Vol. 27 No. 5 情報処理学会論文誌 May 1986

ルール型制御ソフトウェアシステム SCD (Station Coordinator) の開発†

田代 勤†† 鷹田憲久††
都島 功†† 松本邦顕††

製品寿命の短期化，種類の多様化から，生産プロセスの自動化のための制御ソフトの迅速な開発，変更が必要となっている．これに対し，制御仕様をルールとして宣言するだけで，自動的に制御動作を導けるプロダクションシステムの手法が注目されている．しかし，これを制御ソフトの開発に一種の言語として適用する場合，(1)対象ごとのセンサ・アクチュエータ構成およびその変更に柔軟に対処可能なプロセスインタフェース方式，(2)実時間に適用できる高速な処理系，(3)設計時の処理時間の見積り方式の確立が必要となるが，従来これらについてほとんど触れられていない．本論文では，上記を満足する，生産プロセス自動化向きのルール型制御ソフトウェアシステムについて報告する．自動化システムのセンサ・アクチュエータの信号は ON-OFF 型に集約できる．このことから，まず，このような信号の接続，構成変更に柔軟に対応できるビットマップテーブル仲介型のインタフェースを備えた処理系を提案する．次に，実データからその処理時間の評価式を導く．最後に，適用例，工数削減例に触れ，実用性，有効性を示す．本システムは，自動倉庫システム，基板組み立てショップ，鉄鋼プラント等多くに実適用されており，制御ソフト開発工数の削減およびメンテメンスに効果を上げている．

1. まえがき

ジョブショップにおける自動搬送，鉄鋼・自動車等の自動生産，倉庫システムのコンベア・クレーン等の自動運転といった自動化システムでは，扱う製品の寿命の短期化，種類の多様化に伴い，制御ソフトの迅速な開発，変更が要求されている．

このような自動化システムの制御では，ワーク・処理要求の待ち状態，棚・中間品置場の使用状態，設備の動作状態といったシステムの個々の要素の状態の組み合わせに応じ制御を決定する．すなわち，システムがどのような状況にあるかを判別し，その状況において適切な次の動作を決定するという状況判別型の制御が行われる．

従来の自動化システムでは，このような状況判別型制御を実現するプログラムを，FORTRAN, PL/I 等の手続き型言語を用いて開発する方法が採られている．この場合，どのようなシステム状態の組み合わせに対してどういった動作を行うという制御仕様を検討し，さらに，仕様を実現するプログラムの構造・処理フローを設計し，コーディング・デバッグを行うという手順を踏むことになる．しかし，この方法では，(1)システム状態の組み合わせと制御動作との単なる対応という非手続き的内容を，システム状態を判別し動作の決定に至る手順という手続きに置き換えなければならない，(2)制御仕様の非手続き性とその実現プログラムの手続き性のギャップのため，制御ソフト変更時に，仕様を反映している箇所の理解と，新仕様追加箇所の特定が困難といった問題があり，制御ソフトの開発，変更に時間と工数を要している．

以上の問題を回避するには，処理手順を意識することなく，状況判別型制御の仕様を列挙するだけで制御プログラム化できる方式が心要となる．これに対し，近年，ばらばらに記述した判断のルールを必要に応じて自動的に組み合わせ所望の結論を得るという人工知能におけるプロダクションシステム[1),2)]の手法が注目されている．この手法によれば，システム状態の組み合わせと制御動作の対応を，そのままの形で，ルールとして宣言するだけで，推論により自動的にシステム状態を判別し制御動作を決定できるので，状況判別型制御に十分効果的に使用可能なことが期待される．

従来，このようなプロダクションシステムの手法は，診断[3),4)]，事故の復旧支援[5)]，運用指示[6)]，計算機システム構成決定支援[7)]等のコンサルテーションの場で用いられてきた．これらのコンサルテーションシステムでは，人間を相手とすることから，通常，単一の CRT ターミナルが外部とのインタフェースとして使用されるため，接続されたターミナル専用の文字の入出力等の単純なインタフェース手続きを装備しておけば十分であった．また，処理時間もさほど問題にされ

† Development of a Rule-Based Control Software System SCD (Station Coordinator) by TSUTOMU TASHIRO, NORIHISA KOMODA, ISAO TSUSHIMA and KUNIAKI MATSUMOTO (Systems Development Laboratory, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所

なかった．しかし，プロダクションシステムを，制御に，一種のプログラム言語として用いようとする場合，以下の点が新たな問題となる．

（1） 制御においては，種々のセンサ・アクチュエータがインタフェースとして使用される．これらは，制御対象ごとに様々な構成で配置され，また，仕様変更に応じて追加，削除される．プロダクションシステムの柔軟性を損なうことなく，各種センサ・アクチュエータの構成およびその変更に柔軟かつ迅速に対応できる汎用プロセスインタフェース方式が要求される．

（2） 実時間制御に必要な応答速度を確保できる処理系の開発が必要．

（3） 自動化システムの設計段階では，制御計算機システムの構成を決定するために，各種タスク CPU 負荷率の推定が必要となる．また，要求された制御応答速度をクリアできるかの見積りも重要である．これらのために，与えられた制御ルールに対する処理時間の予測方式の確立が要求される．

以上については，従来，その実現方法についてほとんど報告されていない．

筆者らは，今回，これらの要求を満足し得る状況判別型制御用プロダクションシステム型制御核ソフト（ルール型制御ソフトウェアシステム）を開発し，実際の自動化システムの実時間制御に適用した．その結果，制御用としての十分な実用性，有効性を確認したので報告する．まず，状況判別型制御の枠組みにおいて上記(1)，(2)の要求をさらに明確にし，これに応える，作成，変更の容易なルール記述，ビットマップテーブルを仲介とする高柔軟プロセスインタフェース，これらを高速に処理する処理系から成るルール型制御ソフトウェアシステムの構成を示す．次に，実測データおよび実システム制御用ルールから導いた，本ソフトウェアシステムを用いる制御プログラムの処理時間評価式を示し，与えられたルールに対する処理時間の予測方法を明らかにする．最後に，本ソフトウェアシステムが実際の制御にどのように用いられるかの実例，ソフト開発工数の削減例について触れ，その実用性，有効性を示す．

