ICOT Technical Memorandom: TM-0975

TM-0975

KL1上のユニフィケーションプログラムとその評価

越村　三幸（東芝）、藤田　博（三菱）、
長谷川　隆三

November, 1990

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

KL1 上のユニフィケーションプログラムとその評価

越村 三幸* 藤田 博† 長谷川 隆三
(財) 新世代コンピュータ技術開発機構

1 はじめに

KL1 でメタ機能を実現する際[1], オブジェクトレベルの変数の表現方法として,

(a) KL1 変数のまま

(b) 基底項表現

が考えられる. (a) の利点は, 変数管理を言語処理系に任すことにより高速性が期待できること欠点は, unboundness の判定並びに二つの変数の同一性の判定が困難であることである. (b) では上の問題が生じないが、環境 (基底項表現された変数とその値との対の集合) を管理しなければならない.

まず, 我々は (a) のアプローチで KL1 変数の利点を活かしつつ, 問題解決が図れないかどうか検討した. 具体的には, KL1 に unbound という組込み述語が用意されているので, これを利用してユニフィケーションプログラムを書いた. ただし、オブジェクト変数の unboundness はユーザが保証してプログラムしていることを前提とした (この前提は単一 PE 内では保証可能).

ところが, 性能測定の結果, このユニフィケーションプログラムは KL1 の組込み (ボディ側 =) の 5 ~ 21 倍の実行時間を要することがわかった. unbound という組込み述語を使う危険を犯し, 単一 PE 内でしか使えないという制限を課した上, 当初目標の実行時間 10 倍以内のオーバヘッドに抑えられないようでは KL1 変数をそのまま活用するというアプローチは断念せざるをえない.

そこで, (b) のアプローチの検討に移ることにした. (b) では変数管理が必要となる. 2 つの方式の検討を行った.

(b.1) 変数管理表を KL1 のプロセスとして実現する方法. ユニファイする構造体の葉の数くらいの並列度が期待できる.

(b.2) 変数の値を格納するためのメモリ (変数管理表) を切ってアドレス操作をする方法.

(b.3) 変数のタグ部のみを管理表 (1 ビットストリング) で管理する方法.

(b.1) は (b.2),(b.3) に比べストリーム並列性が引き出せ, 並列論理型言語らしいプログラムが書けるが, (b.2),(b.3) に比べ 1 桁低速である. (b.2),(b.3) は変数管理表を持ち回って管理するために逐次的である.

(b.3) は (b.2) に比べて最大 50 % 低速であるが GC の大部分を処理系に任せられる利点がある.

方式 (b) のプログラムでは, KL1 組込みに対しておおよそ 1.5 ~ 4 倍の実行時間内に収まっている.

(a) (b) 共に幾つかのプログラムを書き評価したので以下に述べる.

2 KL1 変数表現

当初この方式では変数管理をまったくする必要がないと期待していたが, 実は変数管理の大部分をしなければならない. ユニフィケーションに失敗した時は未定義変数は未定義変数のままであって欲しいから

*現在, 日本ビジネスオートメーション (株) 所属
†現在, 三菱電機 (株) 中央研究所 所属
[1] ここでは Prolog のインタプリタや一階述語論理のプルーバー を KL1 上で実現することを念頭においている.

である．Prolog のようにバックトラックしてくれれば容易にこの要求にこたえるユニフィケーションプログラムを書くことができるが，単一代入の KL1 ではそうはいかない．

このようなユニフィケーションプログラムを KL1 で書く場合は，オブジェクトレベルのユニフィケーションが失敗しないことを確認してからメタレベルのユニフィケーションをする必要がある．つまり，ユニファイ中にはオブジェクトレベルの変数の値をその変数自身にバインドせずユニファイ終了後にバインドしなければならない．したがってユニファイ中は，変数と変数の値の組の管理つまり変数管理が必須になる．

ユニフィケーションプログラムに変数管理が必要となるので KL1 変数表現の利点はユニフィケーション終了時の変数の値の伝播を処理系に任せられる点だけになってしまう．

変数管理法を代えて 2 通りのプログラムを書いた．

KL1UnifyPO 変数管理を KL1 プロセスにした．具体的には `pool:keyed_set/1` を用いた．キーは変数のプロセッサ番号とプロセッサ内番地の対，データは変数自身とユニファイ中に変数の値をバインドしておく変数の対．

KL1UnifyDO 変数情報を unifier 自身が持ち回る．

3 基底項表現

この方式は変数管理表をプロセス表現するかデータ表現するかの 2 通りの方法がある．

3.1 プロセス表現

この方法が最も KL1 らしいプログラムといえようか．ストリーム並列性がありユニフィケーションのもつ並列性が最も良く反映されるプログラムとなる．

GroundUnifyPO [1] による GHC のプログラムを KL1 用に書き換えた．

3.2 データ表現

オブジェクトレベルの変数の表現とその値の保持の仕方により次の 2 通りの方法を考えた．

ベクタメモリ方式 オブジェクトレベルの変数の値を環境 (ベクタメモリ) に保持する方法．オブジェクトレベルの変数は値の保持されている場所へのポインタで表現される．[2]

ビットメモリ方式 オブジェクトレベルの変数のタグのみを環境 (ビットメモリ) に保持し，オブジェクトレベルの変数の値はメタレベルの変数に保持させる方法．オブジェクトレベルの変数はタグの保持されている場所へのポインタとメタレベルの変数の対によって表現される．[3]

どちらの方式もオブジェクトの表現 (Expression) と環境 (Environment) の対によってオブジェクトを表現する (represent) ことができる．しかし，ベクタメモリ方式では表現を共有し環境を違えることによって異なったオブジェクトを表現できるのに対して，ビットメモリ方式ではそれができない．

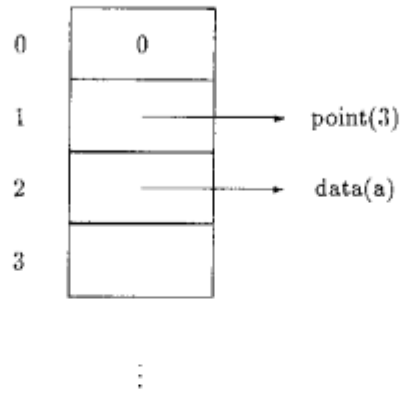
したがって，ベクタメモリ方式を採るかビットメモリ方式を採るかで unify の仕様は違ってくる．

unify(X,Y, Env,NEnv) (ベクタメモリ方式) `X,Y` は `NEnv` のもとで同じオブジェクトを表現する．

unify(X,Y, NX,NY, Env,NEnv) (ビットメモリ方式) `NX,NY` は `NEnv` のもとで同じオブジェクトを表現する．

[2] ユニフィケーションに失敗した場合は単に環境を捨てるだけで良い．

[3] KL1 変数をオブジェクトレベルの変数の値の伝播に使っているので，変数の値を元に戻すための仕掛けが必要になる．



3.2.1 ベクタメモリ方式

KL1 ではベクタというデータ型があるが, これは要素に直接アクセスできるという点がリストに対して著しく有利である. この利点を活かし, ベクタ上に変数セルをとり, 変数はそのベクタインデックスとして表現する. これにより, 変数の unboundness は変数セルに書かれるタグで判定できるし, 2 つの変数の同一性はベクタインデックスの同一性で簡単に判定できる.

変数セルは図のようにベクタ上にとられる. 変数セルには次の 3 種類が入る.

0 未定義

point(Num) 変数番号 `Num` へのポインタ

data(Data) 具体値へのポインタ

この方法で変数の表現法やプログラムの制御を代えて 4 通りのプログラムを書いた.

GroundUnifyVAR 変数の表現を `'$VAR'(N)` (`N` は変数番号 (インデックス)) にする. プログラムの構造は GroundUnify と同じ

GroundUnifyNaive 変数の表現を `{N}` (`N` は変数番号 (インデックス)) にする.

GroundUnifyFlat GroundUnifyNaive の `unify/4` と `unifyVar/4` を一つにまとめた.

GroundUnify GroundUnifyNaive の `unifyVar/4` を `unifyVar/4`, `unifyVarVar/4`, `unifyVarStr/4` に分けた.

GroundUnifyCont GroundUnify で `enqueue_goal` が出ないようにした.

3.2.2 ビットベクタ方式

環境は 1 ビットストリングで表現する. 変数はストリングインデックス (変数番号) と値部の対で表現される. ビットの意味は,

0 (値部は) 未定義

1 (値部は) 変数または具体値へのポインタ

である.

この方法で変数の表現法を代えて 2 通りのプログラムを書いた.

一つはオブジェクトレベルの変数を `{N,Val}` と表現し, もう一つは `{N,Val,FVal}` と表現する. ここで, `N` は変数番号 `Val` は変数値 `FVal` は最終変数値を表す. `Val` と `FVal` には KL1 変数が割り当てられ, 変数の値の参照のために使われる. `Val` の unboundness はビットストリングを参照することで知ることができる. ビットストリングのビット位置 N が 0 の場合 Val は未定義であり, 1 の場合は Val

は他の変数またはデータにユニファイされている．FVal はデレフを KL1 処理系に任すためのもので，{N,Val,FVal} とユニファイされている変数間で共有され変数以外のデータとのユニファイが起こった時にそのデータとユニファイされる．

GroundUnify2Pair 変数の表現は {N,Val} プログラムの構造は GroundUnify と同じ．例えば {1,Val1} と {2,Val2} をユニファイする場合，環境の 1 要素目を 1 に書き換え Val1 と {2,Val2} を (メタレベルの) ユニファイする．

GroundUnify3Pair 変数の表現は {N,Val,FVal}．例えば {1,Val1,FVal1} と {2,Val2,FVal2} をユニファイする場合，環境の 1 要素目を 1 に書き換え Val1 を {2,Val2,FVal2} に具体化し FVal1 と FVal2 を (メタレベルの) ユニファイする．この後で {2,Val2,FVal2} とアトム a をユニファイした場合，環境の 2 要素目を 1 に書き換え，Val2 を a に具体化し FVal2(FVal1) を a に具体化する．

4 展開による最適化

プログラム A.7 GroundUnify においてベクタ同士をユニファイする時に unifyArgs/6 によりベクタ各要素ごとの unify/4 を起動している．これを 6 要素ベクタまでは unifyArgs/6 を起動せずに済むように展開した．得られるプログラムは A.11 GroundUnifyPeval である．

5 評価結果

測定は icpsi167 上の 1 台版 Pseudo マルチ PSI で行った．Version は 1900/0171/0251/011400/0161 である．ヒープメモリは 16'1500 ページに設定した．コンパイルは structured constant モード off で行った．単位はマイクロ秒である．

