ソフトウェア工学 68－1
（１９８９．９．２６）

対話支援型部品合成システムの試作

岸　美保子　　加地　浩一　　松村　一夫
株式会社　東芝　　システム・ソフトウェア技術研究所

本システムは，設計技法として状態遷移を用いたモデルに基づいて構築された部品合成システムである．部品組み込みに際しては，要求仕様を満足する部品を検索する「候補部品検索フェーズ」，部品の組み合わせの整合性を検証する「整合性チェックフェーズ」，部品間のデータの受渡しを整える「インタフェース調整フェーズ」の３つのフェーズを設けた．また，このシステムは，ユーザとの対話をベースとした協調支援型として構築しており，合成メカニズムでカバーできない部分については，ユーザに問い合わせを行う．

本稿では，この部品合成メカニズムと対話支援方式について述べる．

A program synthesis system by the interactive development

Mihoko Kishi, Koichi Kaji and Kazuo Matsumura
Systems and Software Engineering Laboratory, TOSHIBA Corporation
70 Yanagi-cho Saiwai-ku Kawasaki, 210 Japan

In this paper, we describe a prototype of program synthesis system. This system is based on a state-transition model, which can separate a pogram skeleton and parts.

On this system, a program is constructed through three phases, backtracking repeatedly if necessary. First phase refers candidate parts and second phase checks among parts which are selected in the previous phase. Last phase connects data.

The system also includes a graphical man-machine interface because it must interactively acquire knowledge or decision from its user.

1．はじめに

ソフトウェア生産の工業化を目指すＩＭＡＰシステム(Integrated software MAnagement and Production support system)[1]開発の一環として，ソフトウェアの生産性と品質の向上を目的に，部品合成の研究を進めている．我々は，生産現場のあるアプリケーションに対し，部品合成の観点から分析を行ったところ，状態遷移による設計標準化を前提とする開発方法が有効であることが得られた[2]．この経験を基に，部品合成のモデル化を行い，プロトタイプ・システムを構築した[3]．

本稿では，次節で部品合成の考え方を述べ，３節で状態遷移をベースとした部品合成システムにおけるプログラム生成方式の概要を述べる．そして，４節において部品合成（部品組み込み）のメカニズムを説明し，５節において，それに基づく対話方式の設計思想（方針）について述べる．

2．部品合成の考え方

ソフトウェアの部品化・再利用については，多くの人が望み，期待し，挑戦を続け，成功事例も報告されているが，その反面課題も多い．その課題の一つとしてプログラムレベルの部品を再利用単位として作成するのが困難であることが挙げられる．プログラムというものは，要求機能をある特定の状況に合わせて，具体的なデータと手続きを用いて記述するものである．そして，その特定された状況の影響はプログラムコードの細部にまで分散して取り込まれ，これが１機能部品を他のプログラムコードと完全独立に切り離すことができないものにしてしまう．それ故，暗黙の約束により部品は，他の部品やあるいは部品のつなぎになるコードと多少の関わりをもったまま切り出される．暗黙の約束とは，特定の分野のソフトウェア開発に携わる経験を積んだ技術者が共通に持つ設計・開発手順やソフトウェア・アーキテクチャである．我々は，設計の基本的な枠組み（設計モデル，または設計技法）として状態遷移を取り上げ，部品と部品をつなぐコード，または部品間の切り分けきれない依存関係を無視しないで，部品合成のメカニズムに取り込む方法を考えている．その方法を以下に述べる．

2．1　状態遷移による部品の切り出し（部品と制御構造の関係）

我々は，部品は組み立てを意識して切り出しをされなければならないと考えている．部品の組み立てを容易にする方法としては，プログラムの骨格（制御構造）とこれに埋め込まれる部品群として切り出すという方法がある[4]．実際，事務処理分野（バッチ処理）ではこの方法でかなり成功している．この分野の特徴は，10数種の標準処理パターン（プログラムの骨格に相当するもの）を準備しておけば要求の殆どがカバーできるというところにある．我々は，他の分野においてもこのような考えが適用できないか，つまり標準処理パターンに相当するものが抽出できないかという観点で，ある分野のアプリケーションを分析した．その結果，有効な処理パターンを抽出するには至らなかったが，設計の基本的な枠組みとして状態遷移を用いれば，プログラムの骨格を生成することができ，かつそれが有効であるとの結論が得られた[2]．すなわち，対象アプリケーションを状態遷移により分析し，イベントとアクション部の切り出しを行う．イベントとアクション部については，汎用性のある機能単位として部品化を検討し，状態と遷移の情報は，プログラムの骨格（制御構造）と考えることができる．つまり，図１に示すような骨格と部品群という形で部品を切り出すことができる．