2. 状況判別型制御用ルール型制御ソフトウェアシステムの課題

本章では，(1)高柔軟なプロセスインタフェース，(2)実時間制御に必要な応答速度に対する要求について，状況判別型制御の観点からさらに述べる．

2.1 プロセスとのインタフェース

自動化計算機システムは，管理レベルに応じて階層構成となる[8]．状況判別型制御は，プログラマブルコントローラ，シーケンサ等の上位の設備統括制御レベルに位置づけられる．このレベルで制御のために参照される情報には，下位レベルからの入力信号，および，制御対象内の物の位置を追跡しておくトラッキング情報等の計算機内データがある．ルール型制御ソフトウェアシステムは，これらの情報をプロダクションシステムの事実として入力し，ルールで推論し，結論された制御動作を指示信号として下位レベルに出力する．ここで，信号の立ち上り，下り間のタイミング調整等の機能は，通常下位レベルで実施され，設備統括制御レベルでは，ほとんどの信号が ON-OFF の2値信号として扱われる．例えば，荷の載荷・通過，オペレータからの確認・起動，機械のモード等の入力信号は，すべて，ON-OFF 信号である．また，コンベアの起動・停止，分岐装置の方向指示，機械のモード切換え等の指示信号もシーケンサ等への ON-OFF 信号として処理される．一方，トラッキングデータ等計算機内の情報についても，状況判別型制御側からは，例えば，コンベア，ステーションにパレットが有り，無し等の ON-OFF 情報として扱われる．ルール型制御ソフトウェアシステムでは，これら ON-OFF 情報とのインタフェースを，プロダクションシステムの柔軟性を阻害することなく容易に取る，あるいは，修正することができる方式が必要となる．

2.2 応答速度

設備統括制御レベルでは，状況判別型制御用タスクのほかに，CRT タスク，トラッキングタスク，帳票出力タスク，上位レベルとの通信タスク等，他の様々なタスクが同一 CPU 内で動作する．ここで，自動化システム全体が停滞なくスムーズに動作するには，経験上 CPU 負荷率を 50% 以下に抑えることが望ましい．これらのことから，トラッキングタスクを除く他タスクがあまり頻繁に動作しないことを考慮に入れても，状況判別型制御用タスクに分ける負荷はせいぜい20数%である．状況判別型制御で実質的に必要なタスク起動が2秒前後に1回であることから，1回の起動に対し消費できる CPU 時間は数 100 ミリ秒程度となる．ルール型制御ソフトウェアシステムでは，システム状態の取り込み，制御指示の出力を含めこの時間で処理を終了できる処理速度が必要である．

3. ルール型制御ソフトウェアシステム

プロダクションシステムの柔軟性は，要変更部分を事実，ルールのデータとして分離し，処理を推論機構として標準化，固定したことによる．ここで，コンサルテーションシステムでは，外部とのインタフェースが専用，固定化されているため，使用ターミナルに合わせた処理手続きをあらかじめ推論機構内に組み込むことができた．しかし，制御では，外部とのインタフェースは専用，固定化できず，対象ごとに可変となる．このため，推論機構内にインタフェース構成に合わせた処理手続きを組み込む方式では，毎回，推論機構の開発，変更が生じ大きな障害となる．これに対処するには，処理手続き，インタフェース構成情報等の可変部分を推論機構から分離し，推論機構としては，これらを標準手順で扱える方式が必要となる．

本システムでは，状況判別型制御では外部情報をON-OFF 情報として扱えることに着目し，ビットマップテーブルを仲介とし，各ビットとルールの条件，結論の表現との対応を別定義で与えるインタフェース方式とする．これにより，推論機構側では，ビットのON-OFF の検査，書き換えというように処理を標準化できる．一方，プロセス側からは，ON-OFF の入出力信号は，何ら処理手続きを介することなく，直接各ビットアドレスにハード接続できるので，ルールとの対応は定義のみで済み，高柔軟な I/O システム を構築できる．また，トラッキングデータ等についても，データをビット情報に変換する簡単な別タスクを設けるだけで，推論機構に影響を与えずに処理可能となる．

3.1 構　　成

ルール型制御ソフトウェアシステムの構成を**図 1** に示す．

オンラインカーネルは，実時間制御を実行する部分で，ビットマップテーブルを介し ON-OFF プロセス信号を自動入出力する．ユーザは，必要ならば，トラッキングデータ，特殊 I/O 等に対し，ビットマップテーブルとの間の入出力手続きを別タスクとして作成する．

本システムによる制御プログラムは，イベント起動型である．処理要求元からイベント No. 付きで起動がかかると，自動的にセンサ信号を判定する．制御動作が決定されるとプロセスに指令信号を発信する．これらの信号とルールの対応づけは，プロセスインタフェース定義で与える．

ルールエディタは，ルール，プロセスインタフェース定義をマンマシンで誤りなく作成，修正できる機能[9] に加え，ルールの論理レベルの誤り検出を支援する推論構造解析，ルール動作シミュレート機能[10]を備えている．詳細は文献参照とするが，これらにより，ルール作成作業の効率化を図っている．

3.2 制御則の記述 (IF-THEN ルール)

状況判別型制御では，発生イベントに応じて判定すべきシステム状態が限られることが多い．また，システム状況を判別していく過程で，もはや判定する必要のないシステム状態が明らかになることがある．この性質を利用し，無駄なルールの検索を排除することにより処理時間の短縮を図ることができる．本システムでは，このため，および，ルールのモジュール化の促進の観点から，ルールを群に分割して登録しておき，必要に応じて選択して推論を進める機能を提供する．このために，推論用ルールに加え，ルール群を選択し推論を制御するためのメタルールを用意する．

ルールの例を**図 2** に示す．判別すべきシステム状態の組み合わせ条件を IF 部に，その状態の組み合わせが満足された時の結論を THEN 部に記述する．個々の条件，結論は，