問題	1	2	3	4	5	6	7	8	9	10
KL1Builtin	154	2300	446	589	686	2.43	160	332	124	288
KL1UnifyPO	11400	78500	32200	33100	36300	3340	24700	24200	22700	22600
KL1UnifyDO	771	21500	4210	4480	5160	104	3090	3150	2640	2670
GroundUnifyPO	3020	70600	25400	25500	12400	379	6500	6790	8110	8250
GroundUnifyVar	320	4900	894	1120	1590	39.4	591	823	474	686
GroundUnifyNaive	330	4190	881	1110	1560	34.4	604	835	534	670
GroundUnifyFlat	326	4550	928	1140	1640	45.9	732	906	623	788
GroundUnify	306	4180	878	1080	1510	34.2	620	844	487	697
GroundUnifyCont	261	3510	831	915	1170	37.4	603	873	516	737
GroundUnify2Pair	345	4500	992	1260	1830	50.4	682	929	568	876
GroundUnify3Pair	398	4970	1160	1380	1600	66.2	689	915	671	867
GroundUnifyPeval	173	1830	551	593	781	36.2	596	815	511	687

次に KL1 の組込みを 1 したときの速度比を載せる．

問題	1	2	3	4	5	6	7	8	9	10
KL1Builtin	1	1	1	1	1	1	1	1	1	1
KL1UnifyPO	74.0	34.1	72.2	56.2	52.9	1370	154	72.9	183	78.5
KL1UnifyDO	5.01	9.35	9.44	7.60	7.52	42.8	19.3	9.49	21.3	9.27
GroundUnifyPO	19.6	30.7	57.0	43.3	18.1	156	40.6	20.5	65.4	28.6
GroundUnifyVAR	2.08	2.13	2.00	1.90	2.31	16.2	3.69	2.48	3.82	2.38
GroundUnifyNaive	2.14	1.82	1.98	1.88	2.27	14.2	3.78	2.52	4.31	2.33
GroundUnifyFlat	2.12	1.97	2.07	1.94	2.39	18.9	4.58	2.73	5.02	2.74
GroundUnify	1.98	1.82	1.97	1.83	2.20	14.1	3.88	2.54	3.93	2.42
GroundUnifyCont	1.69	1.52	1.86	1.55	1.71	15.4	3.77	2.63	4.16	2.56
GroundUnify2Pair	2.24	1.96	2.22	2.14	2.68	20.7	4.26	2.80	4.58	3.04
GroundUnify3Pair	2.58	2.16	1.17	2.34	2.33	27.2	4.31	2.76	5.41	3.01
GroundUnifyPeval	1.12	0.80	1.23	1.01	1.14	14.9	3.72	2.45	4.12	2.39

問題は以下の通り[4].

1 $X = \mathtt{i(i(X1,Y1),Z1)}$, $Y = \mathtt{i(i(i(B,C),D),i(i(D,B),i(E,B)))}$

2 32 個の異なった変数を持つ binary tree

3 $X = \mathtt{p(h(X1,X1),h(X2,X2),Y2,Y3,X3)}$, $Y = \mathtt{p(X2,X3,h(Y1,Y1),h(Y2,Y2),Y3)}$

4 $A = \mathtt{i(i(X,X),i(i(Y,Y),i(V,i(W,Z))))}$, $B = \mathtt{i(Y,i(Z,i(i(U,U),i(i(V,V),W))))}$

5 $A = \mathtt{(X0,X1,X2,X3,X4,X5,X6,f(X0),f(X6))}$, $B = \mathtt{(X1,X2,X3,X4,X5,X6,a,f(X6),f(X0))}$

6 $A = \mathtt{X}$, $B = \mathtt{f(a)}$

7 $A = \mathtt{[X,x,Y,y,Z,z,U,u,V,v,W,w]}$, $B = \mathtt{[x,X,y,Y,z,Z,u,U,v,V,w,W]}$

8 $A = \mathtt{f(X,x,Y,y,Z,z,U,u,V,v,W,w)}$, $B = \mathtt{f(x,X,y,Y,z,Z,u,U,v,V,w,W)}$

9 $A = \mathtt{[X0,X1,X2,X3,X4,X5,X6,X7,X8,X9]}$, $B = \mathtt{[0,1,2,3,4,5,6,7,8,9]}$

10 $A = \mathtt{f(X0,X1,X2,X3,X4,X5,X6,X7,X8,X9)}$, $B = \mathtt{f(0,1,2,3,4,5,6,7,8,9)}$

6 考察

本小論で述べたユニフィケーションプログラムのアルゴリズムはほぼ同じである. 違いはメタレベルの変数の表現の仕方と変数管理の方法である. 変数表現としては,

KL1 変数のまま オブジェクトレベルの変数をメタレベルの変数で表現する.

基底項表現 オブジェクトレベルの変数をメタレベルの基底項で表現する.

の 2 つを考え, 変数管理法としては

テーブル方式 変数の名前と値の対を表にして一括管理する方法.

プロセス方式 各変数ごとに変数管理プロセスを生成する方法.

の 2 つを考えた. まとめると次のようになる.

	KL1 変数	基底項
テーブル方式	KL1Unify*	GroundUnify*
プロセス方式	-	GrondUnifyPO

[4]詳しくは A.12 測定用プログラム を参照のこと.

6.1 変数表現 - KL1 変数表現と基底項表現 -

オブジェクトレベルの変数を KL1 の変数で表現する方法の最大の利点は変数管理を KL1 言語処理系にまかることが期待できることである．しかし実は値の伝播を除いた部分の変数管理をする必要がある．ユニフィケーション中にオブジェクトレベルの変数に値を代入していくような Prolog 風のプログラムは実用的ではない．というのはユニフィケーションに失敗した場合，オブジェクトの変数の値を元に戻さないといけないからである．したがって，ユニフィケーションプログラムは成功することが分かってから実際にオブジェクトレベルの変数と値をユニファイしなければならない．つまり，変数管理が必要になるわけである[5]

このような理由により KL1 変数表現の利点は全くないと言ってよい．事実基底項表現に比べ 3 ～ 4 倍[6]低速になっている．これは構造体の皮むきをする度に unboundness の検査をしなければならないのと，ユニフィケーションに成功してから本当の変数に値を束縛する必要があるためだと考えられる．

KL1 変数表現を用いる場合，問題自身をユニフィケーションを回避するように形式化しなおすのが懸命であると思われる [2]．

6.2 変数値の伝播

オブジェクトレベルの変数値の伝播を KL1 言語処理系に任せようと試みたプログラムが GroundUnify2Pair と GroundUnify3Pair である．これらのプログラムは変数の KL1 変数表現と基底項表現の折衷表現というべきものである．

基底項表現に比べて，

1. 変数環境がビットストリングであるため消費メモリの節約が期待できる．
2. デレフ用変数 Val, FVal により変数チェインのたぐりの高速化が期待できる．

の 2 点において有利であると考えられる．

1 変数当たりの消費メモリは

GroundUnify2Pair 2 ワード + 1 ビット．

GroundUnify3Pair 3 ワード + 1 ビット．

GroundUnify 未定義の場合，2 ワード．それ以外の場合，4 ワード

である．折衷方式の方が優れているようだが同じ変数の出現が複数ある場合，2 回目以降の 1 出現について GroundUnify2Pair では 2 ワード GroundUnify3Pair では 3 ワード必要なのに対して，GroundUnify は 1 ワードですむので，変数の出現回数が多ければ基底項表現が有利になる．

GroundUnify2Pair,GroundUnify3Pair のスピードは GroundUnify に較べ 1.1 ～ 1.5 倍低速である．折衷方式では変数に対するタグが 0 と 1 と 2 種類なのに対し，基底項方式では 0 と point と data の 3 種類である．したがって，`unifyVar`,`unifyVarVar`,`unifyVarStr` の処理の最適化が少しできるためであると考えられる．

6.3 変数管理 - プロセス方式とテーブル方式 -

プロセス方式による変数管理は並列処理言語らしいプログラミングができユニフィケーション問題の持つ並列性を自然に引き出すことができる．しかしプロセス方式はテーブル方式に比べ 1 桁ほど[7]低速である．標準的なユニフィケーションの並列度が 2 ～ 3 である [1] ことを考えるとプロセッサの台数効果を考慮に入れてもテーブル方式の方が勝っているといえるだろう．

なぜプロセス方式はテーブル方式に比べこんなにも低速なのであろうか? プロセス方式では変数の個数分の変数管理プロセスができる．変数管理が必要になるとこの変数管理プロセスが起動される．一方テーブル方式では変数表をユニファイヤ自身が保持するためこのようなプロセスはできない．変数管理

[5] 一方 Prolog ではバックトラック機構によりプログラムはそのようなことに注意を払う必要はない．こういう点も考えると KL1 は Prolog に比べるといかにメタプログラミングしにくい言語であることか．

[6] KL1UnifyDO と GroundUnify の比較

[7] GroundUnifyPO と GroundUnify の比較

プロセス自体の処理は軽いが, このプロセスは起動されるとすぐに休眠状態となる. このように, プロセス方式では軽いプロセスの切替えが頻繁に起こる.

現在の KL1 言語処理方式 [6] は頻繁にプロセス切替えが起こるようなプログラムに対しては有利な方式とはいえない. しかしプロセス切替えが頻繁に起こることを仮定した処理系 [3] ならばプロセス方式も有利かもしれない.

6.4 完全逐次制御

今回作成したプログラムで最高速なものは GroundUnifyCont である. これは完全逐次プログラムである. つまり, サスペンドしない限りどのゴールも enqueue されない. リストの場合の処理について GroundUnify と比べることによりその差異を考えてみる.

GroundUnify .

```
unify([XH|XT],[YH|YT], T,NT) :- true |
    unify(XH,YH, T,T1), unify(XT,YT, T1,NT).
```

GroundUnifyCont .

```
unify([XH|XT],[YH|YT], Cont, Env,NEnv) :- true |
    unify(XH,YH, [{XT,YT}|Cont], Env,NEnv).
```

現在の KL1 処理方式ではボディ部にユーザゴールが複数ある場合は最初の一つを除き ready queue に入れられる. そして最初の一つは引き続き実行される[8].

GroundUnify の場合リストの Tail の処理 `unify(XT,YT, T1,NT)` は ready queue に enqueue される. 一方 GroundUnifyCont では Tail の処理は自前のスタックに積み, Head 部分の処理を行う (`unify(XH,YH,[{XT,YT}|Cont], Env,NEnv)`). どのゴールも enqueue されない代わりに `unifyCont/3` が呼ばれる. ユニファイする構造体の葉の数を L とすると `unifyCont/3` は L 回呼ばれる. 一方 GroundUnify では L-1 回 ゴールが enqueue される.

GroundUnifyCont は GroundUnify に比べ大体 10 ~ 20 % 高速である. これは完全逐次にした結果, データが常にレジスタ上にあるからである[9]. リダクション数は GroundUnify に比べ L 回増えるのにも拘わらず高速なのはそれにも増して enqueue が重いということである[10].

