〔非公開〕

ＴＲ－Ｃ－０１３４

実行順序解析方式

小林　吉純
Yoshizumi KOBAYASHI

１９９６　３．６

ＡＴＲ通信システム研究所

実行順序解析方式

平成８年３月

小林吉純
ATR通信システム研究所

目次

1．機能

（1）機能概要

実行順序解析とは，仕様内の文が持つデータの流れを解析し，文の実行順序を生成するものである．具体的には解析前ファイル（Ｇｓｒｃ）を読み込み，グラフ理論に基づいて処理手順を解析し，解析結果ファイル（Ｇｓｏｒｔ）に処理の順番を出力する．

図１　概要図

ここでは，「ａにｂを転記する」などの一つの処理手続きの単位を処理ユニットと呼ぶ．処理ユニットに処理ユニット番号，レコード先頭区切り，入力データ項目（以下，入力データフィールドと呼ぶ），出力データ項目（以下，出力データフィールドと呼ぶ），各フィールド間の区切り，コメント記述項目（以下，コメントフィールドと呼ぶ），終端記号を付加し，形式化したものをレコードと呼ぶ．また，１件以上のレコードをハードディスク等の媒体に格納したものをファイルと呼ぶ．

（2）レコード構成

レコードの記述形式を，図２に示す．終端記号については，図２以降では特に明記しないが，レコードの最終フィールドの直後に必ず在るものとする．

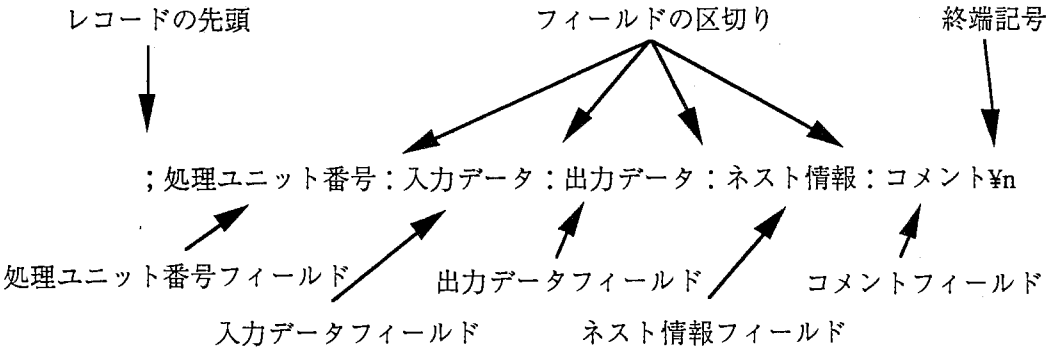


図２　レコード構成図

入力データフィールド，出力データフィールド，ネスト情報フィールド，コメントフィールドのそれぞれにおいて，値が無い場合はそのフィールドには何も記述せず，直前の”：（フィールドの区切り）”に続けて次のフィールドの区切

りを記述する．上記以外のレコードを構成する要素は，省略してはならない．例として，入力データ，出力データが共に無かった場合は次のようになる．

；処理ユニット番号：：：ネスト情報：コメント

（a）レコードの先頭

”；（セミコロン）”を記述する．

（b）処理ユニット番号フィールド

各レコードを識別するためのユニークな番号を記述する．記述例を以下に示す．

；1：入力データ：出力データ：ネスト情報：コメント
；2：入力データ：出力データ：ネスト情報：コメント
；3：入力データ：出力データ：ネスト情報：コメント

（c）フィールドの区切り

”：（コロン）”を記述する．

（d）入力データフィールド

図３のようなデータ構造があり，取引番号を入力データにしたい場合，データ１にもデータ２にも取引番号という項目があるので，取引番号と記述した場合，どちらの方なのか明確でない．そこで，取引記録ファイルの取引番号の場合は，”取引記録ファイルー記録レコードー取引番号”の様に，該当データを包含するものを”ー”でつなぎユニークなデータであることを明記する．