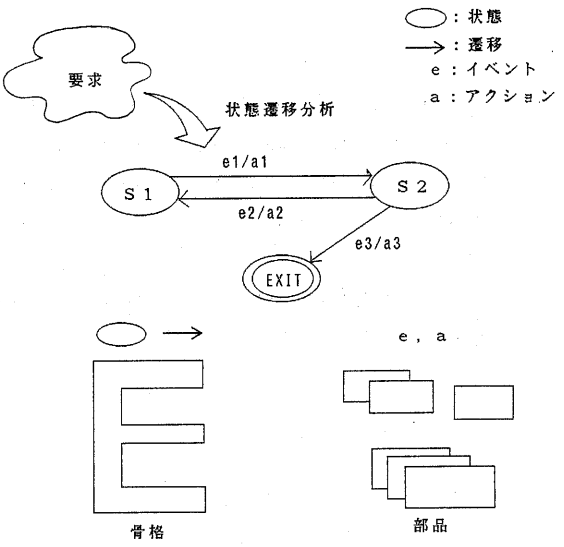


図１　状態遷移による部品の切り出し

2．2　部品のパッケージ化

（部品間の関係）

切り出された部品は，関係の深いものを集めてパッケージ化する．パッケージ内においては凝集性が高く，パッケージ間においては独立性が高くなるようなパッケージ化を行うことにより，可搬性のある保守容易な体系を生み出すことができる．その作成方針として，抽象データ型の考えを導入し，対象（入出力機器，ファイル，テーブル等）に関するデータ群と，それを操作する部品群をひとまとめにすることを考える．さらに，同一パッケージ内の部品依存関係を形式化して記述することにより，部品側から「正しい使われ方」を積極的にアピールできるようにする．

3．プログラム生成方式

状態遷移をベースとした部品合成システムにおけるプログラム生成方式について述べる．図2に示すよう，要求仕様は状態遷移図（状態遷移記述）で記述する．部品ＤＢは，イベント部品ＤＢとアクション部品ＤＢの２種類準備しておく．入力された状態遷移記述の状態（円）と遷移（アロー）の情報からプログラムの骨格を生成し，そこへ状態遷移図上のイベント記述を解析して作成されるイベント判定部品（後述）とアクション記述（［対象名，操作名］の形式で記述）の要求を満たす部品として検索されたアクション部品を組み込み，１本のプログラム（タスク）を生成する．骨格の生成は単純変換で行える[3]ので自動化が可能であるが，部品組み込み部分（特にアクション部品）については，シンプルな記述から機能要求を満足し，かつ組み込まれる状況に適合する部品を検索してくる，かなり難しい作業になる．我々は，２節で述べたように部品をパッケージ化することや部品間の関係を記述することにより，専門家の持つ常識や経験的知識を計算機内に取り込むことを考える．そして，これらを処理するメカニズムを導入し，部品組み込みの自動化を指向する．しかし，知識とその処理メカニズムでカバーしきれないときは，ユーザに新たな情報あるいは複数候補のディシジョンなどを問い合わせなければならない．つまり，本システムはユーザとの対話を行いながらプログラムを生成する方式を採っている．