7 終わりに

いろいろなユニフィケーションプログラムについてその性能を単体で比較した. 最も速いプログラムは完全逐次制御プログラム (GroundUnifyCont) であった. このプログラムは台数効果は全く期待できない. しかし, 台数効果が最も期待できる変数プロセス管理方式は 1 桁低速であることをふまえると完全逐次制御プログラムの方に軍配を挙げざるをえない.

つまり, ユニフィケーションは現 KL1 処理系では粒度が小さ過ぎる問題ということになる. ユニフィケーションは探索問題では肝となる処理ではあるが負荷分散という観点からは, もう少し粒度の大きい仕事を分散させるのが有効であることが分かった.

我々は KL1 でメタプログラミングをする際に必要になるであろう機能をライブラリとして準備中である [4]. このライブラリは本稿での考察をふまえ十分高速なものを目指している.

References

[1] H. Fujita, *Parallel Unification and Meta-Interpreters in GHC*, ICOT TR-468 1989

[8] goto で jump するものと思ってよい

[9] PIM ではこうはならない [5]. したがって,PIM では GroundUnifyCont は遅くなるであろう.

[10] enqueue_goal + proceed は execute より重いということ.

[2] R. Hasegawa, H. Fujita, M. Fujita, *A Parallel Theorem Prover in KL1 and Its Application to Program Synthesis*, 日瑞伊ワークショップ, 1990

[3] K. Ueda, M. Morita, *A New Implementation Technique for Flat GHC*, 7th ICLP, 1990

[4] 越村三幸, 藤田 博, 長谷川隆三, メタライブラリ・マニュアル, ICOT TM 準備中.

[5] 平野喜芳, 後藤厚宏, 並列論理型言語 KL1 のコンパイル方式の改良, JSPP'90, 1990

[6] 宮崎敏彦, 瀧 和男, Multi-PSI における FlatGHC の実現方式, LPC'86 ICOT, 1986

A プログラムリスト

A.1 KL1UnifyPO

```
:- module unify. :- public unify/3,unify/4,unify/5.

unify(_,_,_,_,_) :- true | true .

unify(X,Y,Res) :- true |
    pool:keyed_set(S),
    unify(X,Y,S0,Res1),
    S=[do(S0),get_all(All)],
    unify_result(Res1,All,Res).

unify_result(success,All,Res) :- true |
    Res=success, unify_all(All).
unify_result(fail,_,Res) :- true | Res=fail.

unify_all([{_,{X,NewX}}|Vlist]) :- true |
    X=NewX, unify_all(Vlist).
unify_all([]) :- true | true.

unify(X,Y,S,Res) :- true |
    unbound(X,VarX), unbound(Y,VarY),
    unify1(VarX,VarY,S,Res).

unify1({X},{Y},S,Res) :- vector(X,Size), vector(Y,Size),
    vector_element(X,0,F), vector_element(Y,0,F), atom(F) |
    unify_vector(1,Size,X,Y,S,Res).
unify1({[XH|XT]},{[YH|YT]},S,Res) :- true |
    S = {S1,S2},
    unify(XH,YH, S1, ResH), unify(XT,YT, S2, ResT),
    both(ResH,ResT, Res).
unify1({PE,Adr,X},{Y},S,Res) :- true |
    S=[put({PE,Adr},{X,NewX},OldVal)|S1],
    check_value(OldVal,NewX,Y,S1,Res).
unify1({Y},{PE,Adr,X},S,Res) :- true |
    S=[put({PE,Adr},{X,NewX},OldVal)|S1],
    check_value(OldVal,NewX,Y,S1,Res).
unify1({PEX,AdrX,X},{PEX,AdrX,Y},S,Res) :- true | Res=success, S = [].
otherwise.
unify1({PEX,AdrX,X},{PEY,AdrY,Y},S,Res) :- true |
    S=[put({PEX,AdrX},{X,NewX},OldValX),
       put({PEY,AdrY},{Y,NewY},OldValY)|S1],
    check_value2(OldValX,NewX,Y,OldValY,NewY,X,S1,Res).
unify1({X},{X},S,Res) :- true | Res=success, S=[].
```

```
otherwise.
unify1(_,_,S,Res) :- true | S=[], Res=fail.

check_value({},NewX,Y,S,Res) :- true | NewX=Y, S=[], Res=success.
check_value({{_,OldX}},NewX,Y,S,Res) :- true | OldX=NewX, unify(NewX,Y,S,Res).

check_value2({},NewX,_,{},NewY,_,S,Res) :- true |
    NewX=NewY, S=[], Res=success.
check_value2({{_,OldX}},NewX,Y,{},NewY,_,S,Res) :- true |
    OldX=NewX, NewY = NewX, S = [], Res = success.
check_value2({},NewX,_,{{_,OldY}},NewY,X,S,Res) :- true |
    OldY=NewY, NewX = NewY, S = [], Res = success.
check_value2({{_,OldX}},NewX,_,{{_,OldY}},NewY,_,S,Res) :- true |
    OldX=NewX, OldY=NewY, unify(NewX,NewY,S,Res).

unify_vector(J,L,X,Y,S,Res) :-
    vector_element(X,J,XJ), vector_element(Y,J,YJ), J1 := J+1 |
    S={S1,S2},
    unify_vector(J1,L,X,Y,S1,ResR),
    unify(XJ,YJ,S2,ResJ),
    both(ResJ,ResR,Res).
unify_vector(L,L,_,_,S,Res) :- true | S=[], Res=success.

both(success,success,Ans) :- true | Ans=success.
both(fail,_,Ans) :- true | Ans=fail.
both(_,fail,Ans) :- true | Ans=fail.
```

A.2 KL1UnifyDO

```
:- module unify. :- public unify/3,unify/4,unify/5 .

unify(_,_,_,_) :- true | true.

unify(X,Y,Res) :- true |
    unify(X,Y,[],St,Res1), unify_result(Res1,St,Res).

unify_result(success,All,Res) :- true |
    Res=success, unify_all(All).
unify_result(fail,_,Res) :- true | Res=fail.

unify_all([{_,{X,NewX}}|Vlist]) :- true |
    X=NewX,
    unify_all(Vlist).
unify_all([]) :- true | true.

unify(X,Y,S,St,Res) :- true |
    unbound(X,VarX), unbound(Y,VarY),
    unify1(VarX,VarY,S,St,Res).

unify1({X},{Y},S,St,Res) :- vector(X,Size), vector(Y,Size),
    vector_element(X,0,F), vector_element(Y,0,F) |
    unify_vector(1,Size, X,Y,S,St,Res).
unify1({[XH|XT]},{[YH|YT]},S,St,Res) :- true |
    unify(XH,YH, S,S1, ResH), unify(XT,YT, S1,St, ResT),
    both(ResH,ResT, Res).
unify1({PE,Adr,X},{Y},S,St,Res) :- true |
```

```
    put({PE,Adr},{X,Y},{X,OldX}, S,S1, YN),
    checkValue(YN, OldX,Y, S1,St, Res).
unify1({Y},{PE,Adr,X},S,St,Res) :- true |
    put({PE,Adr},{X,Y},{X,OldX}, S,S1, YN),
    checkValue(YN, OldX,Y, S1,St, Res).
unify1({PEX,AdrX,X},{PEX,AdrX,Y},S,St,Res) :- true | S = St, Res = success.
otherwise.
unify1({PEX,AdrX,X},{PEY,AdrY,Y},S,St,Res) :- true |
    put({PEX,AdrX},{X,NewX},{X,OldX}, S,S1, Y_N_X),
    (Y_N_X = yes    -> Res = success,
        put({PEY,AdrY},{Y,NewX},{Y,NewX}, S1,St, _);
     Y_N_X = no ->
        put({PEY,AdrY},{Y,OldX},{Y,OldY}, S1,S2, YN),
        checkValue(YN, OldX,OldY,S2,St,Res)).
unify1({X},{X},S,St,Res) :- true | Res=success, S=St.
otherwise.
unify1(_,_,S,St,Res) :- true | S=St, Res=fail.

checkValue(yes, _,_, S,St, Res) :- true | S = St, Res = success.
checkValue(no, X,Y, S,St, Res) :- true | unify(X,Y, S,St, Res).

unify_vector(J,L,X,Y,S,St,Res) :-
    vector_element(X,J,XJ), vector_element(Y,J,YJ), J1 := J+1 |
    unify_vector(J1,L,X,Y,S,S1,ResR),
    unify(XJ,YJ,S1,St,ResJ),
    both(ResJ,ResR,Res).
unify_vector(L,L,_,_,S,St,Res) :- true | S=St, Res=success.


both(success,success,Ans) :- true | Ans=success.
both(fail,_,Ans) :- true | Ans=fail.
both(_,fail,Ans) :- true | Ans=fail.

put(Key,InData,OutData, [{Key,OData}|KDs],NKD, YN) :- true |
    OutData = OData, NKD = [{Key,OData}|KDs], YN = no.
put(Key,InData,OutData, [],NKD, YN) :- true |
    NKD = [{Key,InData}], YN = yes.
otherwise.
put(Key,InData,OutData, [{Key1,Data1}|KDs],NKD, YN) :- true |
    NKD = [{Key1,Data1}|NKD1],
    put(Key,InData,OutData, KDs,NKD1, YN).
```