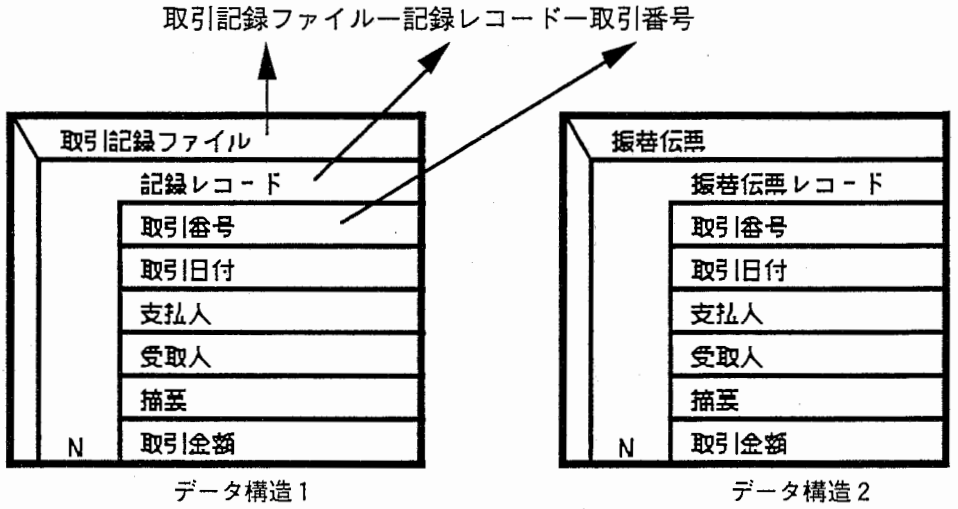


図３　データ構造図

・データが１件の場合

；1：取引記録ファイル-記録レコード-取引番号：出力データ：ネスト情報：コメント

入力するデータが複数の場合，１つ１つのデータは”/（スラッシュ）”で区切る．

・データが複数の場合

;1:取引記録ファイル-記録レコード-取引番号/取引記録ファイル-記録レコード-摘要：出力データ：ネスト情報：コメント

（ｅ）出力データフィールド

このフィールドに記述するデータは，入力データフィールドと同じ書式とする．

入力データフィールドで使用した，データ構造（図３）を使い，使用データも同じで，それが出力データであった場合の記述例を以下に示す．

・データが１件の場合

;1:入力データ：取引記録ファイル-記録レコード-取引番号：ネスト情報：コメント

・データが複数の場合

;1:入力データ：取引記録ファイル-記録レコード-取引番号/取引記録ファイル-記録レコード-摘要：ネスト情報：コメント

（ｆ）ネスト情報フィールド

当フィールドには，　処理ユニットが条件（ＩＦ）文もしくは，反復（ループ）文に包含されない場合は，何も記述しない．それらに包含される場合にのみ記述する．

条件（ＩＦ）文の場合は，ＩＦ－ＴＨＥＮとＥＬＳＥに分けて，それぞれ独立したレコードとして記述する．

処理の制御構造の深まり（=階層）を，”/（スラッシュ）”で区切り列挙する．

つまり，”/（スラッシュ）”の数が多いほど条件もしくは，反復の制御構造が深いことを表す．図４からネストフィールド情報に注目したものを次に示す．

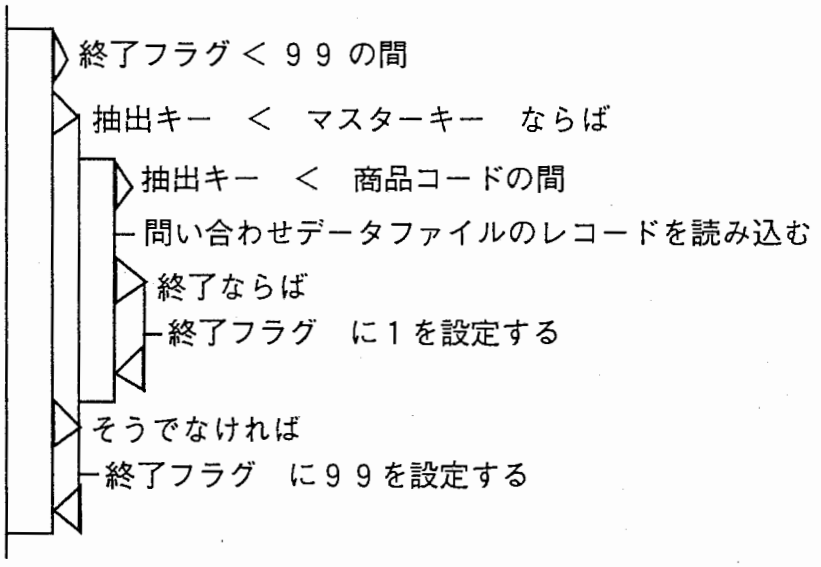
記述が右側に伸びる分量が同じものが同じ階層となる．この例では，２行目と７及び８行目は同じ階層である事が分かる．

```
;1:::LOOP1:
;2:::LOOP1/IF/THEN:
;3:::LOOP1/IF/THEN/LOOP2:
;4:::LOOP1/IF/THEN/LOOP2:
;5:::LOOP1/IF/THEN/LOOP2/IF/THEN:
;6:::LOOP1/IF/THEN/LOOP2/IF/THEN:
;7:::LOOP1/IF/ELSE:
;8:::LOOP1/IF/ELSE:
```

この時，条件（ＩＦ）文の”ＩＦ”，”ＴＨＥＮ”，”ＥＬＳＥ”はキーワードとしているのでこの様に記述する．それ以外の文字列を反復（ループ）文として判断するので，階層によって名称を変えれば，特に記述指定はなく，使用者が

それと分かりやすく記述すればよい.

この記述例では，階層の浅い方をＬＯＯＰ１とし，深い方をＬＯＯＰ２とした.




```
;1:終了フラグ::LOOP1:終了フラグ ＜ ９９ の間
;2:抽出キー/マスターキー::LOOP1/IF/THEN:抽出キー ＜ マスターキー ならば
;3:抽出キー/商品コード::LOOP1/IF/THEN/LOOP2:抽出キー ＜ 商品コードの間
;4::問い合わせデータファイル:LOOP1/IF/THEN/LOOP2:問い合わせデータファイルのレコードを読み込
;5:::LOOP1/IF/THEN/LOOP2/IF/THEN:終了ならば
;6::終了フラグ:LOOP1/IF/THEN/LOOP2/IF/THEN:終了フラグ に１を設定する
;7:::LOOP1/IF/ELSE:そうでなければ
;8::終了フラグ:LOOP1/IF/ELSE:終了フラグ に９９を設定する
```

図４　ネスト情報記述例

（ｇ）コメントフィールド

拡張用のフィールドであり，現在は処理の実行にはなんら関与せず，記述の省略も可能である．図４の例では，処理の内容の覚え書きに使用している.

（ｈ）終端記号

”¥n（改行）”をレコードの終端とする.

改行キーを押下する事により，終端記号を入力したこととなる．入力や出力の記述量が多くなると，１レコードが複数行に渡ることもありえるが，改行キーを押下するまで，１レコードとなる.

2．解析方法

（1）ユニットデータの作成

仕様に記述された処理ユニットの並びを有向グラフとして扱うために，各処理ユニットごとにユニットデータを作成する．このデータの様式は前記の通りで，入出力データをエンティティ，処理ユニットをコンストレイントとして扱っている．

データ項目	→	エンティティ
データ項目番号	→	エンティティ番号
ユニット	→	コンストレイント
ユニット番号	→	コンストレイント番号
解析前実行順序	→	解析前実行順序
入出力項目	→	入出力エンティティ
入出力端子	→	入出力有効枝

各処理ユニットにユニークなユニット番号をコンストレイント番号とし，その入出力データ毎にユニークな番号をエンティティ番号とする．この結果，各ユニットのコンストレイント番号とその入出力データのエンティティ番号を節点としてグラフ解析を行う．

（2）基本的考え方

「入力エンティティ→コンストレイント→出力エンティティ」を1つの単位としてコンストレイントの順序を決定する．すなわち，あるコンストレイントの出力エンティティを，入力エンティティとする他のコンストレイントをサーチして，マッチするものを次のコンストレイントの候補とする訳である．このように，グラフ解析の基本的な考え方は，点在するコンストレイントまたはエンティティから，他のコンストレイントまたはエンティティに有向枝を全て描いてしまうことである．従って，実行順序の決定は，あるコンストレイントの入力エンティティを出力エンティティとして持つ他のコンストレイントが先に実行されることを見つけ出すことになる．

ところで，先頭のコンストレイントを決定する条件は，そのコンストレイントの入力エンティティが他のコンストレイントの出力でないということである．よってそのコンストレイントより先に存在するコンストレイントがないことが確認できたなら，そのコンストレイントの出力エンティティをベースに，次のコンストレイントの実行順序決定へ移る．ただし，先頭のコンストレイントとして複数の候補がある場合，その順序は特定できない．