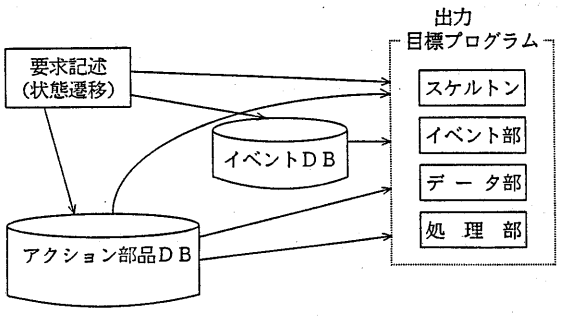


図2　プログラム生成のイメージ図

4．合成メカニズム

本節では，部品の記述方法および組み込みの方法（メカニズム）について詳しく述べる．

4．1　イベント判定部品の作成方法

4．1．1　イベント・パッケージ記述

イベントとは，キーボード，マウス，トラックボール等の入力装置やプロセス送信など外界から入力される信号であると考え，その発信源ごとにパッケージ化を行う．図3にイベント・パッケージ記述の例を示す．２行目から１１行目は仕様部で，１３行目から２２行目が本体部である．１４行目のようなイベントを認識する条件のソースコードの断片をイベント部品と呼ぶ．従って，この例の場合は８個のイベント部品が用意されている．

```
1   ev_package[
2       pkg_spec:
3           (パッケージ名：readchar)
4           (表題：キーボード入力から発生するイベント)
5           (概要：入力バッファから一文字読み込み、イベントを
6                  判定した後読み込んだ文字はバッファに戻す)
7           (発信関数：getch)
8           (受信形式：[型：int,名前：c])
9           (後処理：ungetch)
10          (注意事項：)
11      end_pkg_spec:
12
13      pkg_body:
14          event(英字)( ((c>='a')&&(c<='z'))||((c>='A')&&(c<='Z')) )
15          event(数字)( (c>='0')&&(c<='9') )
16          event(EOF)( c==EOF )
17          event(アスタリスク)( c=='*' )
18          event(一重符)( c=='¥'' )
19          event(二重符)( c=='"' )
20          event(円記号)( c=='¥¥' )
21          event(スラッシュ)( c=='¥/' )
22      end_pkg_body:
23  ]
```

図3　イベント・パッケージ記述の例

4．1．2　イベント判定部品

イベント判定部品とは，状態遷移記述に書かれたイベントを判定するモジュールであり，イベント部品を集めて生成される．図４に，図３のイベント・パッケージを使って生成したイベント判定部品の例を示す．イベントごとに値（数値）を割り当て，それをリターン値として返す．プログラムの骨格は，この値を用いて遷移を制御する．

```
0001  s11()
0002  {
0003      int    e;
0004      char   c;
0005
0006      c = getch();
0007
0008      if( ((c>='a')&&(c<='z'))||((c>='A')&&(c<='Z')) )
0009         e = 0;
0010      else if( c == EOF )
0011         e = 1;
0012      else
0013         e = 2;
0014
0015      ungetch( c );
0016
0017      return( e );
0018  }
```

図４　イベント判定部品の例

4．2　アクション部品の組み込み方法

4．2．1　アクション・パッケージ記述

アクション部品は，2.2で述べた方法でパッケージ化する．図５にアクション・パッケージ記述の例を示す．パッケージ仕様部には，表題や概要の他にパッケージ利用知識を記述する．パッケージ利用知識とは，パッケージ内の部品の使い方を形式的に記述したもので例えば，１３行目のINIT：keytab_initとは，このパッケージに所属するすべての部品を使う以前にkeytab_initという部品が使われていなければならないことを示している．パッケージ本体部には，各アクション部品の記述がされている．各々のアクション部品にも仕様部と本体部がある．１３行目－１５行目がパッケージレベルでの部品利用法を記述しているのに対して，３２行目－３３行目には，部品keytab_initに特定した使い方が記述できる．

```
1    package[
2      pkg_spec:
3        (パッケージ名：keytable)
4        (表題：キーワードをカウントするための表)
5        (概要：)
6        (データ部：
7              #define    STRSIZE    20
8              struct keytable{
9                                  char    word[20];
10                                 int     count;
11                           }keytab;
12             int    n;  /* キーワードの個数 */  )
13       (利用知識：[INIT: keytab_init]
14                  [SELECT: countinc, display ]
15                  [END:] )
16       (注意事項：)
17     pkg_end_spec:
18
19     pkg_body:
20       proc{
21           spec:
22             (部品名：keytab_init)
23             (型：int)
24             (表題：キーテーブルを初期化する)
25             (概要：キーテーブルのキーワードの欄に出現頻度を
26                    カウントするキーワードを書き込む．
27                    出現頻度の欄はすべて０にする．)
28             (検索情報：[OBJ:キーテーブル,OP:初期化する])
29             (参照処理型部品：)
30             (引数：)
31             (リターン値：)
32             (利用知識：[PRE:]
33                        [POST:] )
34             (注意事項：キーワードは，int,char,case,while,for,break)
35           end_spec:
36
37           body:
38             int    i;
39
40             strcpy( keytab.word[0], "int" );
41             strcpy( keytab.word[1], "char" );
                          .
                          .
                          .
```