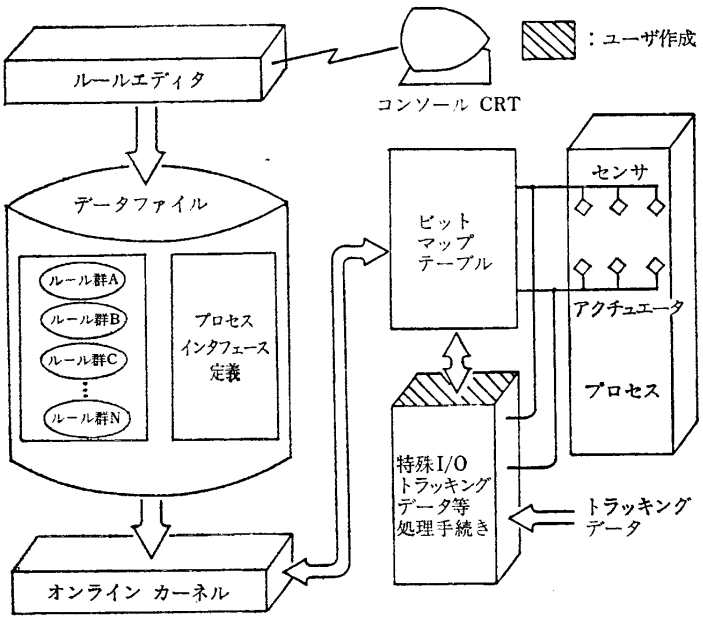

図 1 ルール型制御ソフトウェアシステム
Fig. 1 Configuration of rule-based control software.

```
推論用ルール
(1)  IF     (ステーション<X> ハ アキ)
     IF     (ステーション<X> ハ <Y>サギョウ ショリカノウ)
     IF     (パレット<Z> ノ サギョウ ハ <Y>)
     THEN   (ステーション<X> ハ パレット<Z>
                                  ノ サギョウコウホ)
                                          固定部
                  パラメータ部
(2)  IF     (コンベア<X> ハ リヨウ コウホ)
     THEN   <MIN X>(ワカバンコンベア<X> ヲ リヨウ)
             組み込み手続き
推論制御用メタルール
(3)  IF     (EVENT<28>) イベントNo.
     THEN   SELECT<ASSIGN>
                  ルール群名
(4)  IF     (パレット<X> ハ キンキュウ)
     IF     (パレット<X> ノ サギョウ ハ <Y>)
     IF     (サギョウ<Y> ハ ショリ フカノウ)
     THEN   SELECT<STOPSTAT>
```

図 2　IF-THEN ルールの例
Fig. 2　An example of IF-THEN rules.

任意の日本語文字列でかっこ内に記述する．文字列は，条件，結論の意味を明示する固定部と，その条件，結論の当てはまる対象（設備名，番号等）を示すパラメータ部で構成する．パラメータ部には，定数に加え，変数 (X, Y, Z 等) を記述できる．

推論用ルールは前向き推論で処理する．ここで，IF 部に変数が記述されているとその変数に対応するすべての具体値から，IF 部の AND 条件を満足する値を計算し (JOIN 演算)，結論にあてはめる．本機能により，多くの選択肢の中から，条件によって絞り込み制御動作を決定するという，状況判別の仕様を容易に記述することができる．ルールで記述しきれない数値処理，最適化処理は，手続きとして記述ルールに組み込む．図2(2)の例のように THEN 部に組み込むべき手続き名を指定すると，IF 部の処理の後手続きが Call される．この際，条件を満足する変数値が渡され，手続き内でこの値を任意に変更することで処理結果を結論に反映できる．

メタルールは，上記の観点から，イベント No. に基づき最初に使用するルール群を選択するもの，推論の途中状況により次に使用するルール群を選択するものの 2 種類を用意した．

3.3 プロセスとのインタフェース

本システムでは，前述のように，ビットマップテーブルを仲介とする方式とした．オンラインカーネルは，推論の過程でルールの IF 部でシステム状態のチェックが必要となると，このテーブルの対応ビットの 0, 1 を自動的に判定する．また，THEN 部で制御動作が結論されると，その内容に従い，対応ビットを 0, 1 にセットする．このことにより，プロセスとの間で信号が自動的に入出力され，ユーザは，ルールを作成するのみで制御を実施することが可能となる．

ルールの IF 部の条件，THEN 部の結論と各ビットとの対応づけは，**図 3** の例に示す，プロセスインタフェース定義にて行う．始めに，入力か出力かを指定し，その後に，ルール側のシステム状態あるいは制御動作の表現とビットマップテーブル内のビット位置の対応づけを行う．ビット位置は，行と列の組み合わせで指定する．オンオフ指定は，入力では，ビットが指定された値であれば定義状態が真であることを意味し，出力では，定義した制御動作が結論されるとビットをその値にセットすることを意味する．ユーザは，本定義を入れ換えるのみで，システム構成の変更に容易に対応できる．

3.4 推論機構

3.4.1 ソフトウェアアーキテクチュア

オンラインカーネルでは，前述の，ルールを群に分割し無駄なルール検索を排除する高速化に加え以下の工夫を行った．

ルールの文字列の固定部は，オンラインカーネルにとっては文字列どうしを区別するための情報でしかない．パラメータ部の値のみが制御動作決定に必要となる．この点から，固定部をコード化し，同一コードごとにパラメータ部の値をポインタで結合し管理する方式[11]とした．これにより，処理に必要な情報を高速に取り出すことができる．さらに，ルール群内のルール

(1) 入出力指定：I
　　状態表現：(ステーション〈1〉 ハ アキ)
　　ビット位置：28, 1
　　オンオフ指定：0
(2) 入出力指定：O
　　制御動作表現：(ステーション〈2〉 サギョウ カイシ)
　　ビット位置：12, 2
　　オンオフ指定：1

＿＿：ユーザ入力

図 3　プロセスインタフェース定義の例
Fig. 3　An example of process interface definition.

