およびそのIDが示す文字列は、コンフリクトマネージャ３０３内に文字列リソースとして存在している。
【０１０６】
次に、コンフリクト処理ルールの記述をマークアップ言語（例えば、XML (Extensible Markup Language) ）内におく具体例について説明する。
【０１０７】
図１３は、マークアップ言語内におけるコンフリクト処理ルールの記述例を示している。図示のように、コンフリクト処理ルール部は＜コンフリクトルールズ＞タグの間に記述され、各ルールは＜ルール＞タグで囲まれる。構造が指示されれば、タグ名は任意でかまわない。
【０１０８】
また、コンフリクト処理ルールは、多くの機種のプリンタに共通して適用可能なルール群（普遍（ユニバーサル）ルール）と、特定の機種のプリンタにのみ適用可能なルール群（

ローカルルール）とに分類することができよう。この場合、例えば、図示の如く、普遍ルールは＜コンフリクトルールズ.ユニバーサル＞タグで囲み、ローカルルールは＜コンフリクトルールズ.ローカル＞タグで囲むことも可能である。

【０１０９】

さらに、図１４に示すように、普遍ルールだけを記述したコンフリクトルールズ.ユニバーサルファイルを外部参照用ファイルとして作成し、コンフリクト処理ルール記述ファイルにインクルードするかたちにすることも可能である。

【０１１０】

以上説明したように、実施形態によれば、プログラム開発者などが用意するコンフリクト処理ルールを補完するコンフリクト処理ルールを自動生成するので、より品質の高いコンフリクト処理を実現することができる。

【０１１１】

また、ルール記述の基本をロジックでおこなうことにより、１対多の機能制御が可能となる。依存関係はロジックにより伝播するため、新しいルールの追加が容易である。ルール追加時も記述全体をチェックする必要はない。そして、ロジックの自動生成が行われるのでコンフリクトすべての組み合わせを網羅するようにデータを作成する必要はなくなる。また、汎用的なルールは別に用意できるため反復記述が抑えられ、誤入力と訂正工数を減らすことができる。

【０１１２】

さらに、ユーザインタフェースの更新処理、メッセージ処理もコンフリクト処理ルールに加えることで、開発者にとって可読性の高い、メンテナンスの容易なコーディングを実現することができる。

【０１１３】

また、コンフリクト処理ルールに、ユーザインタフェース更新処理とメッセージ処理を、コンフリクト処理とともに設けることで、コンフリクト処理ルールが変更された場合でも、ユーザインタフェース制御モジュール自体を変更する必要がなくなる。

【０１１４】

なお、上述した実施形態では、プリンタ装置に対して、コンフリクト処理を含むＵＩ制御を行う例について説明したが、プリンタ装置に限らず、ディジタルカメラ、ディジタルレコーダ、イメージスキャナ等の周辺装置、制御機器の他、モデムやルータといったネットワーク関連機器にも適用可能であることはいうまでもない。

【０１１５】

【他の実施形態】

なお、本発明は、複数の機器（例えばホストコンピュータ、インタフェース機器、リーダ、プリンタなど）から構成されるシステムに適用しても、一つの機器からなる装置（例えば、複写機、ファクシミリ装置など）に適用してもよい。

【０１１６】

上述したように、本発明の目的は、前述した実施形態の機能を実現するソフトウェアのプログラムコードを記録した記憶媒体（または記録媒体）を、システムあるいは装置に供給し、そのシステムあるいは装置のコンピュータ（またはＣＰＵやＭＰＵ）が記憶媒体に格納されたプログラムコードを読み出し実行することによっても、達成される。この場合、記憶媒体から読み出されたプログラムコード自体が前述した実施形態の機能を実現することになり、そのプログラムコードを記憶した記憶媒体は本発明を構成することになる。また、コンピュータが読み出したプログラムコードを実行することにより、前述した実施形態の機能が実現されるだけでなく、そのプログラムコードの指示に基づき、コンピュータ上で稼働しているオペレーティングシステム（ＯＳ）などが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【０１１７】