図５　アクション・パッケージ記述の例

4．2．2　アクション部品の組み込み方

アクション部品の組み込みは，以下に示す３つのフェーズを必要に応じてバックトラックを繰り返し，常に全体の調整をとりながら最終決定する．

（１）候補部品検索フェーズ

このフェーズでは，各アクション記述に対して候補となる部品を洗い出し，ある戦略を用いて優先順位付けを行う．

洗い出し

部品の仕様部に検索情報として対象名と操作名で部品の機能を記述しておく（図５，２８行目参照）．状態遷移記述のアクション記述とこれがマッチした部品を候補部品として全て洗い出す．候補部品が複数存在する場合は，次に示す方法で競合解消を行う．

順位付け

プログラムは，あるデータに対して複数の操作を施すことにより目的を達成する．従って，別々のデータにアクセスする部品で構成されることはない．そこで，パッケージの凝集性を活かし，パッケージへのアクセス回数を競合解消の戦略に用いて，洗い出された部品について順位付けを行う．例えば，図６に示すようアクションａ２に対して部品op13と部品op22が候補として洗い出された場合，部品op13が所属するパッケージには別のアクションa1からもアクセスされているのに対し，部品op22が所属するパッケージには他のアクションからのアクセスがない．よってここでは，部品op13の優先度が高いと判断する．

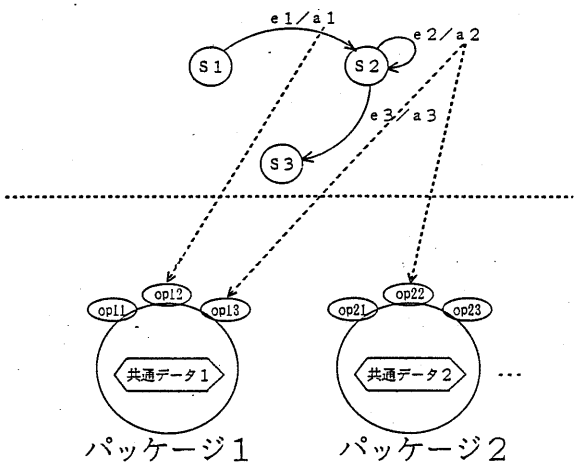


図６　アクセス回数による競合解消戦略

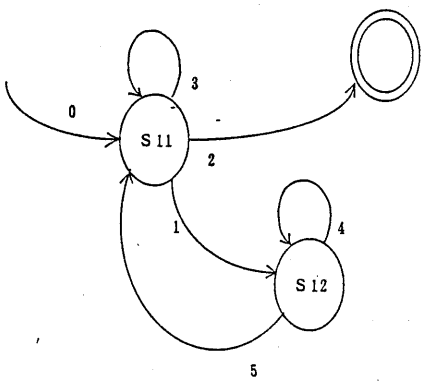
（２）整合性チェックフェーズ

部品使用に対する規約が守られているかチェックしたり，ユーザの記述のもれを指摘するのがこのフェーズである．

チェックには，パッケージや部品の利用知識記述を用いる．利用知識は，部品の使われる順序に関係するものが殆どである．このフェーズでは，部品の使用順序の列を取り出し，利用知識との突合せを行う．

パスの作成

要求として記述した状態遷移がどのように振舞うかその全ての可能性を抽出するのは不可能であるが，なんらかの戦略を設けて，かなり高い精度で振舞いの特徴を吸収するような典型例を抽出することは可能であると考える．そこで，我々は「全アローを最低一回は通過する」よう，状態遷移の振舞い（パス）を抽出することにした．図７にその例を示す．まずアロー０で状態s11に入る．s11から３本のアローが出ているが未通過かつ番号の小さいものを選びながら最終状態まで行く．そうすると０，１，４，５，２でアロー３が残るのでもう一本パスを作成する．この場合，パス２本で全アローを通過することができたことになる．