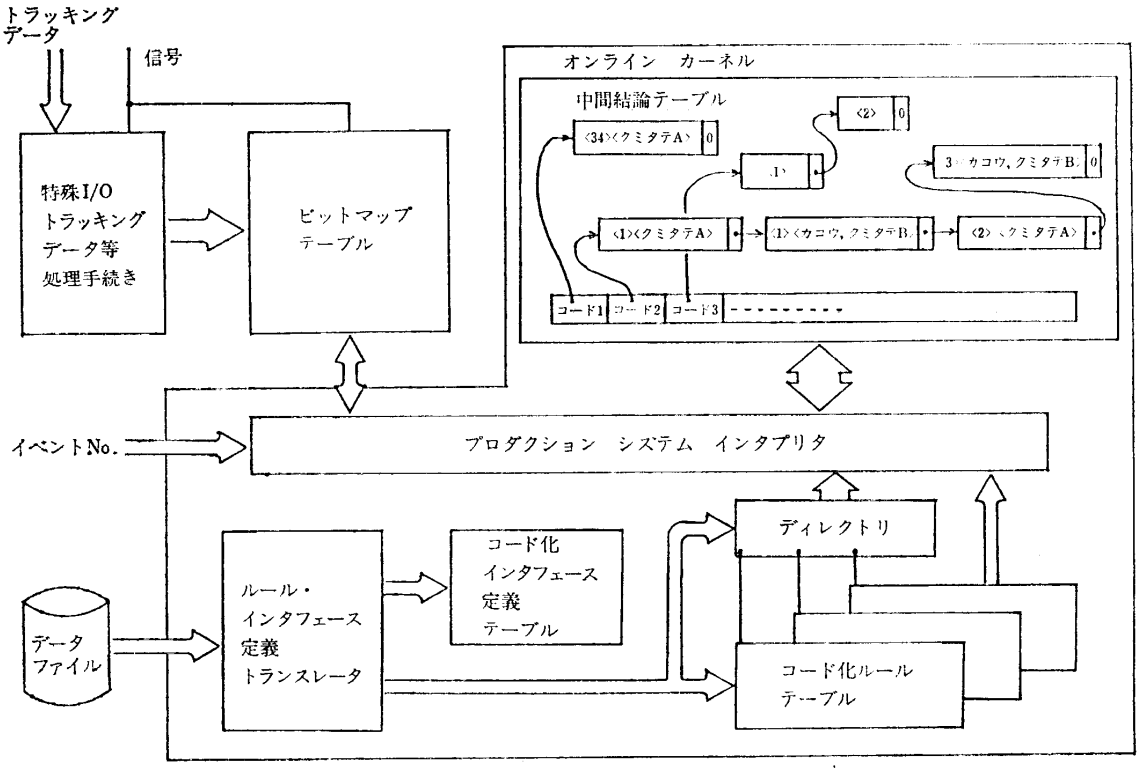

図 4 オンラインカーネルのアーキテクチュア
Fig. 4 Architecture of on-line kernel.

の処理順序の最適化を行っている．すなわち，ルール間の IF 部と THEN 部のつながりを解析し，無駄に IF 部の判定を行わぬよう並べ換えを行う．また，1 ルール内の複数の条件において，満足されそうもない条件を先に処理し，無駄な時間を消費しないよう並べ換える．

オンラインカーネルのアーキテクチュアを**図 4** に示す．ワークメモリは，中間結論テーブル，ビットマップテーブルである．ビットマップテーブルには，前述のように，システム状態，決定された制御動作が書かれる．中間結論テーブルには，推論途中の結論が書かれる．プロダクションメモリは，ディレクトリ，コード化ルールテーブル，コード化インタフェース定義テーブルから成る．ルールは，群ごとにディレクトリによって管理される．プロダクションシステムインタプリタは，推論時，コード化インタフェース定義テーブルの情報に基づき，ビットマップテーブル，中間結論テーブルのどちらをアクセスするかを判断する．ルールのコード化，処理順の最適化はルール・インタフェース定義トランスレータにおいて行う．

3.4.2 処 理 速 度

オンラインカーネルの起動から制御動作決定までの時間は，ルール群の分割方法，ルールの書き方，起動時のシステム状態によって異なる．すなわち，発生イベントの種類，システム状態により，検索されるルール，および，チェックされる条件の数が異なってくる．ここでは，ルール処理時間予測の基礎データとするために，与えられたすべてのルールおよびそのすべての条件チェックが行われ，制御動作が決定されるという状況のもとでの処理速度を示しておく．なお，現実のルールに対する処理時間の予測は次章で述べる．

ルールの処理は，IF 部の条件チェック，条件満足時の結論処理の繰り返しである．このことから，処理時間は，チェックすべき条件数および，結論を下す回数すなわち推論の深さに依存して増大すると考えられる．推論の深さをパラメータとして，IF 部の総条件数と処理時間の関係を実測した結果を，**図 5** に示す．測定は，8 MHz の M 68000 計算機システム，C言語によるものである．なお，現実のルールの処理時間予測において変数を含むルールは，幾つかの定数のみルールに変換して扱えるので，IF 部の条件として定数のみが記述された場合で測定した．処理時間は，IF 部の総条件数に比例することが分かる．また，実際には，推論の深さはルールの分かりやすさの点からも 6

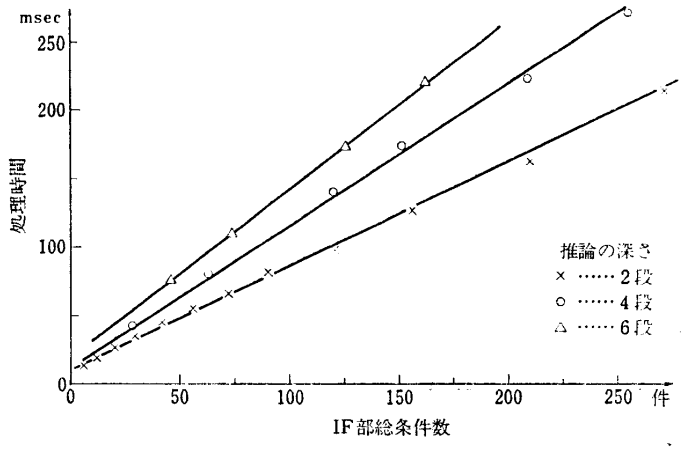

図 5 オンラインカーネルの処理速度
Fig. 5 Inference speed of on-line kernel.