さらに、記憶媒体から読み出されたプログラムコードが、コンピュータに挿入された機能

拡張カードやコンピュータに接続された機能拡張ユニットに備わるメモリに書込まれた後、そのプログラムコードの指示に基づき、その機能拡張カードや機能拡張ユニットに備わるＣＰＵなどが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【０１１８】

本発明を上記記憶媒体に適用する場合、その記憶媒体には、先に説明した図５に示すフローチャートに対応するプログラムコードが格納されることになる。

【０１１９】

【発明の効果】

以上説明したように、本発明によれば、漏れのない確実なコンフリクト処理を実現するとともに、コンフリクト処理ルールの記述方法を改善して、プログラム開発者等による入力工数や人為的ミスを低減することを可能とするユーザインタフェース制御装置および方法ならびに記憶媒体を提供することができる。

【図面の簡単な説明】

【図１】実施形態に係る印刷処理システムのブロック構成図である。

【図２】実施形態におけるＲＡＭ２のメモリマップを示す図である。

【図３】実施形態におけるプリンタドライバＵＩ制御モジュールの概略構成図である。

【図４】実施形態におけるプリンタドライバＵＩ制御モジュールで扱われるデータの関連を説明する模式図である。

【図５】実施形態におけるプリンタドライバＵＩ制御モジュールの処理を示すフローチャートである。

【図６】実施形態におけるコンフリクト処理ルールの一例を示す図である。

【図７】実施形態におけるコンフリクト処理ルールの一例を示す図である。

【図８】実施形態における印刷設定画面の一例を示す図である。

【図９】実施形態における印刷設定画面の一例を示す図である。

【図１０】実施形態におけるコンフリクト処理ルールの一例を示す図である。

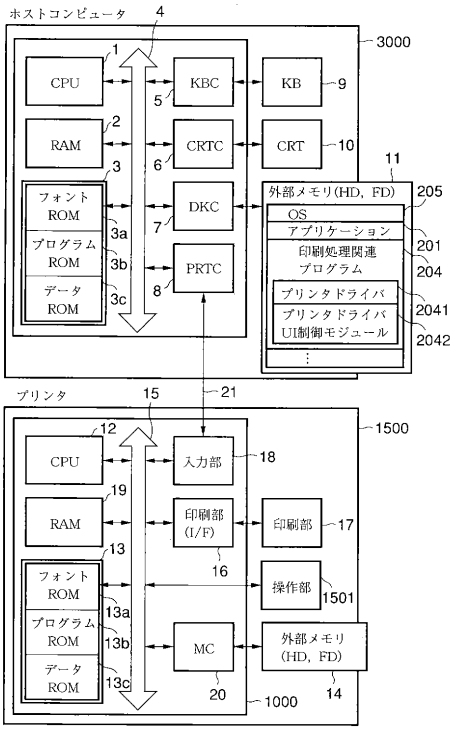
【図１１】実施形態におけるコンフリクト処理ルールの一例を示す図である。

【図１２】実施形態におけるメッセージボックスの一例を示す図である。

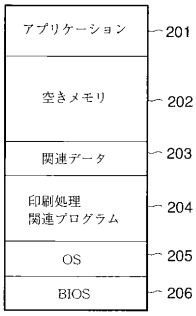
【図１３】実施形態におけるマークアップ言語内におけるコンフリクト処理ルールの記述例を示す図である。