（3）階層構造の取り扱い

仕様では，ループ文，条件文などによってブロックが形成され，全体としては階層的な木構造が作られている．従って，グラフ解析では，深い階層のブロックから順に解析を行ない，その解析が終了したブロックを仮想ユニットとして上層のブロック内のユニット群に含めて実行順序を解析する．

また，仮想ユニットは，そのブロック内のユニットの入力データ項目，出力データ項目のそれぞれの総和を入力データ項目，出力データ項目とするユニットとして扱う．

（4）単一方向閉路の処置

データの更新を行う場合に単一方向閉路がしばしば発生する．すなわち，同じデータ項目を入力と出力の両方に持つ処理ユニットが存在し，これをERモデルに変換すると，エンティティとコンストレイントとの間で互いに入力有向枝，出力有向枝をひくことができないことになる．

本解析では，元の記述順序を優先することとしたので，当該処理ユニットの同一入出力データ項目にそれぞれ異なる名称を与え，以下のユニットの入力データ項目には，当該ユニットの出力データ項目に与えた名称を使用することとする．

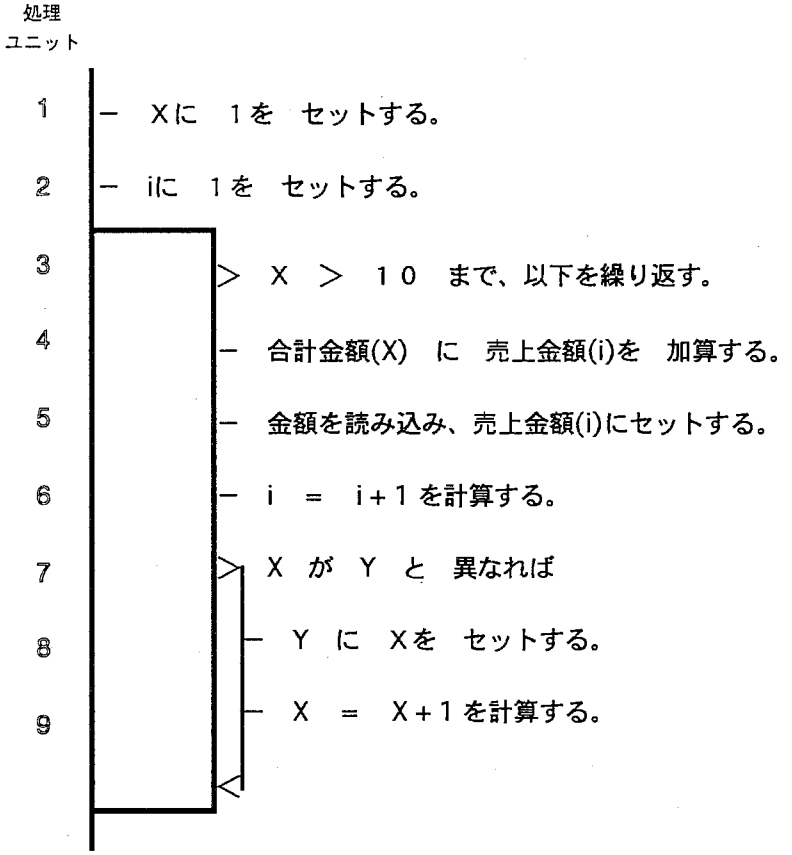
具体的には，グラフ解析情報（節点情報，有向枝情報，コンストレイント情報）の作成時に，データを更新するような処理に関しては，コンストレイントに対する出力の有向枝情報を作成しないということである．

（5）グラフ解析処理

データ形式の変換で作成されたユニットデータから解析グラフを作成し，強連結成分に分割する．さらに，帰りがけ順の走査を行ない，コンストレイント（ユニット）に順序付けを行なう．帰りがけ順走査における任意性の決定に際して解析前実行順序を用い，強連結成分内の順序付けに関しても解析前実行順序とすることにより，仕様記述者の意図を反映させる．

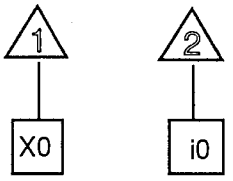
3．実行順序解析の処理

実行順序解析の処理を次の仕様を例として具体的に説明する．



この仕様を解析するための手順は，次のようになる．

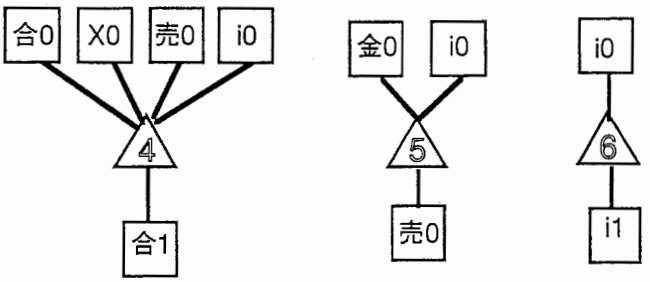
まず，処理ユニット１，２をER図に記述すると



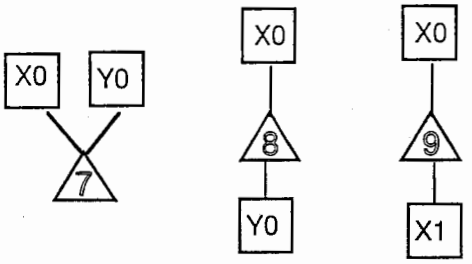
ここで，△はコンストレイントであり処理ユニットに対応し，□はエンティティでデータ項目に対応する．また，仕様書内のX，iをそれぞれX0，i0としたのは，当該データ項目が更新される前後の状態を区別するためである．従って，このER図はあるコンストレイントからそれぞれエンティティX0，i0に有向枝が附加され，処理ユニットからデータが出力されることを示す．

同じく処理ユニット３は

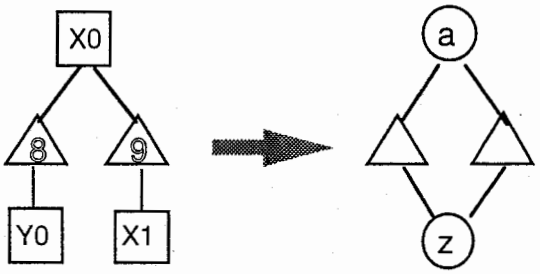
となり，データ項目X0が処理ユニットに入力されることを示す．
次に，処理ユニット４，５，６は，