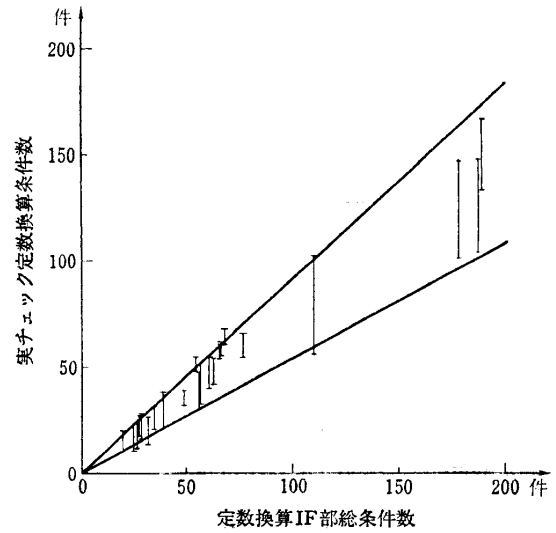

図 6 現実ルールで実際にチェックされる条件の割合い
Fig. 6 Ratio of actual condition check.

段程度が限度であることから処理時間に対し大きな影響とならないことが分かる.

本オンラインカーネルは，推論の深さ4段で 0.96 msec/条件という処理速度を達成しており，現実ルールでは1回の起動でチェックされる条件数が後のグラフで示されるように 100 前後，多くても 200 程度であることから前記要求速度を十分満足していると言える.

4. ルール処理時間の予測

自動化システムの設計では，自動化機器の構成を検討するとともに，これらを制御する計算機構成を決定しなければならない．計算機構成の決定では，処理すべきタスクの内容を整理し，下位レベルのシーケンサ等との負荷分担を考慮しつつ CPU の負荷率を予測し，計算機のグレード，台数を決定する．また，計算機システムが当初の要求性能を満足し得るかの確認が必要となる．従来これらは，開発プログラムステップの見積り，経験から導かれたダイナミックステップの計算式等によって行われている．ルール型制御ソフトウェアシステムにおいても，同様にルールの見積りから処理時間を推定する予測式が要求される.

実際のルールでは，発生イベントの種類，システム状態により，検索されるルールおよびチェックされる条件数が異なってくる．ここで，状況判別型制御では，イベントの種類により使用されるルール群を特定できるので，検索されるルールの総条件数についてはあらかじめ予測可能である．本章では，実際に自動化制御に使用されたルールを用い，記述された総条件に対しどの程度の割合で実際の条件チェックが行われるかを分析し，これと，3.4 節の実測値から処理時間の推定式を導く.

現実のルールにおいて，発生イベント，システム状態ごとに，実際にチェックされる条件数を調査し，その時に使用されたルール群のルールの総条件数との関係を分析した結果を**図 6** に示す．横軸に総条件数，縦軸にチェックされた条件数を示してある．なお，実際のルールでは，幾つかの定数記述のルールを変数を用い1ルールとして表現する場合があるが，これに対しては，定数ルールで記述した場合に換算してある．図6から，実際にチェックされる条件の割合は約 92～55%の幅でほぼ一定であることが分かる．幅があるのは，少ない条件のチェックで結論が下せる場合と，その条件が満足されない場合，さらに別の条件をチェックして結論を下す場合とがあるからである.

以上の結果および図5の実測値から次の推定式が導かれる．なお，現実のルールでは推論の深さは4段以下がほとんどなので，この時の処理速度を用いてある.

$$\text{平均処理時間} = 0.74 \times 0.96 \times \text{定数換算総条件数} \times \alpha \quad [\text{msec}] \qquad (1)$$

$$\text{最大処理時間} = 0.92 \times 0.96 \times \text{定数換算総条件数} \times \alpha \quad [\text{msec}] \qquad (2)$$

上式において，α は CPU ごとの補正係数であり，実際に使用する計算機システムと今回実測に用いた 8 MHz の M 68000 計算機システムの CPU 処理速度の比率である．設計では，まず，ルール群構成，各ルール群でチェックすべき総条件数を見積り，上記(1)，

(2)式からそれぞれ，CPU 負荷率，最悪応答時間を予測しており，良好な結果を得ている．

5. 自動倉庫システムへの適用例と評価

製品の多様化傾向から，倉庫は，単なる購入品，素材の保管の場から，生産工場内の中間部品，仕掛品の管理の場へと移行し，自動化工場の一コンポーネントとして機能するようになった[12]．

この場合，周辺システムとの連続性を損なわぬよう，自動倉庫は，エレベータ，コンベア，自動搬送台車等の周辺機器と組み合わされシステムとして制御される．当然，倉庫の自動化工場内への組み込み形態は多様であり，システム構成，制御方法は顧客ごとに異なる．制御ソフトは毎回これに合わせて作成されている．また，周辺機器の変更ごとに修正される．