A.3 GroundUnifyPO

```
:- module unify.
:- public unify/3,unify/4,unify/5.

unify(X,Y, _,Rst) :- true | unify(X,Y, Rst).

unify(X,Y,Res) :- true |
    unify(X,Y,S1,sw(S,success),Abort),
    merge({S1,OutS2}, InS),
    sift(InS,OutS,OutS2,Abort),
    uResult(S,Abort,OutS,Res).

uResult(_,abort(all),_,Res) :- true | Res=fail.
```

```
uResult(success,Abort,OutS,Res) :- true | Res=OutS, Abort=abort(filters).

unify(_,_, OutS,_,abort(_)) :- true | OutS = [].
alternatively.
unify(X,Y, OutS,SW,Abort) :- true | unify1(X,Y, OutS,SW,Abort).

unify1(V,T, OutS,SW,_) :- V='$VAR'(_) | unifyVars(V,T, OutS,SW).
unify1(T,V, OutS,SW,_) :- V='$VAR'(_) | unifyVars(V,T, OutS,SW).
unify1([XCar|XCdr],[YCar|YCdr], OutS,sw(A,Z),Abort) :- true |
    OutS = {OutS1,OutS2},
    unify(XCar,YCar, OutS1,sw(A,M),Abort),
    unify(XCdr,YCdr, OutS2,sw(M,Z),Abort).
unify1(X,Y, OutS,SW,Abort) :- vector(X,Size), vector(Y,Size),
    vector_element(X,0,F), vector_element(Y,0,F), F \= '$VAR' |
    unifyArgs(1,Size, X,Y, OutS,SW,Abort).
otherwise.
unify1(X,X, OutS,sw(A,Z),_) :- true | OutS = [], A = Z.
otherwise.
unify1(X,Y, OutS,_,Abort) :- true | OutS = [], Abort = abort(all).

unifyArgs(N,N, _,_, OutS,sw(A,Z),_) :- true | OutS = [], A = Z.
unifyArgs(M,N, X,Y, OutS,sw(A,Z),Abort) :-
    vector_element(X,M,Xm), vector_element(Y,M,Ym) |
    OutS = {OutS1,OutS2},
    unify(Xm,Ym, OutS1,sw(A,B),Abort),
    unifyArgs(~(M+1),N, X,Y, OutS2,sw(B,Z),Abort).

unifyVars(V,V, OutS,sw(A,Z)) :- true | OutS = [], A = Z.
otherwise.
unifyVars(V,T, OutS,SW) :- true | OutS = [subst(V,T,T,FinT,SW)].

sift(_,OutS,OutS2,abort(all)) :- true | OutS = [], OutS2 = [].
sift(_,OutS,OutS2,abort(filters)) :- true | OutS=[], OutS2 = [].
sift([subst(V,IniV,CurV,FinV,sw(A,Z))|InS],OutS,OutS2,Abort) :- true |
    A=Z,
    sift_final(V,FinV,OutS,OutS1),
    OutS2 = {OutS21,OutS22},
    filter(InS,V,IniV,CurV,FinV,InS1,OutS21,Abort),
    sift(InS1,OutS1,OutS22,Abort).

sift_final(V,V,OutS,OutS1) :- true | OutS=OutS1.
otherwise.
sift_final(V,T,OutS,OutS1) :- true | OutS=[V=T|OutS1].

filter(InS,_,_,CurV,FinV,InS1,OutS2,abort(_)) :- true |
    InS1 = InS, OutS2 = [], FinV=CurV.
alternatively.
filter(InS,V,IniV,CurV,FinV,OutS,OutS2,Abort) :- wait(InS) |
    filter1(InS,V,IniV,CurV,FinV,OutS,OutS2,Abort).

filter1([subst(V,IniV1,_,FinV1,SW)|InS],
       V,IniV,CurV,FinV,OutS,OutS2,Abort) :- true |
    FinV1=FinV,
    OutS2 = {OutS21,OutS22},
    unify(IniV1,IniV,OutS21,SW,Abort),
    filter(InS,V,IniV,CurV,FinV,OutS,OutS22,Abort).
```

```
filter1([subst(U,V,_,FinU,SW)|InS],
        V,IniV,CurV,FinV,OutS,OutS2,Abort) :- true |
    FinU=FinV,
    OutS=[subst(U,IniV,CurV,FinV,SW)|OutS1],
    filter(InS,V,IniV,CurV,FinV,OutS1,OutS2,Abort).
otherwise.
filter1([subst(U,IniU,CurU,FinU,SW)|InS],
        V,IniV,CurV,FinV,OutS,OutS2,Abort) :- true |
    deref(CurU,V=FinV,CurU1,Abort),
    deref(CurV,U=FinU,CurV1,Abort),
    OutS=[subst(U,IniU,CurU1,FinU,SW)|OutS1],
    filter(InS,V,IniV,CurV1,FinV,OutS1,OutS2,Abort).

deref(_,_,_,abort(all)) :- true | true.
alternatively.
deref(X,Subst,NX,Abort) :- wait(X) | deref1(X,Subst,NX,Abort).

deref1('$VAR'(I),'$VAR'(I)=S,NX,_) :- true | NX=S.
otherwise.
deref1('$VAR'(I),'$VAR'(J)=S,NX,_) :- true | NX = '$VAR'(I).
deref1(X,Subst,NX,Abort) :- vector(X,Size), vector_element(X,0,F), F \= '$VAR' |
   derefArgs(0,Size, X,Subst,NX,Abort).
deref1(X,_,NX,_) :- atom(X) | NX = X.
deref1(X,_,NX,_) :- integer(X) | NX = X.

derefArgs(N,N, X,_,NX,_) :- true | NX = X.
derefArgs(M,N, X,Subst,NX,Abort) :- M < N |
    set_vector_element(X,M, Xm,NXm, X1),
    deref(Xm,Subst,NXm, Abort),
    derefArgs(M+1),N, X1,Subst,NX,Abort).
```

A.4 GroundUnifyVAR

```
:- module unify. :- public unify/3, unify/4, unify/5.

unify(_,_,_) :- true | true. unify(_,_,_,_,_) :- true | true.
%
%   unify(+X,+Y, +Table,-Table)
%
unify(X,Y, fail,NT) :- true | NT = fail.
otherwise.
unify([XH|XT],[YH|YT], T,NT) :- true | unify(XH,YH, T,T1), unify(XT,YT, T1,NT).
unify('$VAR'(N),Y, T,NT) :- integer(N) | unifyVar(N, Y, T,NT).
unify(X,'$VAR'(M), T,NT) :- integer(M) | unifyVar(M, X, T,NT).
otherwise.
unify(X,Y, T,NT) :- vector(X, Size), vector(Y, Size),
    vector_element(X,0,F), vector_element(Y,0,F), atom(F) |
    unifyArgs(1,Size, X,Y, T,NT).
otherwise.
unify(X,X, T,NT) :- true | NT = T.
otherwise.
unify(X,Y, _,NT) :- true | NT = fail.

unifyArgs(M,N, X,Y, T,NT) :-
    vector_element(X, M, Xm), vector_element(Y, M, Ym), M1 := M + 1 |
    unify(Xm,Ym, T,T1),
```

```
    unifyArgs(M1,N, X,Y, T1,NT).
unifyArgs(N,N, _,_, T,NT) :- true | NT = T.

unifyVar(XNum, '$VAR'(YNum), T,NT) :- integer(YNum) | unifyVarVar(XNum,YNum, T,NT).
otherwise.
unifyVar(XNum,Y, T,NT) :- true | unifyVarStr(XNum,Y, T,NT).

unifyVarVar(XNum,XNum, T,NT) :- true | NT = T.
otherwise.
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,XNum,point(XRef)) |
    unifyVarVar(XRef,YNum, T,NT).
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,XNum,data(XVal)) |
    unifyVarStr(YNum,XVal, T,NT).
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,YNum,point(YRef)) |
    unifyVarVar(XNum,YRef, T,NT).
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,YNum,data(YVal)) |
    unifyVarStr(XNum,YVal, T,NT).
unifyVarVar(XNum,YNum, T,NT) :-
    vector_element(T,XNum,0), vector_element(T,YNum,0) |
    set_vector_element(T,XNum, _,point(YNum), NT).

unifyVarStr(XNum,Y, T,NT) :- vector_element(T,XNum,point(XRef)) |
    unifyVarStr(XRef,Y, T,NT).
unifyVarStr(XNum,Y, T,NT) :- vector_element(T,XNum,data(XVal)) |
    unify(XVal,Y, T,NT).
unifyVarStr(XNum,Y, T,NT) :- vector_element(T,XNum,0) |
    set_vector_element(T,XNum, _,data(Y),NT).
```

A.5 GroundUnifyNaive

```
:- module unify. :- public unify/3,unify/4,unify/5.

unify(_,_,_) :- true | true. unify(_,_,_,_,_) :- true | true.
%
%   unify(+X,+Y, +Table,-Table)
%
unify(X,Y, fail,NT) :- true | NT = fail.
otherwise.
unify({N},Y, T,NT) :- integer(N) | unifyVar(N, Y, T,NT).
unify(X,{M}, T,NT) :- integer(M) | unifyVar(M, X, T,NT).
unify([XH|XT],[YH|YT], T,NT) :- true |
    unify(XH,YH, T,T1), unify(XT,YT, T1,NT).
unify(X,Y, T,NT) :- vector(X, Size), vector(Y, Size),
    vector_element(X,0,F), vector_element(Y,0,F), atom(F)|
    unifyArgs(1,Size, X,Y, T,NT).
otherwise.
unify(X,X, T,NT) :- true | NT = T.
otherwise.
unify(X,Y, _,NT) :- true | NT = fail.

unifyArgs(M,N, X,Y, T,NT) :-
    vector_element(X, M, Xm), vector_element(Y, M, Ym), M1 := M + 1 |
    unify(Xm,Ym, T,T1),
    unifyArgs(M1,N, X,Y, T1,NT).
unifyArgs(N,N, _,_, T,NT) :- true | NT = T.
```

```
unifyVar(XNum, {YNum}, T,NT) :- XNum =:= YNum | NT = T.
otherwise.
unifyVar(XNum, Y, T,NT) :- vector_element(T,XNum,point(XRef)) |
    unifyVar(XRef, Y, T,NT).
unifyVar(XNum, Y, T,NT) :- vector_element(T,XNum,data(XVal)) |
    unify(XVal, Y, T,NT).
unifyVar(XNum, {YNum}, T,NT) :- vector_element(T,YNum,point(YRef)) |
    unifyVar(XNum, {YRef}, T,NT).
unifyVar(XNum, {YNum}, T,NT) :- vector_element(T,YNum,data(YVal)) |
    unifyVar(XNum, YVal, T,NT).
unifyVar(XNum, {YNum}, T,NT) :-
    vector_element(T,XNum,0), vector_element(T,YNum,0) |
    set_vector_element(T,XNum, _,point(YNum),NT).
otherwise.          % vector_element(T,XNum,0)
unifyVar(XNum, Y, T,NT) :- true | set_vector_element(T,XNum, _,data(Y), NT).
```

A.6 GroundUnifyFlat

```
:- module unify. :- public unify/3,unify/4,unify/5.

unify(_,_,_) :- true | true . unify(_,_,_,_,_) :- true | true .
%
%   unify(+X,+Y, +Table,-Table)
%
unify(X,Y, fail,NT) :- true | NT = fail.
unify({XNum},{YNum}, T,NT) :- XNum =:= YNum | NT = T.
otherwise.
unify([XH|XT],[YH|YT], T,NT) :- true |
    unify(XH,YH, T,T1), unify(XT,YT, T1,NT).
unify(X,Y, T,NT) :- vector(X, Size), vector(Y, Size),
    vector_element(X,0,F), vector_element(Y,0,F), atom(F)|
    unifyArgs(1,Size, X,Y, T,NT).
unify({XNum},Y, T,NT) :- vector_element(T,XNum,point(XRef)) |
    unify({XRef},Y, T,NT).
unify({XNum},Y, T,NT) :- vector_element(T,XNum,data(XVal)) |
    unify(XVal,Y, T,NT).
unify(X,{YNum}, T,NT) :- vector_element(T,YNum,point(YRef)) |
    unify(X,{YRef}, T,NT).
unify(X,{YNum}, T,NT) :- vector_element(T,YNum,data(YVal)) |
    unify(X,YVal, T,NT).
unify({XNum},{YNum}, T,NT) :-
    vector_element(T,XNum,0), vector_element(T,YNum,0) |
    set_vector_element(T,XNum, _,point(YNum), NT).
otherwise.
unify({XNum},Y, T,NT) :- vector_element(T,XNum,0) |
    set_vector_element(T,XNum, _,data(Y), NT).
unify(X,{YNum}, T,NT) :- vector_element(T,YNum,0) |
    set_vector_element(T,YNum, _,data(X), NT).
otherwise.
unify(X,X, T,NT) :- true | NT = T.
otherwise.
unify(X,Y, _,NT) :- true | NT = fail.

unifyArgs(M,N, X,Y, T,NT) :-
    vector_element(X, M, Xm), vector_element(Y, M, Ym), M1 := M + 1 |
    unify(Xm,Ym, T,T1),
```

```
    unifyArgs(M1,N, X,Y, T1,NT).
unifyArgs(N,N, _,_, T,NT) :- true | NT = T.
```