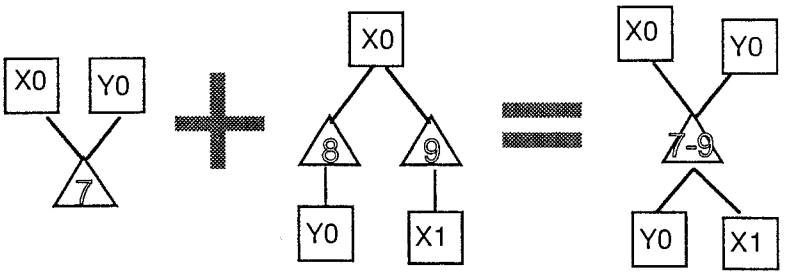
となり，同様に処理ユニット７，８，９は，



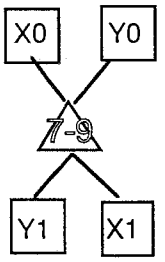
となる．
さて，グラフ解析は最も制御レベルの深いブロックから順に実施する．この例の場合は，まず最も深い条件ブロックに含まれる処理ユニット８，９についてのグラフ解析を行なうために，これらをまとめてER図で示すと次の図（左）のようになる．次に入次数０のエンティティを始点aに，出次数０のエンティティをzに縮退すると，下図の右のようになり，これら２つのコンストレイントの間に順序関係はないことが解る．



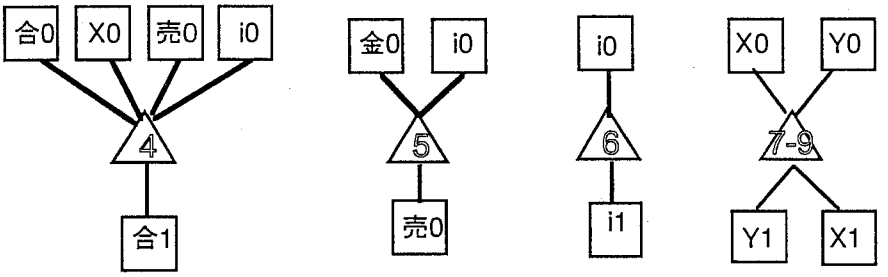
従って，処理ユニット８と９はグラフ解析の結果ではどちらが先に実行されても問題は起こらないが，利用者の意図を反映するため仕様書に記述された順序を優先する．
さて，上位のブロックの解析に移る前に，この条件ブロックを１つの仮想ユニットとして圧縮処理を行なっておく．すなわち，処理ユニット７，８，９を一つの仮想ユニットにまとめる．



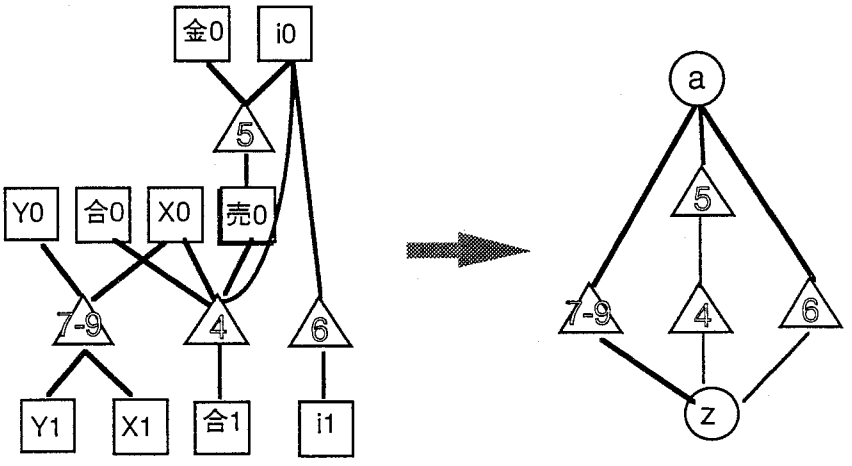
ここで問題となるのは，仮想ユニット（7－9）の入力と出力に同じデータ項目Y0が含まれるため，単一方向閉路となる．従って，これを更新項目とみなし，入出力データ項目をユニーク化する．