ルール型制御ソフトウェアシステムは，このような

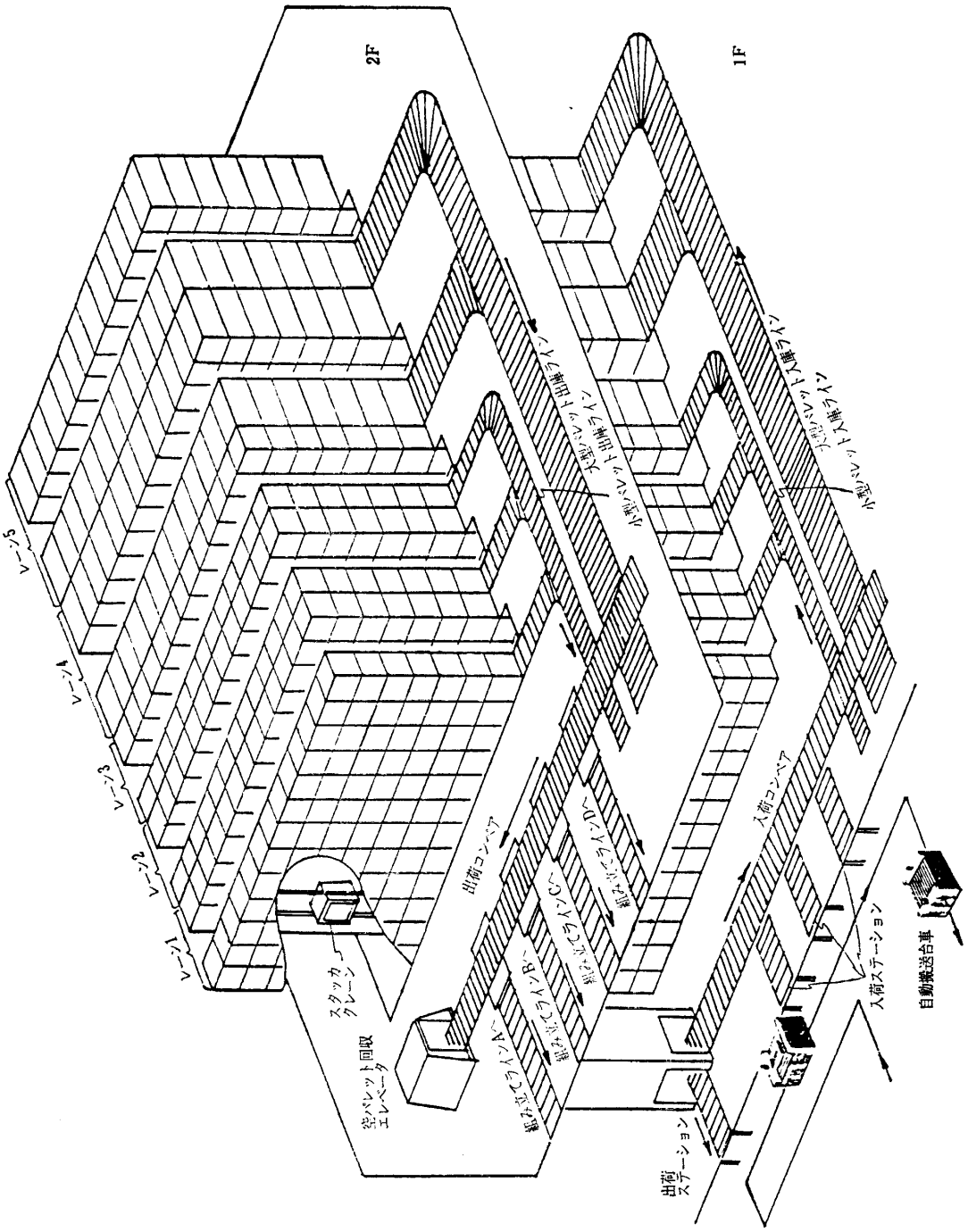

図 7 自動倉庫システムの例

Fig. 7 An example of automatic warehouse system.

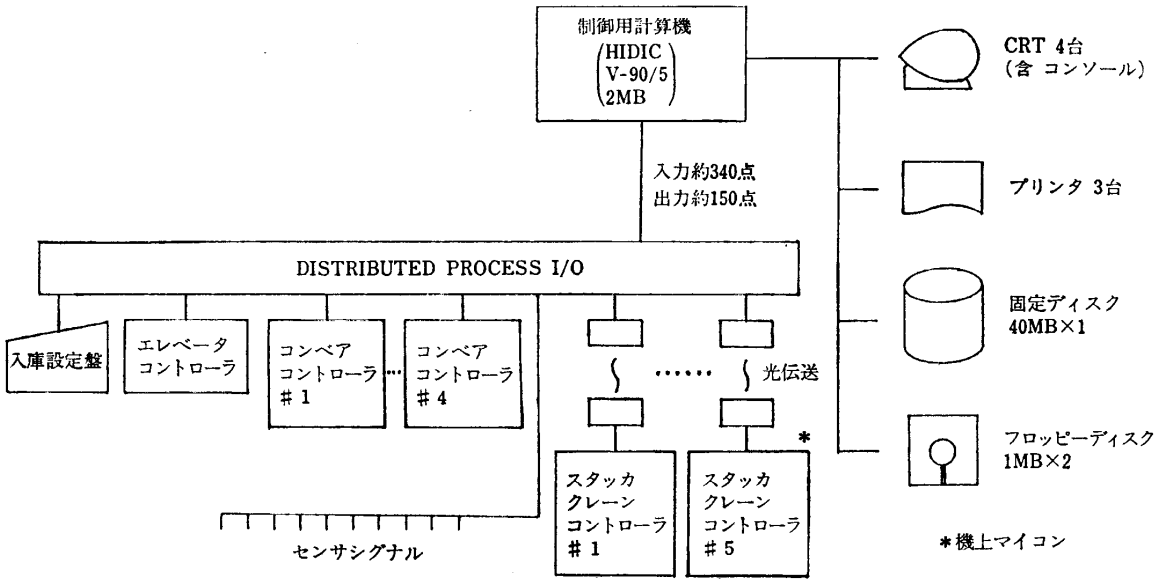

図 8 制御用コンピュータ構成例
Fig. 8 An example of control computer configuration.