【図１４】実施形態におけるマークアップ言語内におけるコンフリクト処理ルールの記述例を示す図である。

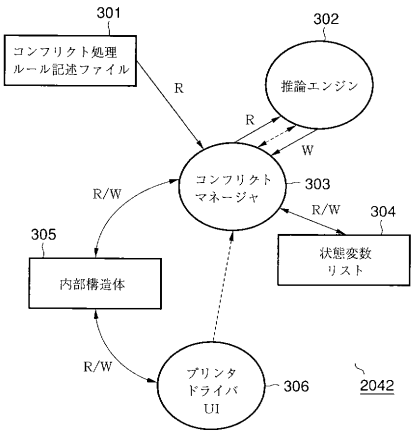
【図１】


ホストコンピュータ
4
3000
CPU
1
KBC
5
KB
9
RAM
2
CRTC
6
CRT
10
11
フォント ROM
3
DKC
7
外部メモリ(HD, FD)
205
プログラム ROM
3a
OS
データ ROM
3b
アプリケーション
201
3c
PRTC
8
印刷処理関連プログラム
204
プリンタドライバ
2041
プリンタドライバ UI制御モジュール
2042
プリンタ
15
21
1500
CPU
12
入力部
18
RAM
19
印刷部 (I/F)
印刷部
17
フォント ROM
13
16
操作部
1501
プログラム ROM
13a
MC
外部メモリ (HD, FD)
データ ROM
13b
13c
20
1000
14

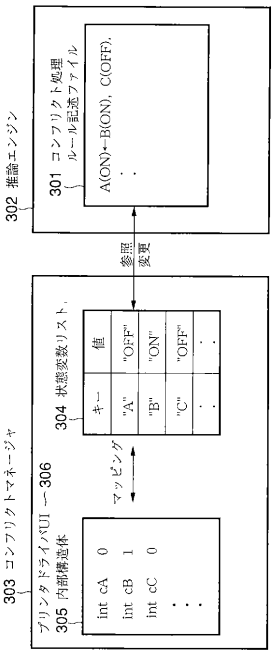
【図２】


アプリケーション
201
空きメモリ
202
関連データ
203
印刷処理関連プログラム
204
OS
205
BIOS
206

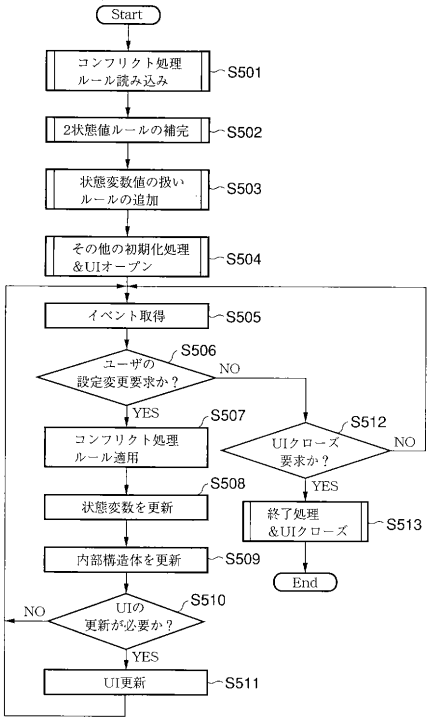
【図３】


301
コンフリクト処理ルール記述ファイル
302
推論エンジン
R
R
W
コンフリクトマネージャ
303
R/W
R/W
304
305
状態変数リスト
内部構造体
R/W
プリンタドライバ UI
306
2042

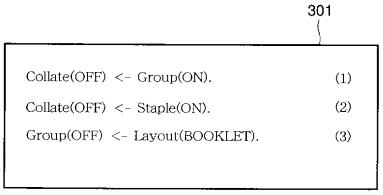
【図４】


302 推論エンジン
301 コンフリクト処理ルール記述ファイル
A(ON)←B(ON), C(OFF),
参照
変更
303 コンフリクトマネージャ
プリンタドライバUI 306
305 内部構造体
int cA 0
int cB 1
int cC 0
マッピング
304 状態変数リスト
キー
値
"A"
"OFF"
"B"
"ON"
"C"
"OFF"