次に上位のブロックである繰返しに含まれる処理ユニット4，5，6および仮想ユニット（7－9）についてER図に現すと以下のようになる．

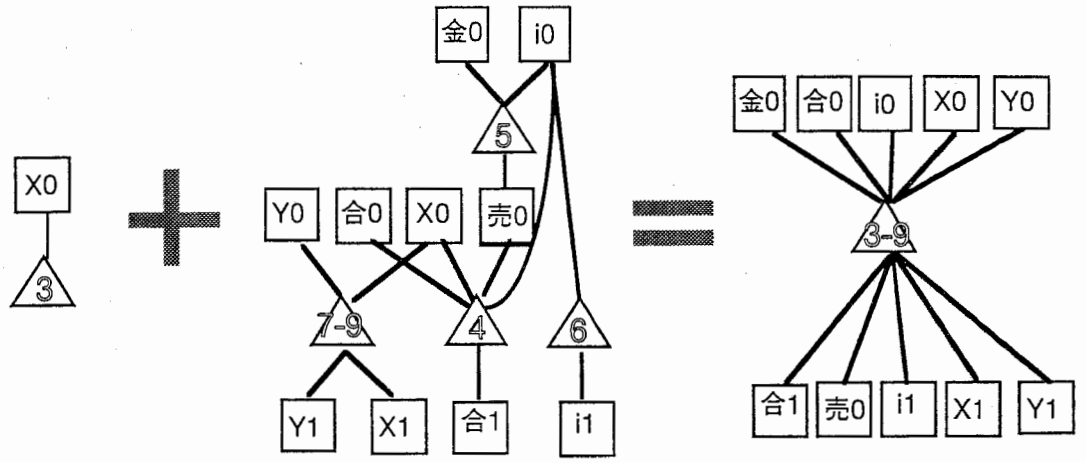


これらを解析のためにグラフ化し，入次数と出次数が0のものを縮退すると以下のようになる．

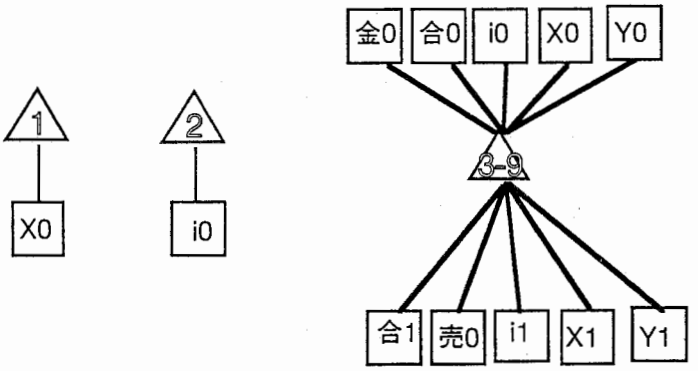


従って，処理ユニット5は4より先に実行する必要があり，並べ替え処理が発生する．

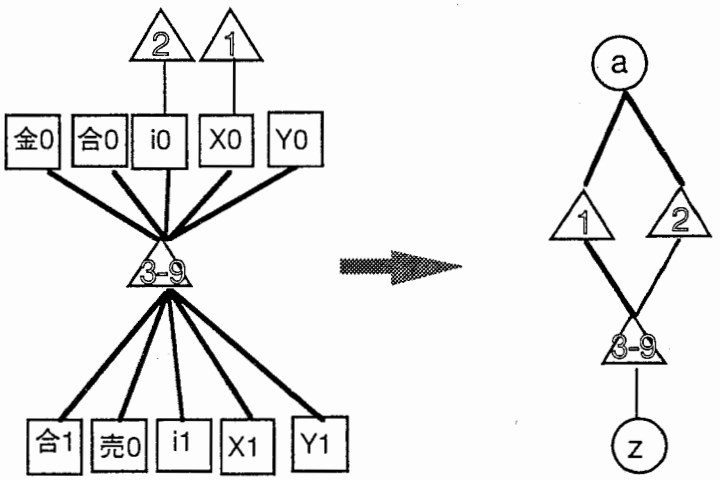
続いて，繰返しブロックそのものを仮想ユニットとして取り扱うために，処理ユニット３と（４－９）を組み合わせる．



この仮想ユニット（３－９）を最上位のブロックに含まれる処理ユニット１，２と解析を行なう．



従って，これらの処理ユニットの解析結果とER図は，以下のようになる．



この結果，処理ユニット１と２は，グラフ解析ではどちらが先に実行されても問題はないが，仕様書に記述された順番を維持することとするので，この両者は並び換わらない．ただし，仮想ユニット（３－８）は処理ユニット１，２の後に実行されなければならない．従って，この仕様を解析すると，処理ユニット４と処理ユニット５が並び換えられる事になる．

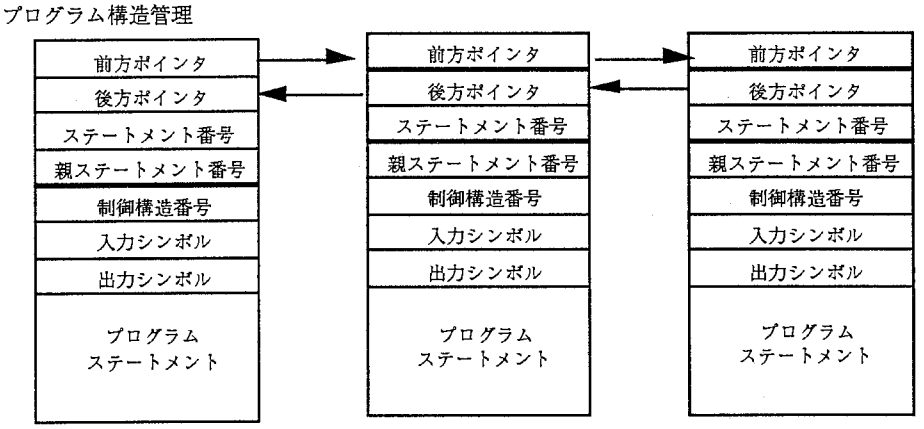
4．実現方式

（1）プログラム構造の管理

仕様に記述された個々の処理ユニットを解析するために，以下のようなデータ形式で処理ユニットを取り扱う．ここで使用する各情報の意味は次のとおりである．

- ポインタ：プログラム構造管理を実現するために使用するメモリの管理用ポインタ
- 制御構造番号：ループ文，条件文などにより形成されるブロックの番号
- ステートメント番号：仕様に記述された順序番号
- 親ステートメント番号：ブロックの基準となるステートメントの番号
- 入力シンボル：入力データ項目あるいは入力エンティティ
- 出力シンボル：出力データ項目あるいは出力エンティティ
- プログラムステートメント：処理ユニットに記述された処理内容

このうち，制御構造番号はそれぞれの処理ユニットが含まれる制御レベルの親子関係を保存し，並び変えを行うブロック範囲の認識を行うのに用いる．

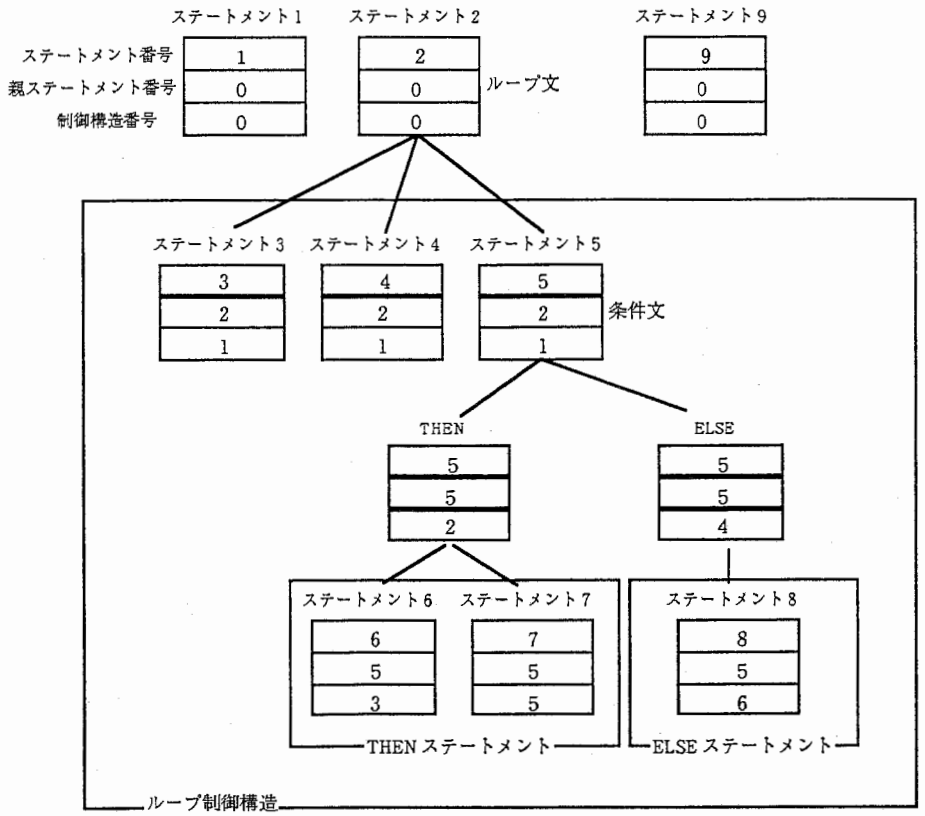


例えば，繰返し文や条件文を構成するブロックは次のように管理する．

下図に示すような階層は，条件文や繰返し文に含まれる処理ユニット群を１つのロジカルシーケンスと見做した場合に生じる，ブロックの呼び出し関係を表したものである．従って，下図の例では処理ユニット２（図ではステートメント２と表示している）の繰返し文には，処理ユニット３，４，５が含まれることを示す．ただし，ここでは，繰り返し終了ユニットは認識せず，ブロック内の処理ユニットについてのみ並び変えを行う．