自動倉庫システムの制御ソフトの開発，変更に効果的に使用されている．以下では，実例により，その実用性，有効性を示す．

5.1 自動倉庫システム

図7に，ある組み立て工場内の自動倉庫システムの例を示す．

部品は，パレットに入れられ自動搬送台車で入荷ステーションへ運ばれて来る．ここで，パレットには，大型，小型があり，それぞれ，大型倉庫（レーン4，5），小型倉庫（レーン1，2，3）に仕分けして保管される．入荷されたパレットは，入荷コンベアにより，大型／小型パレット入庫ラインに供給される．

部品の出庫は，各組み立てラインからの要求によって行う．要求のあった部品を乗せたパレットは，大型／小型パレット出庫ラインから出庫され，出荷コンベアに供給される．

各組み立てラインのコンベアにて荷降ろしした空パレットは，空パレット回収エレベータにて1階に降ろされ，出荷ステーションに戻される．この際，自動搬送台車による回収が間に合わず，あふれた空パレットは，再び入荷コンベアを通し，一時倉庫内に保管しておき，状況に応じて出荷コンベアを通し回収する．

5.2 制御計算機システム構成

前節の自動倉庫システムの制御用計算機構成を**図8**に示す．全体の制御は M 68000 ベースの制御用計算機 HIDIC V-90/5 で行っており，プロセスとは，分散型プロセス I/O 装置を介して結合されている．

2階の各組み立てラインからの部品出庫要求は，出庫設定器で行う．エレベータ，コンベアコントローラは簡単なシーケンサである．以上までに対する入出力，および，センサ信号はすべて ON-OFF 信号であり，ビットマップテーブルと直接接続されている．

スタッカクレーンの動作制御は，機上のマイコンベースのコントローラによって行われている．クレーンコントローラとは，シリアルな光伝送で接続される．この I/O とビットマップテーブルとの間の情報交換は，手続きによっている．

CRT，プリンタは，システムのモニタ，帳票出力等に使用される．

5.3 制御ルール

ルール型制御ソフトウェアシステムは，前記自動倉庫システムにおいて，コンベア，エレベータの制御，スタッカクレーンへの作業指示といった，ほとんどすべての自動化制御に使用されている．

制御ルールのルール群構成および例を**図9**に示す．

イベント振り分けルール群は，台車の到着，パレットの載荷，スタッカクレーンの動作終了等のイベント発生に従って，どのルール群を使用するかの選択を行う．

入荷ステーション決定ルール群は，次に入荷するステーションを決定し，パレットを入荷コンベアに押し出す指令を発信する．押し出されたパレットは，入荷コンベア制御，入庫パレット判定，大型／小型パレット入庫ライン制御のルール群の指令により，倉庫の入

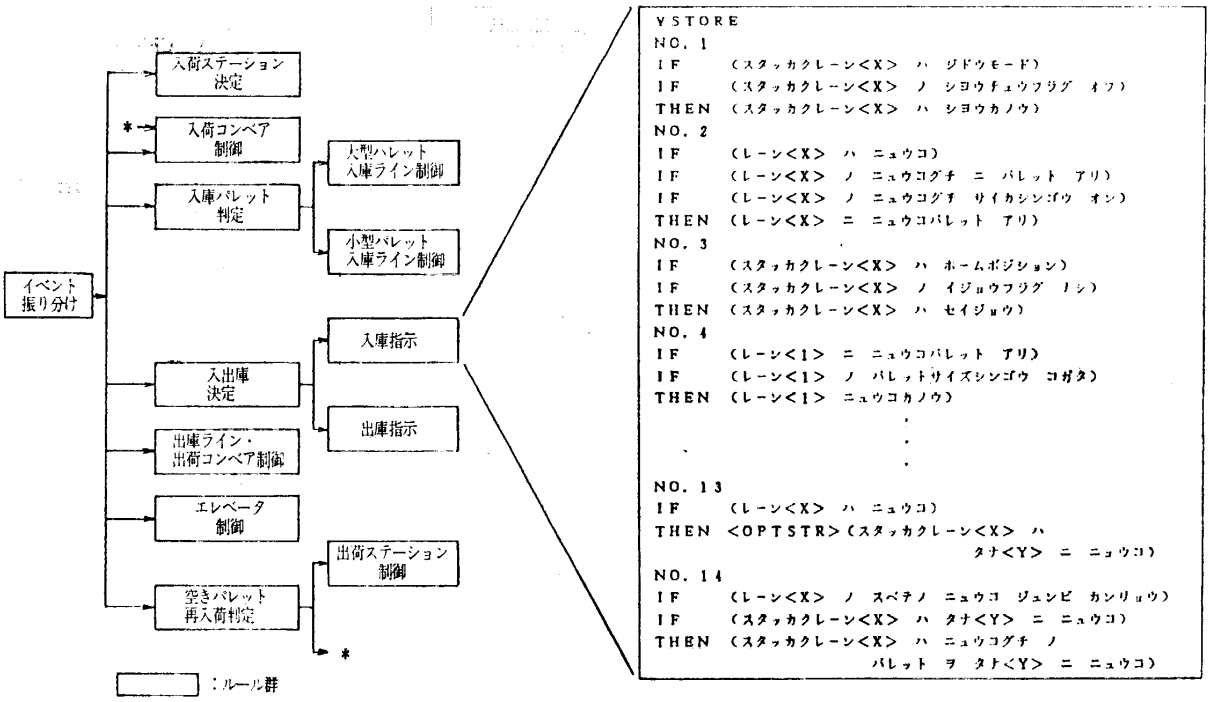

図 9 制御ルール群構成およびルール例
Fig. 9 An example of overall rule-group configuration and rules.