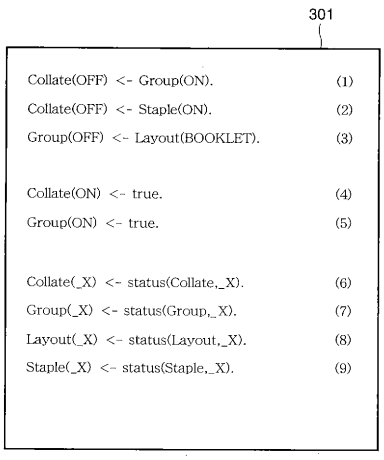
【図５】


Start
コンフリクト処理ルール読み込み
S501
2状態値ルールの補完
S502
状態変数値の扱いルールの追加
S503
その他の初期化処理&UIオープン
S504
イベント取得
S505
ユーザの設定変更要求か？
S506
NO
YES
コンフリクト処理ルール適用
S507
UIクローズ要求か？
S512
NO
YES
状態変数を更新
S508
終了処理&UIクローズ
S513
内部構造体を更新
S509
End
UIの更新が必要か？
S510
NO
YES
UI更新
S511

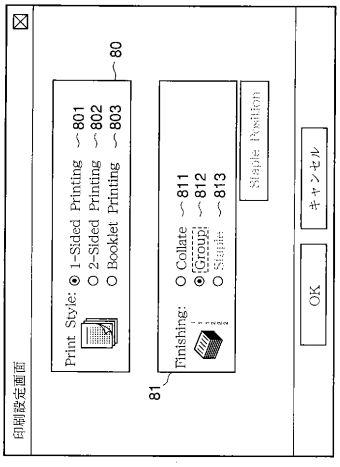
【図６】


301
Collate(OFF) <- Group(ON). (1)
Collate(OFF) <- Staple(ON). (2)
Group(OFF) <- Layout(BOOKLET). (3)

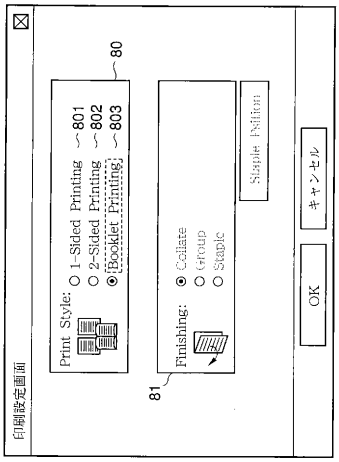
【図７】


301
Collate(OFF) <- Group(ON). (1)
Collate(OFF) <- Staple(ON). (2)
Group(OFF) <- Layout(BOOKLET). (3)
Collate(ON) <- true. (4)
Group(ON) <- true. (5)
Collate(_X) <- status(Collate,_X). (6)
Group(_X) <- status(Group,_X). (7)
Layout(_X) <- status(Layout,_X). (8)
Staple(_X) <- status(Staple,_X). (9)

【図８】


印刷設定画面
80
Print Style:
1-Sided Printing 801
2-Sided Printing 802
Booklet Printing 803
81
Finishing:
Collate 811
Group 812
Staple 813
Staple Position
OK
キャンセル

【図９】



【図１０】

```
Collate(OFF) <- Group(ON).                    (1)
Collate(OFF) <- Staple(ON).                   (2)
Group(OFF) <- Layout(BOOKLET).                (3)
                {disable}
Collate(ON) <- true.                          (4)
Group(ON) <- true.                            (5)
Collate(_X) <- status(Collate,_X).            (6)
Group(_X) <- status(Group,_X).                (7)
Layout(_X) <- status(Layout,_X).              (8)
Staple(_X) <- status(Staple,_X).              (9)
```