A.7 GroundUnify

```
:- module unify. :- public unify/3,unify/4,unify/5.

unify(_,_,_) :- true | true . unify(_,_,_,_,_) :- true | true .

unify(X,Y, fail,NT) :- true | NT = fail.
otherwise.
unify({N},Y, T,NT) :- integer(N) | unifyVar(N, Y, T,NT).
unify(X,{M}, T,NT) :- integer(M) | unifyVar(M, X, T,NT).
unify([XH|XT],[YH|YT], T,NT) :- true |
    unify(XH,YH, T,T1), unify(XT,YT, T1,NT).
unify(X,Y, T,NT) :- vector(X, Size), vector(Y, Size),
    vector_element(X,0,F), vector_element(Y,0,F), atom(F)|
    unifyArgs(1,Size, X,Y, T,NT).
otherwise.
unify(X,X, T,NT) :- true | NT = T.
otherwise.
unify(X,Y, _,NT) :- true | NT = fail.

unifyArgs(M,N, X,Y, T,NT) :-
    vector_element(X, M, Xm), vector_element(Y, M, Ym), M1 := M + 1 |
    unify(Xm,Ym, T,T1),
    unifyArgs(M1,N, X,Y, T1,NT).
unifyArgs(N,N, _,_, T,NT) :- true | NT = T.

unifyVar(XNum, {YNum}, T,NT) :- integer(YNum) | unifyVarVar(XNum,YNum, T,NT).
otherwise.
unifyVar(XNum, Y, T,NT) :- true | unifyVarStr(XNum,Y, T,NT).

unifyVarVar(XNum,XNum, T,NT) :- true | NT = T.
otherwise.
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,XNum,point(XRef)) |
    unifyVarVar(XRef,YNum, T,NT).
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,XNum,data(XVal)) |
    unifyVarStr(YNum,XVal, T,NT).
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,YNum,point(YRef)) |
    unifyVarVar(XNum,YRef, T,NT).
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,YNum,data(YVal)) |
    unifyVarStr(XNum,YVal, T,NT).
unifyVarVar(XNum,YNum, T,NT) :-
    vector_element(T,XNum,0), vector_element(T,YNum,0) |
    set_vector_element(T,XNum, _,point(YNum), NT).

unifyVarStr(XNum,Y, T,NT) :- vector_element(T,XNum,point(XRef)) |
    unifyVarStr(XRef,Y, T,NT).
unifyVarStr(XNum,Y, T,NT) :- vector_element(T,XNum,data(XVal)) |
    unify(XVal,Y, T,NT).
unifyVarStr(XNum,Y, T,NT) :- vector_element(T,XNum,0) |
    set_vector_element(T,XNum, _,data(Y),NT).
```

A.8 GroundUnifyCont

```
:- module unify. :- public unify/3, unify/4, unify/5.

unify(_,_,_) :- true | true .
%
%   unify(+X,+Y, +Env,-Env)
%
unify(X,Y, Env,NEnv) :- true | unify(X,Y, [], Env,NEnv).

unify({N},Y, Cont, Env,NEnv) :- true | unifyVar(N,Y, Cont, Env,NEnv).
unify(X,{M}, Cont, Env,NEnv) :- true | unifyVar(M,X, Cont, Env,NEnv).
unify([XH|XT],[YH|YT], Cont, Env,NEnv) :- true |
    unify(XH,YH, [{XT,YT}|Cont], Env,NEnv).
unify({F,X},{F,Y}, Cont, Env,NEnv) :- true |
    unify(X,Y, Cont, Env,NEnv).
unify(X,Y, Cont, Env,NEnv) :- vector(X, Size), vector(Y, Size), Size > 2,
    vector_element(X,0,F), vector_element(Y,0,F), atom(F),
    vector_element(X,1,X1), vector_element(Y,1,Y1) |
    unify(X1,Y1, [{2,Size,X,Y}|Cont], Env,NEnv).
otherwise.
unify(X,X, Cont, Env,NEnv) :- true | unifyCont(Cont, Env,NEnv).
otherwise.
unify(X,Y, _, _,NEnv) :- true | NEnv = fail.

unifyCont([], Env,NEnv) :- true | NEnv = Env.
unifyCont([{X,Y}|Cont], Env,NEnv) :- true | unify(X,Y, Cont, Env,NEnv).
unifyCont([{N,M,NX,NY}|Cont], Env,NEnv) :- N1 := N+1, N1 =:= M,
    vector_element(NX,N, NXn), vector_element(NY,N, NYn) |
    unify(NXn,NYn, Cont, Env,NEnv).
unifyCont([{N,M,NX,NY}|Cont], Env,NEnv) :- N1 := N+1, N1 < M,
    vector_element(NX,N, NXn), vector_element(NY,N, NYn) |
    unify(NXn,NYn, [{N1,M, NX,NY}|Cont], Env,NEnv).

unifyVar(XNum, {YNum}, Cont, Env,NEnv) :- integer(YNum) |
    unifyVarVar(XNum,YNum, Cont, Env,NEnv).
otherwise.
unifyVar(XNum, Y, Cont, Env,NEnv) :- true |
    unifyVarStr(XNum,Y, Cont, Env,NEnv).

unifyVarVar(XNum,XNum, Cont, Env,NEnv) :- true | unifyCont(Cont, Env,NEnv).
otherwise.
unifyVarVar(XNum,YNum, Cont, Env,NEnv) :- vector_element(Env,XNum,point(XRef)) |
    unifyVarVar(XRef,YNum, Cont, Env,NEnv).
unifyVarVar(XNum,YNum, Cont, Env,NEnv) :- vector_element(Env,XNum,data(XVal)) |
    unifyVarStr(YNum,XVal, Cont, Env,NEnv).
unifyVarVar(XNum,YNum, Cont, Env,NEnv) :- vector_element(Env,YNum,point(YRef)) |
    unifyVarVar(XNum,YRef, Cont, Env,NEnv).
unifyVarVar(XNum,YNum, Cont, Env,NEnv) :- vector_element(Env,YNum,data(YVal)) |
    unifyVarStr(XNum,YVal, Cont, Env,NEnv).
unifyVarVar(XNum,YNum, Cont,Env,NEnv) :-
    vector_element(Env,XNum,0), vector_element(Env,YNum,0) |
    set_vector_element(Env,XNum, _,point(YNum), Env1),
    unifyCont(Cont, Env1,NEnv).

unifyVarStr(XNum,Y, Cont, Env,NEnv) :- vector_element(Env,XNum,point(XRef)) |
    unifyVarStr(XRef,Y, Cont, Env,NEnv).
```

```
unifyVarStr(XNum,Y, Cont, Env,NEnv) :- vector_element(Env,XNum,data(XVal)) |
    unify(XVal,Y, Cont, Env,NEnv).
unifyVarStr(XNum,Y, Cont, Env,NEnv) :- vector_element(Env,XNum,0) |
    set_vector_element(Env,XNum, _,data(Y),Env1),
    unifyCont(Cont, Env1,NEnv).
```

A.9 GroundUnify2Pair

```
:- module unify. :- public unify/3,unify/4,unify/5.

unify(_,_,_) :- true | true. unify(_,_,_,_,_) :- true | true.
%
%   unify(+X,+Y, +Table,-Table)
%
unify(X,Y, fail,NT) :- true | NT = fail.
otherwise.
unify({N,X},Y, T,NT) :- integer(N) | unifyVar(N,X, Y, T,NT).
unify(X,{M,Y}, T,NT) :- integer(M) | unifyVar(M,Y, X, T,NT).
unify([XH|XT],[YH|YT], T,NT) :- true |
    unify(XH,YH, T,T1), unify(XT,YT, T1,NT).
unify(X,Y, T,NT) :- vector(X, Size), vector(Y, Size),
    vector_element(X,0,F), vector_element(Y,0,F), atom(F)|
    unifyArgs(1,Size, X,Y, T,NT).
otherwise.
unify(X,X, T,NT) :- true | NT = T.
otherwise.
unify(X,Y, _,NT) :- true | NT = fail.

unifyArgs(M,N, X,Y, T,NT) :-
    vector_element(X, M, Xm), vector_element(Y, M, Ym), M1 := M + 1 |
    unify(Xm,Ym, T,T1),
    unifyArgs(M1,N, X,Y, T1,NT).
unifyArgs(N,N, _,_, T,NT) :- true | NT = T.

unifyVar(XNum,XVal, {YNum,YVal}, T,NT) :- integer(YNum) |
    unifyVarVar(XNum,XVal, YNum,YVal, T,NT).
otherwise.
unifyVar(XNum,XVal, Y, T,NT) :- true | unifyVarStr(XNum,XVal, Y, T,NT).

unifyVarVar(XNum,_, XNum,_, T,NT) :- true | NT = T.
otherwise.
unifyVarVar(XNum,XVal, YNum,YVal, T,NT) :- string_element(T,XNum,2#"1") |
    unifyVar(YNum,YVal, XVal, T,NT).
unifyVarVar(XNum,XVal, YNum,YVal, T,NT) :- string_element(T,YNum,2#"1") |
    unifyVar(XNum,XVal, YVal, T,NT).
unifyVarVar(XNum,XVal, YNum,YVal, T,NT) :-
    string_element(T,XNum,2#"0"), string_element(T,YNum,2#"0") |
    XVal = {YNum,YVal},
    set_string_element(T, XNum,2#"1", NT).

unifyVarStr(XNum,XVal, Y, T,NT) :- string_element(T,XNum, 2#"0") |
    XVal = Y, set_string_element(T, XNum,2#"1", NT).
unifyVarStr(XNum,XVal, Y, T,NT) :- string_element(T,XNum, 2#"1") |
    unify(XVal,Y, T,NT).
```