庫口まで移動する．どの入庫ラインのどの入庫口へ運ぶかは，パレットの種別，入庫口の滞留状態等から決められる．

スタッカクレーンに対する，入出庫の指示は，入出庫決定，入庫／出庫指示ルール群で行う．組み立てラインからの出庫要求，入庫口の滞留状態からどのクレーンが何をするかを決定する．

出庫されたパレットは，出庫ライン・出荷コンベア制御ルール群の指令で，出庫要求のあったラインに供給され，空パレットは，エレベータ制御ルール群の指令で1階に降ろされる．

降ろされた空パレットは，空パレット再入荷判定，出荷ステーション制御，前述の入荷コンベア制御ルール群の指令で，出荷あるいは再入荷が行われる．

以上，延べ約600定数換算条件の判定を行う約80のルールが使用された．

5.4 効　　果

現在，自動倉庫システムの制御においては，トラッキングデータ，特殊 I/O が標準化されてきており，ビットマップテーブルとの間の情報交換手続きも整備されている．自動倉庫システムでは，これらを使用し短期間に顧客ごとの制御ソフトを開発することが可能となってきている．

実際の経験では，本システムおよび標準化モジュールを使用することで，従来に比べプログラム開発量，開発工数を 1/3 に削減できており，有効性を確認している．

なお，図9のルールでは，1イベントに対し平均約80定数換算条件が処理されるが，この時の平均処理時間の予測値は，式(1)から，56.8 msec となる．一方実測結果では，平均 52.9 msec の処理時間となっておりよく一致している（実測は 8 MHz M 68000 計算機システムによった）．

6. むすび

プロダクションシステムの手法をベースとして，制御ソフトの開発，変更を容易かつ迅速に行えるルール型制御ソフトウェアシステムを開発した．本システムは，例に示した自動倉庫システムのほかにも，多くの自動倉庫システムの制御，製鉄所のビレット精整ラインの制御[13]，基板組み立てショップの台車制御等の実際のシステムに適用されており，制御ソフト開発工数の削減，その後の拡張，メンテナンスの容易性といった効果を上げている．

謝辞　本研究に関してご討論いただいた東京工業大学市川惇信教授に深く感謝の意を表します．また，本研究の機会を与えていただいた，(株)日立製作所情報事業本部本部長常務三浦武雄博士，同システム開発研

究所所長川崎淳博士，並びに本研究のご指導をいただいた同システム開発研究所第1部部長春名公一博士に対し感謝いたします．また，本システムの開発および実システムへの適用に関しご討議，ご協力いただいた同大みか工場，笠戸工場，神奈川工場並びにシステム開発研究所の各位に感謝いたします．

参考文献

1) Nilson, N. J.: *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto (1980).
2) Barr, A. and Feigenbaum, E. A.: *The Handbook of Artificial Intelligence*, William Kaufman Inc., Los Altos (1981).
3) Shortliffe, E. H.: *Computer-Based Medical Consultations: MYCIN*, American Elsevier, New York (1976).
4) 元田ほか：システムの機能に関する知識を用いた原子炉異常診断の試み，計測と制御，Vol. 22, No. 19, pp. 791-796 (1983).
5) 松本ほか：知識ベースに基づく電力系統復旧方式の決定法，電気学会論文誌 B, Vol. 103-B, No. 3, pp. 175-182 (1983).
6) 荒屋ほか：ヒューリスティクスを利用した列車運転整理手法，電気学会論文誌 C, Vol. 103-C, No. 11, pp. 249-256 (1983).
7) McDermott, J.: R1: A Rule-Based Configurator of Computer Systems, *Artif. Intell.*, Vol. 19, No. 1, pp. 39-88 (1982).
8) Komoda, N. et al.: An Autonomous Decentralized Control System for Factory Automation, *IEEE Comput.*, Vol. 17, No. 12, pp. 73-83 (1984).
9) 薦田ほか：ルールベースコントローラ用ルール入力画面エディタ，情報処理学会第29回全国大会予稿集，pp. 1249-1250 (1984).
10) 田代ほか：ルール型制御システム SCD 用ルール作成支援ソフトウェア，情報処理学会第31回全国大会予稿集，pp. 981-982 (1985).
11) 田代ほか：ルールベースコントローラにおける推論機構，情報処理学会第26回全国大会予稿集，pp. 1049-1050 (1983).
12) 高橋：倉庫の自動化設計，日刊工業新聞社，東京 (1971).
13) 都島ほか：流れ作業ライン制御へのルール型制御方式の適用—製鉄所のビレット精整ライン制御への適用—，計測自動制御学会論文誌，Vol. 21, No. 10, pp. 1113-1120 (1985).

(昭和60年9月27日受付)
(昭和61年2月20日採録)

田代　勤（正会員）

昭和28年生．昭和51年東京工業大学工学部制御工学科卒業．昭和53年同大学大学院総合理工学研究科システム科学専攻修士課程修了．同年(株)日立製作所入社．システム開発研究所にて，ソフトウェア開発技法，事象駆動型システム制御に関する研究に従事．昭和60年9月より，英国 Edinburgh 大学 Artificial Intelligent 学科に客員研究員として留学中．IEEE，計測自動制御学会各会員．

薦田　憲久（正会員）

昭和25年生．昭和47年3月大阪大学工学部電気工学科卒業．昭和49年3月同大学院修士課程修了．同年，(株)日立製作所に入社，システム開発研究所にてシステム計画技法，システム構造化技法，FA システム制御技法などに関する研究に従事．現在，同所第1部主任研究員．昭和56年8月～昭和57年8月 UCLA Engineering Systems 学科に留学．工学博士．IEEE，電気学会，計測自動制御学会などの会員．

都島　功（正会員）

昭和20年生．昭和43年名古屋工業大学計測工学科卒業．昭和45年大阪大学基礎工学部制御工学科修士課程修了．同年(株)日立製作所入社，中央研究所配属．昭和48年システム開発研究所の発足に伴い転属，現在，第一部主任研究員．一貫して，物流システムの計画，管理，制御技法に関する研究に従事．IEEE，電気学会，計測自動制御学会各会員．

松本　邦顕（正会員）

昭和18年生．昭和43年3月九州工業大学制御工学科卒業．昭和45年3月同大学院修士課程修了．同年(株)日立製作所入社，日立研究所配属．昭和48年2月同社システム開発研究所の発足に伴い転属，現在，第一部主任研究員．上下水道およびダムなどの水管理制御システム，ビル空調およびソーラなどの省エネルギー管理制御システム，半導体プロセス制御システム，FA 制御システム等の研究に従事．工学博士．IEEE，電気学会，電子通信学会各会員．