パス１　　０，１，４，５，２
パス２　　０，３，２

図７　パスの作成

部品列の生成

上記の方法で作成したパスに対し，その各々のアロー上に記述されているアクション記述の第一候補の部品を割り当てて作成するのが部品列である．

部品列と利用知識の突合せ

部品列は，部品利用順序の典型である．これとパッケージ及び各アクション部品の仕様部に記述された利用知識を突合せ，不具合が生じたらワーニングを出してユーザに知らせる．このような方法によるチェックは，十分ではない（パスの作成のされ方により，不具合が見つけだされない場合でも利用知識に反していることがある）が，ワーニングが出た場合は，要求仕様そのものや候補部品検索フェーズでの部品選択が利用知識に反しているということが確実に言える．

（３）インタフェース調整フェーズ

（１）（２）のフェーズで各アクション記述に対する部品が決定すると最後にこのフェーズで部品間のデータの受渡しを調整する．このフェーズを支援する方法として「型」に注目することは有効である．部品の引数にどのデータを渡すかはデータの「型」が同じでなければならないことを用いて絞り込める．しかし，ターゲット言語に準備されている基本型だけでは，効果が薄いため，我々はユーザ定義型の概念を導入したり，データに属性を付けたりして，引数に受け渡すデータの候補に対する制約をより強力なものにした．

５．対話支援方式

３節で述べたように，本システムは，ユーザとの対話をベースとした協調支援型として構築している．そのため，マンマシン・インタフェースの果たす役割は重要で，単にシステムの操作性を向上させるだけでなく，人間の思考活動を支援するものでなければならない．本節では，アクション部品組み込み時の対話支援方式について述べる．

５．１　要求項目と解決策

部品組み込みは3.3.2で述べたように，候補部品検索フェーズ，整合性チェックフェーズ，インタフェース調整フェーズの３つのフェーズを必要に応じてバックトラックを繰り返しながら行われる．つまり，各フェーズでエラーやワーニングが検出されれば，システムがメッセージを出力し，ユーザが修正を行って，処理をやり直すという作業が繰り返されて合成が行われる．従って，この作業が円滑に行われるためには，以下のことが必要である．

(1)なぜエラーやワーニングが出たのかユーザにわかるように表示する

(2)修正に必要な情報が参照しやすく，また参照した内容を入力に反映しやすい環境を構築する

上記２点について，次のような方法で対処する．まず(1)については，ユーザ自身の記述，すなわち要求仕様記述（状態遷移図）そのものを対話画面にすることにより，ユーザ指向の対話を実現する．つまり，エラーやワーニングが発生した位置は，できるだけ状態遷移図上に示す努力をする．また，メッセージ出力のタイミングや内容はユーザの思考にかなったものにする．現状では，ユーザは合成メカニズムの概要（合成が３つのフェーズを経て行われること）を理解していることを前提とし，合成の進捗状況を示すことにより思考を助ける．(2)については，部品ＤＢの情報に常にアクセス可能とし，かつダイレクトマニピュレーションにて部品の差し替えが行えるようにする．

５．２　画面仕様と対話方式

５．２．１　対話画面

対話画面は，以下の４つのウィンドゥで構成される（図８）．

（１）パネルウインドゥ

合成，検索，候補，終了のコマンドを起動する．

（２）エディットウインドゥ

状態遷移図を作成，編集する．

（３）レイアウトウインドゥ

全階層の状態遷移図のレイアウトを表示する．

（４）編集コマンドウインドゥ

状態遷移図編集のコマンドを与える．

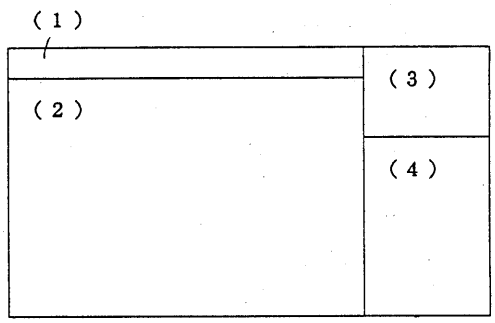


図８　対話画面の構成

５．２．２　合成過程表示バー

部品組み込み時の各フェーズのバックトラックの様子をユーザに知らせるため，合成過程表示バーを設ける．エラーなく通過したフェーズは緑，現在処理中のフェーズは赤くなる．