A.10 GroundUnify3Pair

```
:- module unify. :- public unify/3,unify/4,unify/5.

unify(_,_,_) :- true | true . unify(_,_,_,_,_) :- true | true.
%
%   unify(+X,+Y, +Table,-Table)
%
unify(X,Y, fail,NT) :- true | NT = fail.
otherwise.
unify({N,X,FX},Y, T,NT) :- integer(N) | unifyVar(N,X,FX, Y, T,NT).
unify(X,{M,Y,FY}, T,NT) :- integer(M) | unifyVar(M,Y,FY, X, T,NT).
unify([XH|XT],[YH|YT], T,NT) :- true |
    unify(XH,YH, T,T1), unify(XT,YT, T1,NT).
unify(X,Y, T,NT) :- vector(X, Size), vector(Y, Size),
    vector_element(X,0,F), vector_element(Y,0,F), atom(F)|
    unifyArgs(1,Size, X,Y, T,NT).
otherwise.
unify(X,X, T,NT) :- true | NT = T.
otherwise.
unify(X,Y, _,NT) :- true | NT = fail.

unifyArgs(M,N, X,Y, T,NT) :-
    vector_element(X, M, Xm), vector_element(Y, M, Ym), M1 := M + 1 |
    unify(Xm,Ym, T,T1),
    unifyArgs(M1,N, X,Y, T1,NT).
unifyArgs(N,N, _,_, T,NT) :- true | NT = T.

unifyVar(XNum,XVal,FXVal, {YNum,YVal,FYVal}, T,NT) :- integer(YNum) |
    unifyVarVar(XNum,XVal,FXVal, YNum,YVal,FYVal, T,NT).
otherwise.
unifyVar(XNum,XVal,FXVal, Y, T,NT) :- true |
    unifyVarStr(XNum,XVal,FXVal, Y, T,NT).

unifyVarVar(XNum,_,_, XNum,_,_, T,NT) :- true | NT = T.
otherwise.
unifyVarVar(_,_,FXVal, YNum,YVal,FYVal, T,NT) :- wait(FXVal) |
    unifyVar(YNum,YVal,FYVal, FXVal, T,NT).
unifyVarVar(XNum,XVal,FXVal, _,_,FYVal, T,NT) :- wait(FYVal) |
    unifyVar(XNum,XVal,FXVal, FYVal, T,NT).
alternatively.
unifyVarVar(XNum,XVal,FXVal, YNum,YVal,FYVal, T,NT) :- string_element(T,XNum,2#"1") |
    unifyVar(YNum,YVal,FYVal, XVal, T,NT).
unifyVarVar(XNum,XVal,FXVal, YNum,YVal,FYVal, T,NT) :- string_element(T,YNum,2#"1") |
    unifyVar(XNum,XVal,FXVal, YVal, T,NT).
unifyVarVar(XNum,XVal,FXVal, YNum,YVal,FYVal, T,NT) :-
    string_element(T,XNum,2#"0"), string_element(T,XNum,2#"0") |
    XVal = {YNum,YVal,FYVal}, FXVal = FYVal,
    set_string_element(T, XNum,2#"1", NT).

unifyVarStr(XNum,XVal,FXVal, Y, T,NT) :- wait(FXVal) | unify(FXVal, Y, T,NT).
alternatively.
unifyVarStr(XNum,XVal,FXVal, Y, T,NT) :- string_element(T,XNum,2#"0") |
    FXVal = Y, XVal = Y,
    set_string_element(T, XNum,2#"1", NT).
unifyVarStr(XNum,XVal,FXVal, Y, T,NT) :- string_element(T,XNum,2#"1") |
    unify(XVal, Y, T,NT).
```

A.11 GroundUnifyPeval

```
:- module unify. :- public unify/3,unify/4,unify/5.

unify(_,_,_) :- true | true . unify(_,_,_,_,_) :- true | true .

unify(X,Y, fail,NT) :- true | NT = fail.
otherwise.
unify({N},Y, T,NT) :- integer(N) | unifyVar(N, Y, T,NT).
unify(X,{M}, T,NT) :- integer(M) | unifyVar(M, X, T,NT).
unify([XH|XT],[YH|YT], T,NT) :- true |
    unify(XH,YH, T,T1), unify(XT,YT, T1,NT).
unify({F,X},{F,Y}, T,NT) :- true | unify(X,Y, T,NT).
unify({F,X,X1},{F,Y,Y1}, T,NT) :- true |
    unify(X,Y, T,T1), unify(X1,Y1, T1,NT).
unify({F,X,X1,X2},{F,Y,Y1,Y2}, T,NT) :- true |
    unify(X,Y, T,T1), unify(X1,Y1, T1,T2), unify(X2,Y2, T2,NT).
unify({F,X,X1,X2,X3},{F,Y,Y1,Y2,Y3}, T,NT) :- true |
    unify(X,Y, T,T1), unify(X1,Y1, T1,T2),
    unify(X2,Y2, T2,T3), unify(X3,Y3, T3,NT).
unify({F,X,X1,X2,X3,X4},{F,Y,Y1,Y2,Y3,Y4}, T,NT) :- true |
    unify(X,Y, T,T1), unify(X1,Y1, T1,T2),
    unify(X2,Y2, T2,T3), unify(X3,Y3, T3,T4), unify(X4,Y4, T4,NT).
unify(X,Y, T,NT) :- vector(X, Size), vector(Y, Size), Size > 6,
    vector_element(X,0,F), vector_element(Y,0,F), atom(F)|
    unifyArgs(1,Size, X,Y, T,NT).
otherwise.
unify(X,X, T,NT) :- true | NT = T.
otherwise.
unify(X,Y, _,NT) :- true | NT = fail.

unifyArgs(M,N, X,Y, T,NT) :-
    vector_element(X, M, Xm), vector_element(Y, M, Ym), M1 := M + 1 |
    unify(Xm,Ym, T,T1),
    unifyArgs(M1,N, X,Y, T1,NT).
unifyArgs(N,N, _,_, T,NT) :- true | NT = T.

unifyVar(XNum, {YNum}, T,NT) :- integer(YNum) | unifyVarVar(XNum,YNum, T,NT).
otherwise.
unifyVar(XNum, Y, T,NT) :- true | unifyVarStr(XNum,Y, T,NT).

unifyVarVar(XNum,XNum, T,NT) :- true | NT = T.
otherwise.
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,XNum,point(XRef)) |
    unifyVarVar(XRef,YNum, T,NT).
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,XNum,data(XVal)) |
    unifyVarStr(YNum,XVal, T,NT).
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,YNum,point(YRef)) |
    unifyVarVar(XNum,YRef, T,NT).
unifyVarVar(XNum,YNum, T,NT) :- vector_element(T,YNum,data(YVal)) |
    unifyVarStr(XNum,YVal, T,NT).
unifyVarVar(XNum,YNum, T,NT) :-
    vector_element(T,XNum,0), vector_element(T,YNum,0) |
    set_vector_element(T,XNum, _,point(YNum), NT).

unifyVarStr(XNum,Y, T,NT) :- vector_element(T,XNum,point(XRef)) |
    unifyVarStr(XRef,Y, T,NT).
```

```
unifyVarStr(XNum,Y, T,NT) :- vector_element(T,XNum,data(XVal)) |
    unify(XVal,Y, T,NT).
unifyVarStr(XNum,Y, T,NT) :- vector_element(T,XNum,0) |
    set_vector_element(T,XNum, _,data(Y),NT).
```