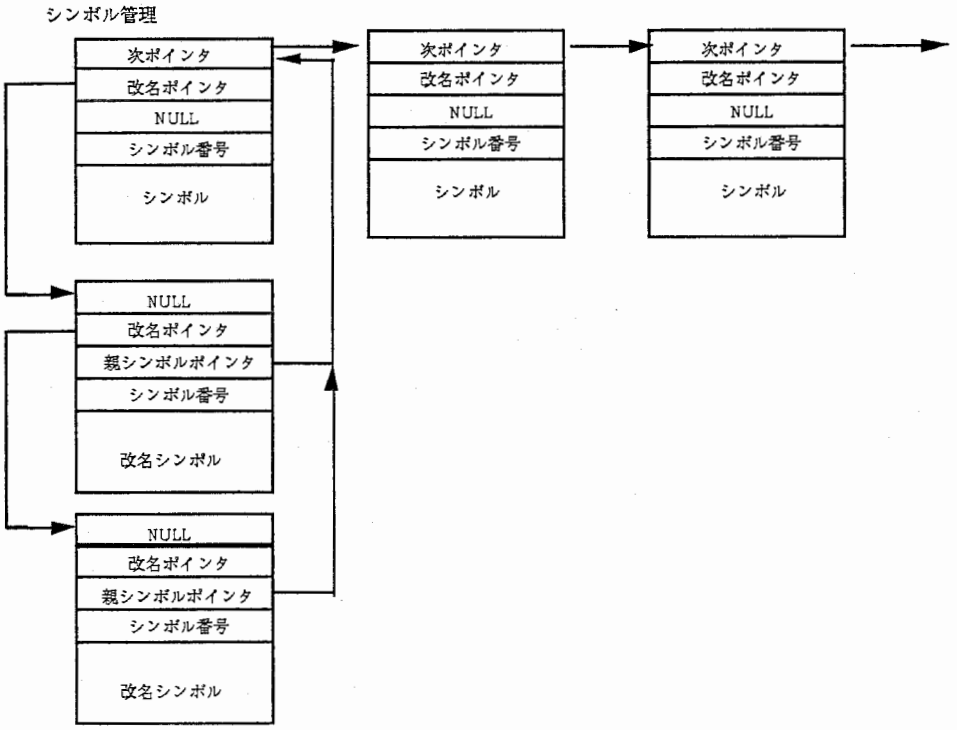
なお，図にも現われているように，繰返しブロック内にはさらに条件ブロックが含まれ，さらに条件ブロックの中にTHENブロック，ELSEブロックが含まれている．したがって，ブロック単位では次の順序にて並び換え処理を実行する．

THENブロック　→　ELSEブロック →　条件ブロック →　繰返しブロック



（2）データ構造の管理

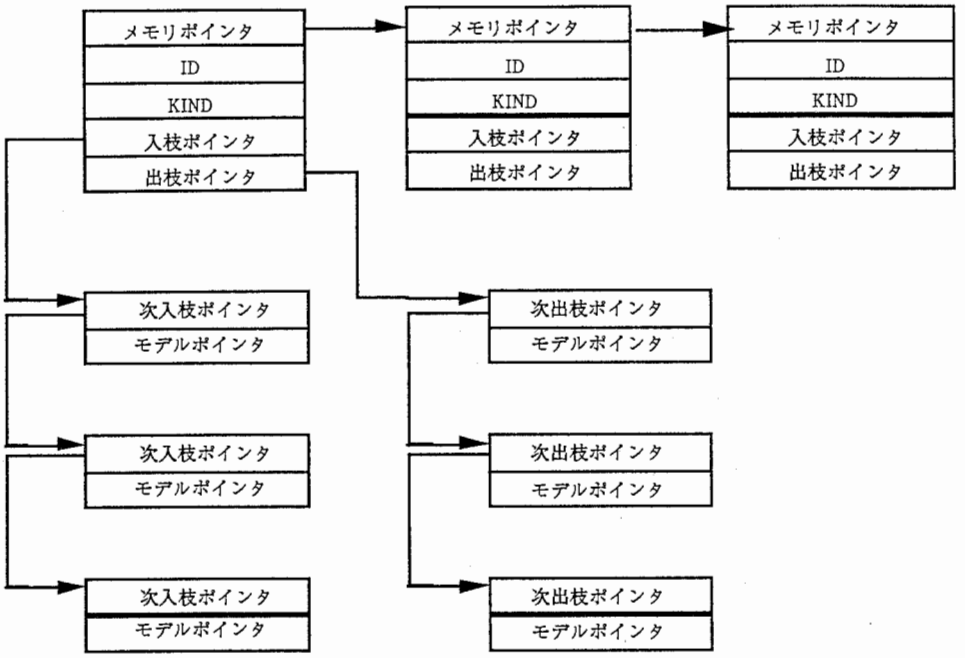
一方，データ構造はそれ自体が構造化されているので，その包含関係などを把握するために次のような方法で管理することとした．ここで，シンボルとはデータ構造を構成するデータ項目をシンボル番号（および改名シンボル番号）にて示したもので，データ構造の親シンボルの参照は構造ポインタにて連携する．



前図は，データ構造の解析時に発見したシンボルを順次シンボルテーブルとして定義していることを示すものである．この時（x=x+1）等のような閉ループが発生しているシンボルがあれば改名シンボルとして定義し，閉ループの参照関係に矛盾がないようにしている．データ構造の親子関係は，構造ポインタを用いて親シンボルを示すことにより，その関係を定義している．