5．2．3　エラー，ワーニング表示

各フェーズで検出されるエラーやワーニングは，状態遷移図上の原因となった位置にメッセージを出力し直感的理解の促進をはかる．例えば，候補部品検索フェーズにおいて候補部品が全くなかった場合，そのアクション記述の横に「候補部品がありません」というメッセージが書かれたサブウインドゥを表示する．また，整合性チェックフェーズでのワーニングは，図9のように表示される．ワーニングがあれば，画面右下にサブウインドゥが現れ，その一番上のエリアにワーニングメッセージが表示されるが，これはマウス選択により，さらに詳細な情報を得ることができる．つまり，真ん中のエリアに，部品列との突合せで不具合を摘出した利用知識が表示され，エディットウインドゥ上の状態遷移図は問題の部品列の元になったパスを赤くし，各アローには順番を示す番号がふられる．次候補アイコン（一番下のエリア）をクリックすると同一のワーニングに対する別のパスが表示される．これにより，ユーザは番号順にアローをトレースして利用知識との相違を検出できる．さらに，インタフェース調整フェーズにおいては，（この場合エラーではないが）部品名の横にサブウインドゥを開いて引数情報と受け渡しの候補となっているデータの概要一覧を表示する．

5．2．4　候補と検索

アクション記述に対する第一候補部品は，システムが決めるが，ユーザにとっては第二，第三候補も時として貴重な参考情報となる．パネルウインドゥにある候補アイコンをクリックした後，各アクション記述の横に表示されている部品名（第一候補）をクリックすると候補表が表示される（図１０）．候補表には，左から順番に（候補部品検索フェーズで）システムが決めた順位でそのアクションに対する候補部品が表示される．また，これらはマウス操作により，現在組み込まれている部品との差し替えができる．

部品の詳細を確認したいときは，パネルウインドゥにある検索アイコンをクリックした後，画面上にある（サブウインドゥ上でもよい）部品名をクリックすると図１０のように詳細ウインドゥが現れてそこに部品の詳細情報が表示される．詳細ウインドゥには，レベルアイコンがついていてこれにより，詳細度を調節することができる．さらに，対象名，操作名，パッケージ名をキーにして部品を検索することもできる．システムで使用されている用語を知りたいときのために，部品ＤＢ内に登録されている部品の対象名一覧，操作名一覧，パッケージ名一覧も参照できる．

詳細ウインドゥで確認または検索した部品は，単純なマウス操作で現在組み込まれている部品と差し替えることができる．

6．おわりに

状態遷移をベースとした部品合成システムのメカニズムと対話方式について述べた．現在，この考えによるプロトタイプ・システムは，EWS-AS4000上で稼働中である．また，いくつかのトイプログラムに対し適用を行い，評価を行っている．今後は，アプリケーションに対する適用を行い，問題点を抽出し，合成時に用いる戦略や利用知識などの強化をはかる．

［参考文献］

[1]高橋生宗，大筆豊，他
「IMAPシステム(1)-(10)」
情報処理学会第31回全国大会予稿集pp.429-508
[2]加地浩一，松村一夫
「設計標準化に基づく部品合成システムの考察」
情報処理学会ソフトウェア工学研究会予稿集SE-60-4
[3]加地浩一，岸美保子，松村一夫
「状態遷移をベースとした部品合成システムの試作」
情報処理学会第38回全国大会予稿集pp.1199-1200
[4]古宮誠一，原田実
「部品合成による自動プログラミング」
情報処理　VOL.28 1987 NO.10