A.12 測定用プログラム

```
:- module utest. :- public do/3,do/4.

%
%   do(+UnifierType,+ProblemID, -Time)
%
do(Type,6, T) :- true | do0(100000, Type,6, T).
otherwise.
do(Type,Prob, T) :- true | do0(10000, Type,Prob, T).

do0(N, -1,Prob, T) :- true | do(N, {0,Prob,_},-1, T).
do0(N, 0,Prob, T) :- true | do(N, {0,Prob,_},0, T).
do0(N, 1,Prob, T) :- true | do(N, {1,Prob,vector},1, T).
do0(N, cont,Prob, T) :- true | do(N, {1,Prob,vector},2, T).
do0(N, 2,Prob, T) :- true | do(N, {2,Prob,vector},1, T).
do0(N, -2,Prob, T) :- true | do(N, {2,Prob,string},1, T).
do0(N, 3,Prob, T) :- true | do(N, {3,Prob,vector},1, T).
do0(N, -3,Prob, T) :- true | do(N, {3,Prob,string},1, T).
do0(N, 4,Prob, T) :- true | do(N, {4,Prob,vector},1, T).


%
%   There are several representations of object's variable
%
%       Type 0 : KL1 Variable
%            1 : {N}
%            2 : {N,Val}
%            3 : {N,Val,FVal}
%            4 : '$VAR'(N)
%
%   There are two environment type Vector and String.
%
%       Kind -1 : builtin
%             0 : unify:unify/3.
%             1 : unify:unify/4.
%             2 : unify:unify/5.
%
do(N, Info,Kind, T) :- N > 0, system_timer(_,T0) |
    empty(N, Info,Kind),
    do1(N, Info,Kind, T0,T).

do1(N, Info,Kind, T0,T) :- system_timer(_,T1) |
    body(N, Info,Kind),
    do2(T0,T1, T).

do2(T0,T1, T) :- system_timer(_,T2) | T := (T2-T1-T1+T0)/16.

%
%   empty(+Count, +{Type,ProblemID,EnvType},+UnifyKind)
%
```

```
empty(N, Info,Kind) :- N > 0 |
    prob(Info, Tuple),
    unifyEmpty(Kind, Tuple),
    empty(~(N-1), Info,Kind).
empty(0, _,_) :- true | true.

unifyEmpty(_, {_,_,_}) :- true | true.

%
%   body(+Count, +{Type,ProblemID,EnvType},+UnifyKind)
%
body(N, Info,Kind) :- N > 0 |
    prob(Info, Tuple),
    unify(Kind, Tuple),
    body(~(N-1), Info,Kind).
body(0, _,_) :- true | true.

unify(-1, {X,Y,_}) :- true | X = Y.
unify(0, {X,Y,_}) :- true | unify:unify(X,Y,_).
unify(1, {X,Y,Env}) :- true | unify:unify(X,Y, Env,_).
unify(2, {X,Y,Env}) :- true | unify:unify(X,Y, □, Env,_).

%
%   Problems
%
%   prob(+{Type,ProblemID,EnvType}, -{X,Y,Environment})
%
% problems for KL1
prob({0,0,_}, Tuple) :- true | Tuple = {X,Y,0},
    X = i(A,i(A,A)),
    Y = i(i(i(B,C),D),i(i(D,B),i(E,B))).
prob({0,1,_}, Tuple) :- true | Tuple = {X,Y,0},
    X = i(i(X1,Y1),Z1),
    Y = i(i(i(B,C),D),i(i(D,B),i(E,B))).
prob({0,2,_}, Tuple) :- true | Tuple = {X,Y,_},
 X= i(i(i(i(i(V1,V2),i(V3,V4)),i(i(V5,V6),i(V7,V8))),
        i(i(i(V9,V10),i(V11,V12)),i(i(V13,V14),i(V15,V16)))),
      i(i(i(i(V17,V18),i(V19,V20)),i(i(V21,V22),i(V23,V24))),
        i(i(i(V25,V26),i(V27,V28)),i(i(V29,V30),i(V31,V32))))),
 Y= i(i(i(i(i(1,2),i(3,4)),i(i(5,6),i(7,8))),i(i(i(9,10),i(11,12)),i(i(13,14),i(15,16)))),
      i(i(i(i(17,18),i(19,20)),i(i(21,22),i(23,24))),i(i(i(25,26),i(27,28)),i(i(29,30),i(31,32))))).
prob({0,3,_}, Tuple) :- true | Tuple = {X,Y,0},
    X=p(h(X1,X1),h(X2,X2),Y2,Y3,X3),
    Y=p(X2,X3,h(Y1,Y1),h(Y2,Y2),Y3).
prob({0,4,_}, Tuple) :- true | Tuple = {A,B,0},
    A=i(i(X,X),i(i(Y,Y),i(V,i(W,Z)))),
    B=i(Y,i(Z,i(i(U,U),i(i(V,V),W)))).
prob({0,5,_}, Tuple) :- true | Tuple = {A,B,_},
    A= (X0,X1,X2,X3,X4,X5,X6,f(X0),f(X6)),
    B= (X1,X2,X3,X4,X5,X6,a,f(X6),f(X0)).
prob({0,6,_}, Tuple) :- true | Tuple = {A,B,0},
    A= X, B= f(a).
prob({0,7,_}, Tuple) :- true | Tuple = {A,B,0},
    A= [X,x,Y,y,Z,z,U,u,V,v,W,w],
    B= [x,X,y,Y,z,Z,u,U,v,V,w,W].
prob({0,8,_}, Tuple) :- true | Tuple = {A,B,0},
```

```
    A= f(X,x,Y,y,Z,z,U,u,V,v,W,w),
    B= f(x,X,y,Y,z,Z,u,U,v,V,w,W).
prob({0,9,_}, Tuple) :- true | Tuple = {A,B,0},
    A= [X0,X1,X2,X3,X4,X5,X6,X7,X8,X9],
    B= [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9].
prob({0,10,_}, Tuple) :- true | Tuple = {A,B,0},
    A= f(X0,X1,X2,X3,X4,X5,X6,X7,X8,X9),
    B= f( 0, 1, 2, 3, 4, 5, 6, 7, 8, 9).
%  problems for {N}
prob({1,0,EnvType}, Tuple) :- true | Tuple = {X,Y,Env},
    X = i({0},i({0},{0})),
    Y = i(i(i({1},{2}),{3}),i(i({3},{1}),i({4},{1}))),
    new_env(EnvType,5, Env).
prob({1,1,EnvType}, Tuple) :- true | Tuple = {X,Y,Env},
    X = i(i({0},{1}),{2}),
    Y = i(i(i({3},{4}),{5}),i(i({5},{3}),i({6},{3}))),
    new_env(EnvType,7, Env).
prob({1,2,EnvType}, Tuple) :- true | Tuple = {X,Y,Env},
 X= i(i(i(i(i({1},{2}),i({3},{4})),i(i({5},{6}),i({7},{8}))),
        i(i(i({9},{10}),i({11},{12})),i(i({13},{14}),i({15},{16})))),
      i(i(i(i({17},{18}),i({19},{20})),i(i({21},{22}),i({23},{24}))),
        i(i(i({25},{26}),i({27},{28})),i(i({29},{30}),i({31},{0}))))),
 Y= i(i(i(i(i(1,2),i(3,4)),i(i(5,6),i(7,8))),i(i(i(9,10),i(11,12)),i(i(13,14),i(15,16)))),
      i(i(i(i(17,18),i(19,20)),i(i(21,22),i(23,24))),i(i(i(25,26),i(27,28)),i(i(29,30),i(31,32))))),
    new_env(EnvType,32, Env).
prob({1,3,EnvType}, Tuple) :- true | Tuple = {X,Y,Env},
    X=p(h({0},{0}),h({1},{1}),{4},{5},{2}),
    Y=p({1},{2},h({3},{3}),h({4},{4}),{5}),
    new_env(EnvType,6, Env).
prob({1,4,EnvType}, Tuple) :- true | Tuple = {A,B,Env},
    A=i(i({0},{0}),i(i({1},{1}),i({4},i({5},{2})))),
    B=i({1},i({2},i(i({3},{3}),i(i({4},{4}),{5})))),
    new_env(EnvType,6, Env).
prob({1,5,EnvType}, Tuple) :- true | Tuple = {A,B,Env},
    A= ({0},{1},{2},{3},{4},{5},{6},f({0}),f({6})),
    B= ({1},{2},{3},{4},{5},{6},a,f({6}),f({0})),
    new_env(EnvType,7, Env).
prob({1,6,EnvType}, Tuple) :- true | Tuple = {A,B,Env},
    A= {0},
    B= f(a),
    new_env(EnvType,1, Env).
prob({1,7,EnvType}, Tuple) :- true | Tuple = {A,B, Env},
    A= [{0},x,{1},y,{2},z,{3},u,{4},v,{5},w],
    B= [x,{0},y,{1},z,{2},u,{3},v,{4},w,{5}],
    new_env(EnvType,6, Env).
prob({1,8,EnvType}, Tuple) :- true | Tuple = {A,B, Env},
    A= f({0},x,{1},y,{2},z,{3},u,{4},v,{5},w),
    B= f(x,{0},y,{1},z,{2},u,{3},v,{4},w,{5}),
    new_env(EnvType,6, Env).
prob({1,9,EnvType}, Tuple) :- true | Tuple = {A,B, Env},
    A= [{0},{1},{2},{3},{4},{5},{6},{7},{8},{9}],
    B= [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
    new_env(EnvType,10, Env).
prob({1,10,EnvType}, Tuple) :- true | Tuple = {A,B, Env},
    A= f({0},{1},{2},{3},{4},{5},{6},{7},{8},{9}),
    B= f( 0, 1, 2, 3, 4, 5, 6, 7, 8, 9),
```

```
    new_env(EnvType,10, Env).
% problems for {N,Val}
prob({2,0,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = i({0,X0},i({0,X0},{0,X0})),
    Y = i(i(i({1,X1},{2,X2}),{3,X3}),i(i({3,X3},{1,X1}),i({4,X4},{1,X1}))),
    new_env(EnvType,5, Env).
prob({2,1,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = i(i({0,X0},{1,X1}),{2,X2}),
    Y = i(i(i({3,X3},{4,X4}),{5,X5}),i(i({5,X5},{3,X3}),i({6,X6},{3,X3}))),
    new_env(EnvType,7, Env).
prob({2,2,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
 X= i(i(i(i(i({1,X1},{2,X2}),i({3,X3},{4,X4})),i(i({5,X5},{6,X6}),i({7,X7},{8,X8}))),
        i(i(i({9,X9},{10,X10}),i({11,X11},{12,X12})),i(i({13,X13},{14,X14}),i({15,X15},{16,X16})))),
      i(i(i(i({17,X17},{18,X18}),i({19,X19},{20,X20})),i(i({21,X21},{22,X22}),i({23,X23},{24,X24}))),
        i(i(i({25,X25},{26,X26}),i({27,X27},{28,X28})),i(i({29,X29},{30,X30}),i({31,X31},{0,X0}))))),
 Y= i(i(i(i(i(1,2),i(3,4)),i(i(5,6),i(7,8))),i(i(i(9,10),i(11,12)),i(i(13,14),i(15,16)))),
      i(i(i(i(17,18),i(19,20)),i(i(21,22),i(23,24))),i(i(i(25,26),i(27,28)),i(i(29,30),i(31,32))))),
    new_env(EnvType,32, Env).
prob({2,3,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X=p(h({0,X0},{0,X0}),h({1,X1},{1,X1}),{4,X4},{5,X5},{2,X2}),
    Y=p({1,X1},{2,X2},h({3,X3},{3,X3}),h({4,X4},{4,X4}),{5,X5}),
    new_env(EnvType,6, Env).
prob({2,4,EnvType}, Tuple) :- true | Tuple = {A,B,Env},
    A=i(i({0,X0},{0,X0}),i(i({1,X1},{1,X1}),i({4,X4},i({5,X5},{2,X2})))),
    B=i({1,X1},i({2,X2},i(i({3,X3},{3,X3}),i(i({4,X4},{4,X4}),{5,X5})))),
    new_env(EnvType,6, Env).
prob({2,5,EnvType}, Tuple) :- true | Tuple = {A,B, Env},
    A= ({0,X0},{1,X1},{2,X2},{3,X3},{4,X4},{5,X5},{6,X6},f({0,X0}),f({6,X6})),