（3）ERモデルの管理方法

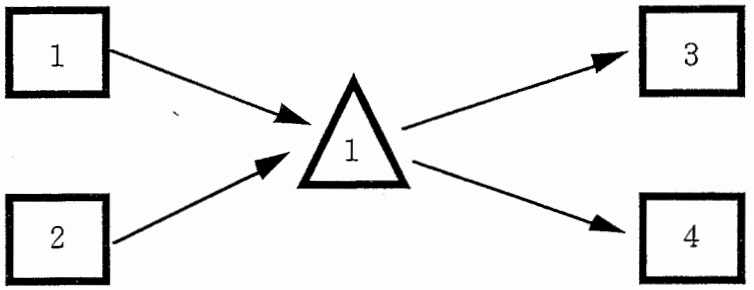
これらをERモデルに変換した後のデータを次のような形式で管理する．



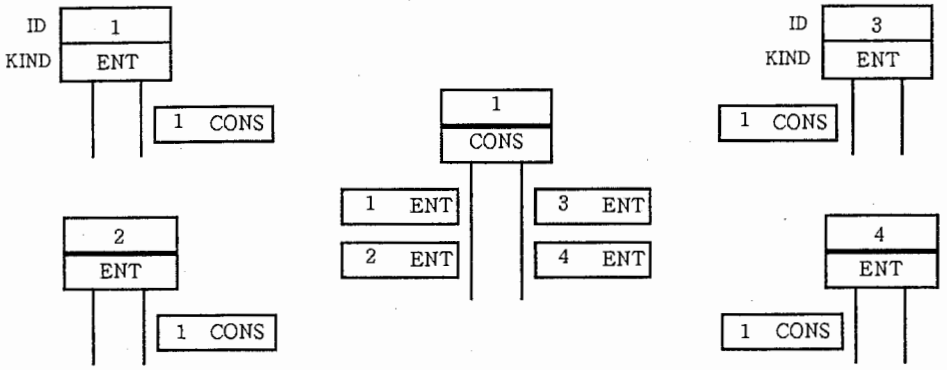
この意味は次の通りである．

- メモリポインタ：ERモデルを管理するために使用するメモリの管理用ポインタ
- KIND：エンティティ／コンストレイントの識別種別
- ID：エンティティもしくはコンストレイントのID番号
  （エンティティならシンボル番号，コンストレイントならステートメント番号が設定される．）
- 入枝／出枝ポインタ：エンティティもしくは，コンストレイントへ入力／出力するモデルへのポインタ

例えば次のような図で表されるERモデルを例にとると，



次のような形で管理される事になる．



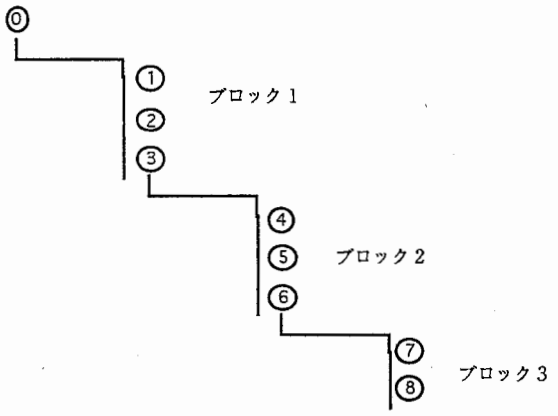
上図は，エンティティ・コンストレイントモデルに対して入枝・出枝を示す枝モデルを持つことにより，ERモデルを表現している事を示している．

（4）制御構造の解析方法

さて，本システムにおける制御構造の解析は次のような方法によっている．

例として次のような仕様書を入力したとすると，前述の解析方法に基づいて繰返し文と条件文ごとに処理ユニットをブロックに分割する．これを制御構造に表すと下側の図のようになる．

```
Ｘに１をセットする ----------①        ブロック１
ｉ に１をセットする ----------②
Ｘ＞１０ まで、以下を繰り返す ----------③
    合計金額（X）に 売上金額（i）を加算する ----------④    ブロック２
    ｉ＝ｉ－１を計算する ----------⑤
    Ｘが Ｙと異なれば ----------⑥
        Ｙに Ｘをセットする ----------⑦    ブロック３
        Ｘ＝Ｙ＋１を計算する ----------⑧
```



（５）データ構造の解析方法

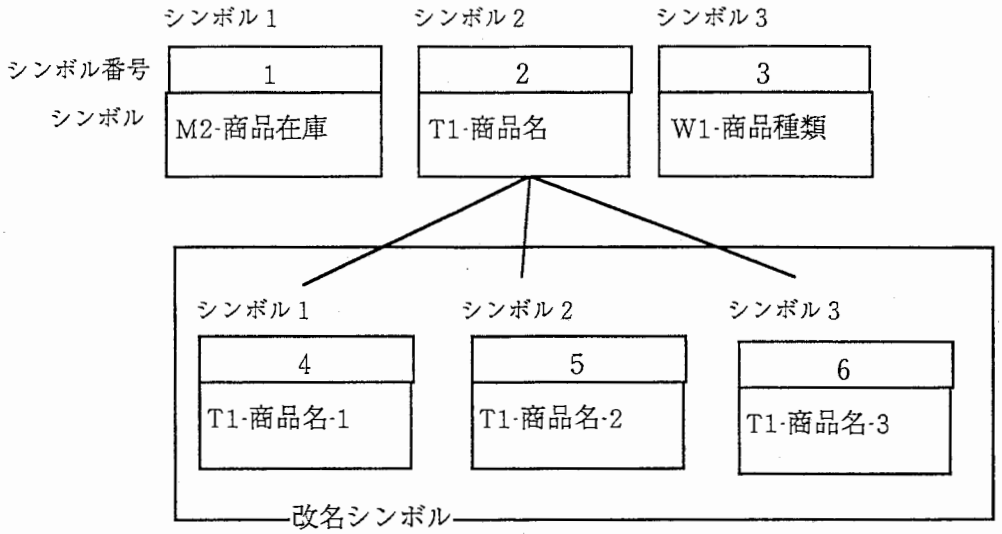
たとえば，エディタで次のようなデータ構造図が定義されていたとすると

T1-商品名			
T1-商品名-1	数字	1	
T1-商品名-2	数字	1	
T1-商品名-3	数字	1	

このようなデータ構造を管理するには次のような方法で行なう．

すなわち，M1-商品在庫・M1-商品在庫（名称）・M1-商品在庫（在庫数）・M1-商品在庫（金額）はそれぞれシンボルリストにより管理され，管理番号としてのシンボル番号が振られる．（項目２を参照のこと）