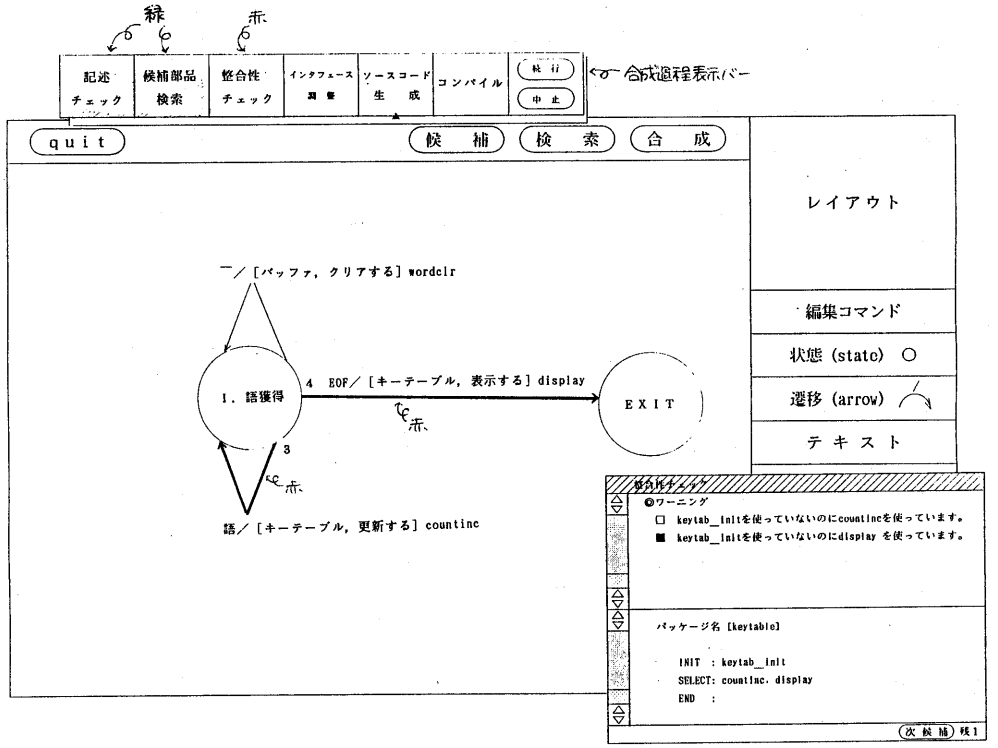


図9　整合性チェックのワーニング

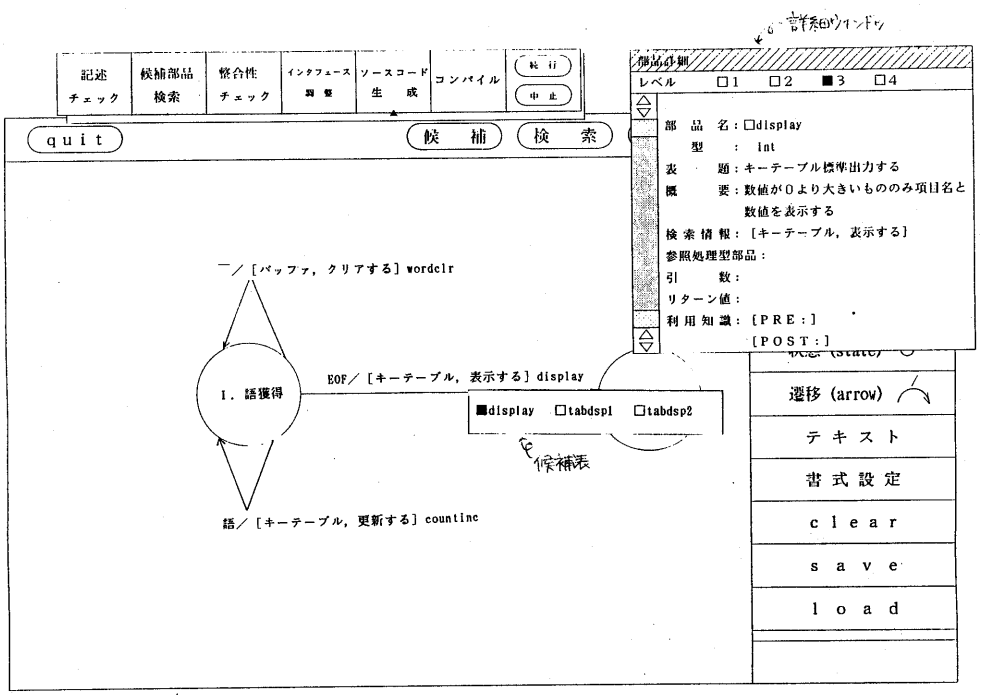


図１０　候補表と詳細ウインドゥ