    B= ({1,X1},{2,X2},{3,X3},{4,X4},{5,X5},{6,X6},a,f({6,X6}),f({0,X0})),
    new_env(EnvType,7, Env).
prob({2,6,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = {0,X0},
    Y = f(a),
    new_env(EnvType,1, Env).
prob({2,7,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = [{0,X0},x,{1,X1},y,{2,X2},z,{3,X3},u,{4,X4},v,{5,X5},w],
    Y = [x,{0,X0},y,{1,X1},z,{2,X2},u,{3,X3},v,{4,X4},w,{5,X5}],
    new_env(EnvType,6, Env).
prob({2,8,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = f({0,X0},x,{1,X1},y,{2,X2},z,{3,X3},u,{4,X4},v,{5,X5},w),
    Y = f(x,{0,X0},y,{1,X1},z,{2,X2},u,{3,X3},v,{4,X4},w,{5,X5}),
    new_env(EnvType,6, Env).
prob({2,9,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = [{0,X0},{1,X1},{2,X2},{3,X3},{4,X4},{5,X5},{6,X6},{7,X7},{8,X8},{9,X9}],
    Y = [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
    new_env(EnvType,10, Env).
prob({2,10,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = f({0,X0},{1,X1},{2,X2},{3,X3},{4,X4},{5,X5},{6,X6},{7,X7},{8,X8},{9,X9}),
    Y = f( 0, 1, 2, 3, 4, 5, 6, 7, 8, 9),
    new_env(EnvType,10, Env).
% problems for {N,Val,FVal}
prob({3,0,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = i({0,X0,Y0},i({0,X0,Y0},{0,X0,Y0})),
    Y = i(i(i({1,X1,Y1},{2,X2,Y2}),{3,X3,Y3}),i(i({3,X3,Y3},{1,X1,Y1}),i({4,X4,Y4},{1,X1,Y1}))),
    new_env(EnvType,5, Env).
prob({3,1,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
```

```
    X = i(i({0,X0,Y0},{1,X1,Y1}),{2,X2,Y2}),
    Y = i(i(i({3,X3,Y3},{4,X4,Y4}),{5,X5,Y5}),i(i({5,X5,Y5},{3,X3,Y3}),i({6,X6,Y6},{3,X3,Y3}))),
    new_env(EnvType,7, Env).
prob({3,2,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
 X= i(i(i(i(i({1,X1,Y1},{2,X2,Y2}),i({3,X3,Y3},{4,X4,Y4})),
          i(i({5,X5,Y5},{6,X6,Y6}),i({7,X7,Y7},{8,X8,Y8}))),
        i(i(i({9,X9,Y9},{10,X10,Y10}),i({11,X11,Y11},{12,X12,Y12})),
          i(i({13,X13,Y13},{14,X14,Y14}),i({15,X15,Y15},{16,X16,Y16})))),
      i(i(i(i({17,X17,Y17},{18,X18,Y18}),i({19,X19,Y19},{20,X20,Y20})),
          i(i({21,X21,Y21},{22,X22,Y22}),i({23,X23,Y23},{24,X24,Y24}))),
        i(i(i({25,X25,Y25},{26,X26,Y26}),i({27,X27,Y27},{28,X28,Y28})),
          i(i({29,X29,Y29},{30,X30,Y30}),i({31,X31,Y31},{0,X0,Y0}))))),
 Y= i(i(i(i(i(1,2),i(3,4)),i(i(5,6),i(7,8))),i(i(i(9,10),i(11,12)),i(i(13,14),i(15,16)))),
    i(i(i(i(17,18),i(19,20)),i(i(21,22),i(23,24))),i(i(i(25,26),i(27,28)),i(i(29,30),i(31,32))))),
    new_env(EnvType,32, Env).
prob({3,3,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X=p(h({0,X0,Y0},{0,X0,Y0}),h({1,X1,Y1},{1,X1,Y1}),{4,X4,Y4},{5,X5,Y5},{2,X2,Y2}),
    Y=p({1,X1,Y1},{2,X2,Y2},h({3,X3,Y3},{3,X3,Y3}),h({4,X4,Y4},{4,X4,Y4}),{5,X5,Y5}),
    new_env(EnvType,6, Env).
prob({3,4,EnvType}, Tuple) :- true | Tuple = {A,B,Env},
    A=i(i({0,X0,Y0},{0,X0,Y0}),i(i({1,X1,Y1},{1,X1,Y1}),i({4,X4,Y4},i({5,X5,Y5},{2,X2,Y2})))),
    B=i({1,X1,Y1},i({2,X2,Y2},i(i({3,X3,Y3},{3,X3,Y3}),i(i({4,X4,Y4},{4,X4,Y4}),{5,X5,Y5})))),
    new_env(EnvType,6, Env).
prob({3,5,EnvType}, Tuple) :- true | Tuple = {A,B, Env},
    A= ({0,X0,Y0},{1,X1,Y1},{2,X2,Y2},{3,X3,Y3},{4,X4,Y4},{5,X5,Y5},{6,X6,Y6},f({0,X0,Y0}),f({6,X6,Y6})),
    B= ({1,X1,Y1},{2,X2,Y2},{3,X3,Y3},{4,X4,Y4},{5,X5,Y5},{6,X6,Y6},a,f({6,X6,Y6}),f({0,X0,Y0})),
    new_env(EnvType,7, Env).
prob({3,6,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = {0,X0,Y0},
    Y = f(a),
    new_env(EnvType,1, Env).
prob({3,7,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = [{0,X0,Y0},x,{1,X1,Y1},y,{2,X2,Y2},z,{3,X3,Y3},u,{4,X4,Y4},v,{5,X5,Y5},w],
    Y = [x,{0,X0,Y0},y,{1,X1,Y1},z,{2,X2,Y2},u,{3,X3,Y3},v,{4,X4,Y4},w,{5,X5,Y5}],
    new_env(EnvType,6, Env).
prob({3,8,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = f({0,X0,Y0},x,{1,X1,Y1},y,{2,X2,Y2},z,{3,X3,Y3},u,{4,X4,Y4},v,{5,X5,Y5},w),
    Y = f(x,{0,X0,Y0},y,{1,X1,Y1},z,{2,X2,Y2},u,{3,X3,Y3},v,{4,X4,Y4},w,{5,X5,Y5}),
    new_env(EnvType,6, Env).
prob({3,9,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = [{0,X0,Y0},{1,X1,Y1},{2,X2,Y2},{3,X3,Y3},{4,X4,Y4},{5,X5,Y5},
         {6,X6,Y6},{7,X7,Y7},{8,X8,Y8},{9,X9,Y9}],
    Y = [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
    new_env(EnvType,10, Env).
prob({3,10,EnvType}, Tuple) :- true | Tuple = {X,Y, Env},
    X = f({0,X0,Y0},{1,X1,Y1},{2,X2,Y2},{3,X3,Y3},{4,X4,Y4},{5,X5,Y5},
          {6,X6,Y6},{7,X7,Y7},{8,X8,Y8},{9,X9,Y9}),
    Y = f( 0, 1, 2, 3, 4, 5, 6, 7, 8, 9),
    new_env(EnvType,10, Env).
%  problems for '$VAR'(N)
prob({4,0,EnvType}, Tuple) :- true | Tuple = {X,Y,Env},
    X = i('$VAR'(0),i('$VAR'(0),'$VAR'(0))),
    Y = i(i(i('$VAR'(1),'$VAR'(2)),'$VAR'(3)),i(i('$VAR'(3),'$VAR'(1)),i('$VAR'(4),'$VAR'(1)))),
    new_env(EnvType,5, Env).
prob({4,1,EnvType}, Tuple) :- true | Tuple = {X,Y,Env},
    X = i(i('$VAR'(0),'$VAR'(1)),'$VAR'(2)),
```

```
    Y = i(i(i('$VAR'(3),'$VAR'(4)),'$VAR'(5)),i(i('$VAR'(5),'$VAR'(3)),i('$VAR'(6),'$VAR'(3)))),
    new_env(EnvType,7, Env).
prob({4,2,EnvType}, Tuple) :- true | Tuple = {X,Y,Env},
 X= i(i(i(i('$VAR'(1),'$VAR'(2)),i('$VAR'(3),'$VAR'(4))),
          i(i('$VAR'(5),'$VAR'(6)),i('$VAR'(7),'$VAR'(8)))),
        i(i(i('$VAR'(9),'$VAR'(10)),i('$VAR'(11),'$VAR'(12))),
          i(i('$VAR'(13),'$VAR'(14)),i('$VAR'(15),'$VAR'(16))))),
      i(i(i(i('$VAR'(17),'$VAR'(18)),i('$VAR'(19),'$VAR'(20))),
          i(i('$VAR'(21),'$VAR'(22)),i('$VAR'(23),'$VAR'(24)))),
        i(i(i('$VAR'(25),'$VAR'(26)),i('$VAR'(27),'$VAR'(28))),
          i(i('$VAR'(29),'$VAR'(30)),i('$VAR'(31),'$VAR'(0))))))),
 Y= i(i(i(i(i(1,2),i(3,4)),i(i(5,6),i(7,8))),i(i(i(9,10),i(11,12)),i(i(13,14),i(15,16)))),
      i(i(i(i(17,18),i(19,20)),i(i(21,22),i(23,24))),i(i(i(25,26),i(27,28)),i(i(29,30),i(31,32))))),
    new_env(EnvType,32, Env).
prob({4,3,EnvType}, Tuple) :- true | Tuple = {X,Y,Env},
    X=p(h('$VAR'(0),'$VAR'(0)),h('$VAR'(1),'$VAR'(1)),'$VAR'(4),'$VAR'(5),'$VAR'(2)),
    Y=p('$VAR'(1),'$VAR'(2),h('$VAR'(3),'$VAR'(3)),h('$VAR'(4),'$VAR'(4)),'$VAR'(5)),
    new_env(EnvType,6, Env).
prob({4,4,EnvType}, Tuple) :- true | Tuple = {A,B,Env},
    A=i(i('$VAR'(0),'$VAR'(0)),i(i('$VAR'(1),'$VAR'(1)),i('$VAR'(4),i('$VAR'(5),'$VAR'(2))))),
    B=i('$VAR'(1),i('$VAR'(2),i(i('$VAR'(3),'$VAR'(3)),i(i('$VAR'(4),'$VAR'(4)),'$VAR'(5))))),
    new_env(EnvType,6, Env).
prob({4,5,EnvType}, Tuple) :- true | Tuple = {A,B,Env},
    A= ('$VAR'(0),'$VAR'(1),'$VAR'(2),'$VAR'(3),'$VAR'(4),'$VAR'(5),'$VAR'(6),f('$VAR'(0)),f('$VAR'(6))),
    B= ('$VAR'(1),'$VAR'(2),'$VAR'(3),'$VAR'(4),'$VAR'(5),'$VAR'(6),a,f('$VAR'(6)),f('$VAR'(0))),
    new_env(EnvType,7, Env).
prob({4,6,EnvType}, Tuple) :- true | Tuple = {A,B,Env},
    A= '$VAR'(0),
    B= f(a),
    new_env(EnvType,1, Env).
prob({4,7,EnvType}, Tuple) :- true | Tuple = {A,B, Env},
    A= ['$VAR'(0),x,'$VAR'(1),y,'$VAR'(2),z,'$VAR'(3),u,'$VAR'(4),v,'$VAR'(5),w],
    B= [x,'$VAR'(0),y,'$VAR'(1),z,'$VAR'(2),u,'$VAR'(3),v,'$VAR'(4),w,'$VAR'(5)],
    new_env(EnvType,6, Env).
prob({4,8,EnvType}, Tuple) :- true | Tuple = {A,B, Env},
    A= f('$VAR'(0),x,'$VAR'(1),y,'$VAR'(2),z,'$VAR'(3),u,'$VAR'(4),v,'$VAR'(5),w),
    B= f(x,'$VAR'(0),y,'$VAR'(1),z,'$VAR'(2),u,'$VAR'(3),v,'$VAR'(4),w,'$VAR'(5)),
    new_env(EnvType,6, Env).
prob({4,9,EnvType}, Tuple) :- true | Tuple = {A,B, Env},
    A= ['$VAR'(0),'$VAR'(1),'$VAR'(2),'$VAR'(3),
        '$VAR'(4),'$VAR'(5),'$VAR'(6),'$VAR'(7),'$VAR'(8),'$VAR'(9)],
    B= [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
    new_env(EnvType,10, Env).
prob({4,10,EnvType}, Tuple) :- true | Tuple = {A,B, Env},
    A= f('$VAR'(0),'$VAR'(1),'$VAR'(2),'$VAR'(3),
        '$VAR'(4),'$VAR'(5),'$VAR'(6),'$VAR'(7),'$VAR'(8),'$VAR'(9)),
    B= f( 0, 1, 2, 3, 4, 5, 6, 7, 8, 9),
    new_env(EnvType,10, Env).

new_env(string,Size, Env) :- true | new_string(Env,Size,1).
new_env(vector,Size, Env) :- true | new_vector(Env,Size).
```