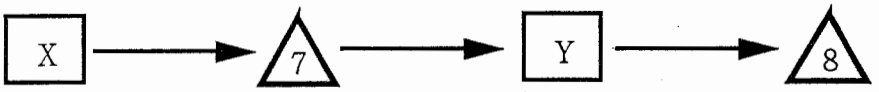
M1-商品在庫は，M1-商品在庫（名称）・M1-商品在庫（在庫数）・M1-商品在庫（金額）より構造ポインタにより親子であることを定義している．



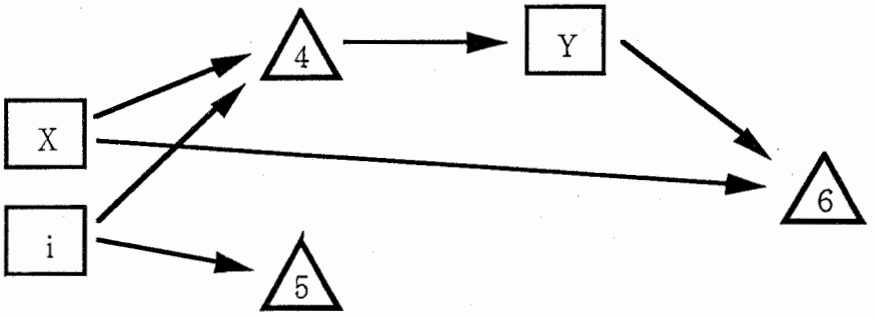
（６）グラフ解析の方法

上記の例をERモデルに展開すると次のようになる.

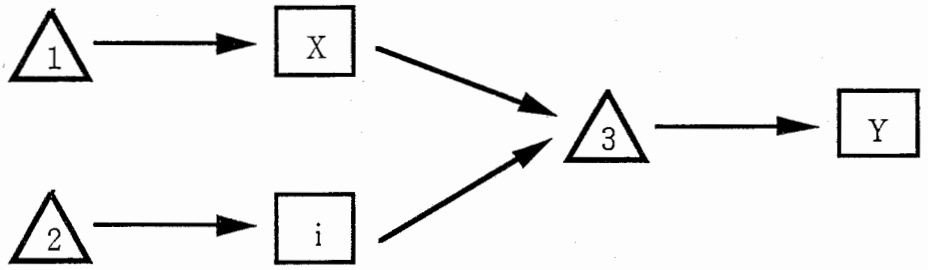
ブロック３



ブロック２

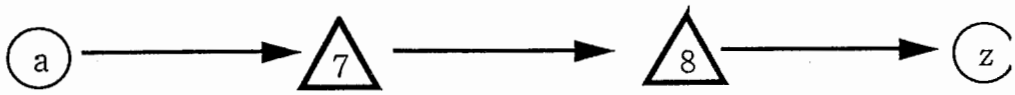


ブロック１

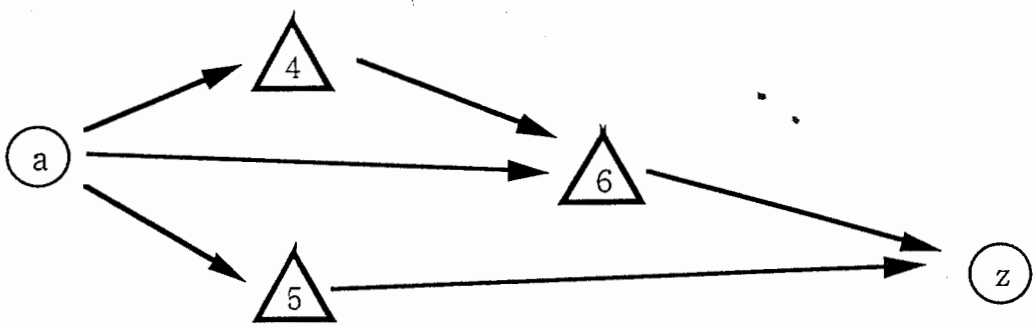


これを，縮退すると以下のようになる．

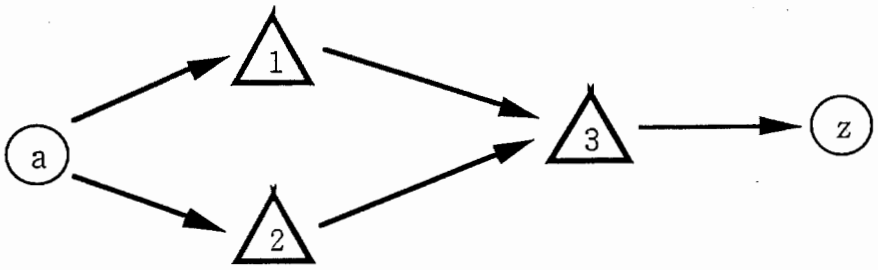
ブロック３



ブロック２



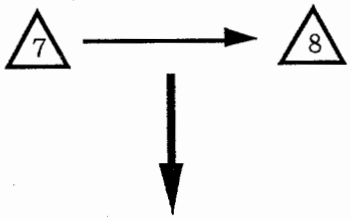
ブロック１



（７）並べ替えの方法

従って，これらのコンストレイントの並び換えは次のように行われる．まず，制御ブロックの最深部のブロックについての並び換えが行われる．

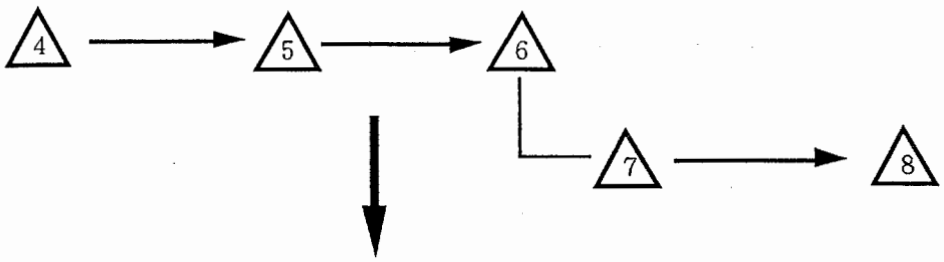
ブロック３



次に，その上位のブロックについての並び換えが行われる．

このときユニット６は，ブロック３の親ユニットであるためブロック３の順序を保持している．

ブロック２



同様に，その上位のブロックについての並び換えが行われる．

親ユニットはそれぞれ下位ブロックの順序を保持したまま自分が属しているブロックで順序並び換えが行われる．

ブロック１