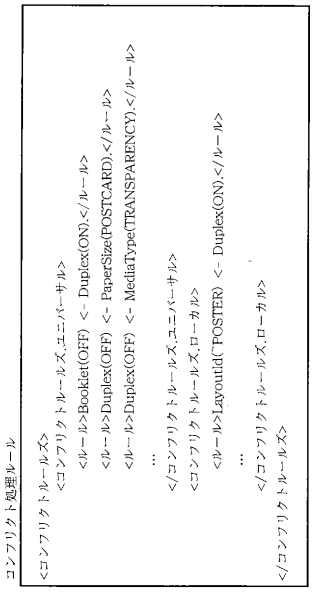
【図１１】

```
Collate(OFF) <- Group(ON).                    (1)
Collate(OFF) <- Staple(ON).                   (2)
Group(OFF) <- Layout(BOOKLET).                (3)
                {disable}
                {Message(MSG001)}
Collate(ON) <- true.                          (4)
Group(ON) <- true.                            (5)
Collate(_X) <- status(Collate,_X).            (6)
Group(_X) <- status(Group,_X).                (7)
Layout(_X) <- status(Layout,_X).              (8)
Staple(_X) <- status(Staple,_X).              (9)
```

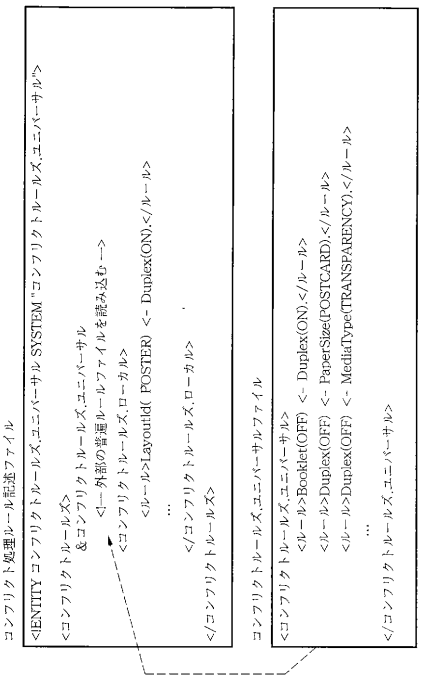
【図１２】



【図１３】


コンフリクト処理ルール
<コンフリクトルールズ>
<コンフリクトルールズ.ユニバーサル>
<ルール>Booklet(OFF) <- Duplex(ON).</ルール>
<ルール>Duplex(OFF) <- PaperSize(POSTCARD).</ルール>
<ルール>Duplex(OFF) <- MediaType(TRANSPARENCY).</ルール>
...
</コンフリクトルールズ.ユニバーサル>
<コンフリクトルールズ.ローカル>
<ルール>LayoutId("POSTER) <- Duplex(ON).</ルール>
...
</コンフリクトルールズ.ローカル>
</コンフリクトルールズ>

【図１４】


コンフリクト処理ルール記述ファイル
<!ENTITY コンフリクトルールズ.ユニバーサル SYSTEM "コンフリクトルールズ.ユニバーサル">
<コンフリクトルールズ>
&コンフリクトルールズ.ユニバーサル
<!-- 外部の普遍ルールファイルを読み込む -->
<コンフリクトルールズ.ローカル>
<ルール>LayoutId(POSTER) <- Duplex(ON).</ルール>
...
</コンフリクトルールズ.ローカル>
</コンフリクトルールズ>
コンフリクトルールズ.ユニバーサルファイル
<コンフリクトルールズ.ユニバーサル>
<ルール>Booklet(OFF) <- Duplex(ON).</ルール>
<ルール>Duplex(OFF) <- PaperSize(POSTCARD).</ルール>
<ルール>Duplex(OFF) <- MediaType(TRANSPARENCY).</ルール>
...
</コンフリクトルールズ.ユニバーサル>

フロントページの続き

(56)参考文献 特開平１１－１２６１２９（ＪＰ，Ａ）
実開平０４－０３３１３４（ＪＰ，Ｕ）
特開２０００－２４２５９７（ＪＰ，Ａ）
特開2002－196905（ＪＰ，Ａ）
特開2002－202868（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 3/00