グラフ書き換え言語 LMNtal における閉包計算の冗長なマッチング削減手法

早稲田大学 基幹理工学部
情報理工学科 4年 上田研究室
1W182043-1 今川　連

本発表の概要

- 背景
  - グラフ書き換え言語 **LMNtal**
    - ルールに記述したグラフ構造を繰り返し探し，書き換える
  - **閉包計算**を行うプログラム
    - 閉包：与えられたタプルの多重集合と，
      そこからルールによって導出可能なタプルをすべて含む最小の多重集合
    - 既に探索を行ったグラフ構造に対して再度マッチングを試してしまう場合がある

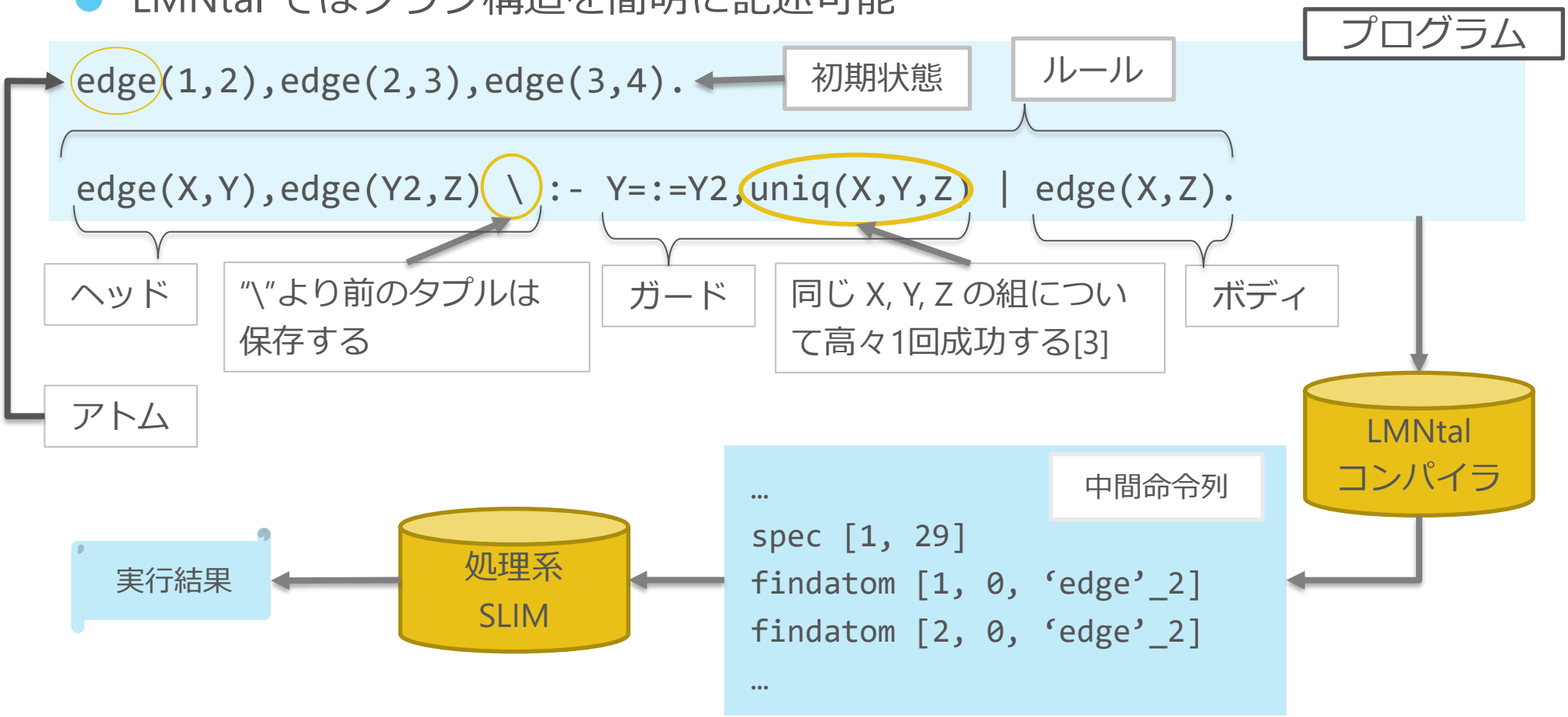
グラフ構造を探すことを**マッチング**という

重要な応用として
タプルの書き換え
がある

- 目的
  - 閉包計算を行うプログラムにおける，冗長なマッチングの削減
- 貢献
  - 差分アトムリストを用いたマッチング手法により，閉包計算を行うプログラムの冗長なマッチングを**すべて削減**
    （既存のプログラムの**3分の1の実行時間**で実行できた）

背景：グラフ書き換え言語 LMNtal [1]と処理系 SLIM[2]

- LMNtal ではグラフ構造を簡明に記述可能



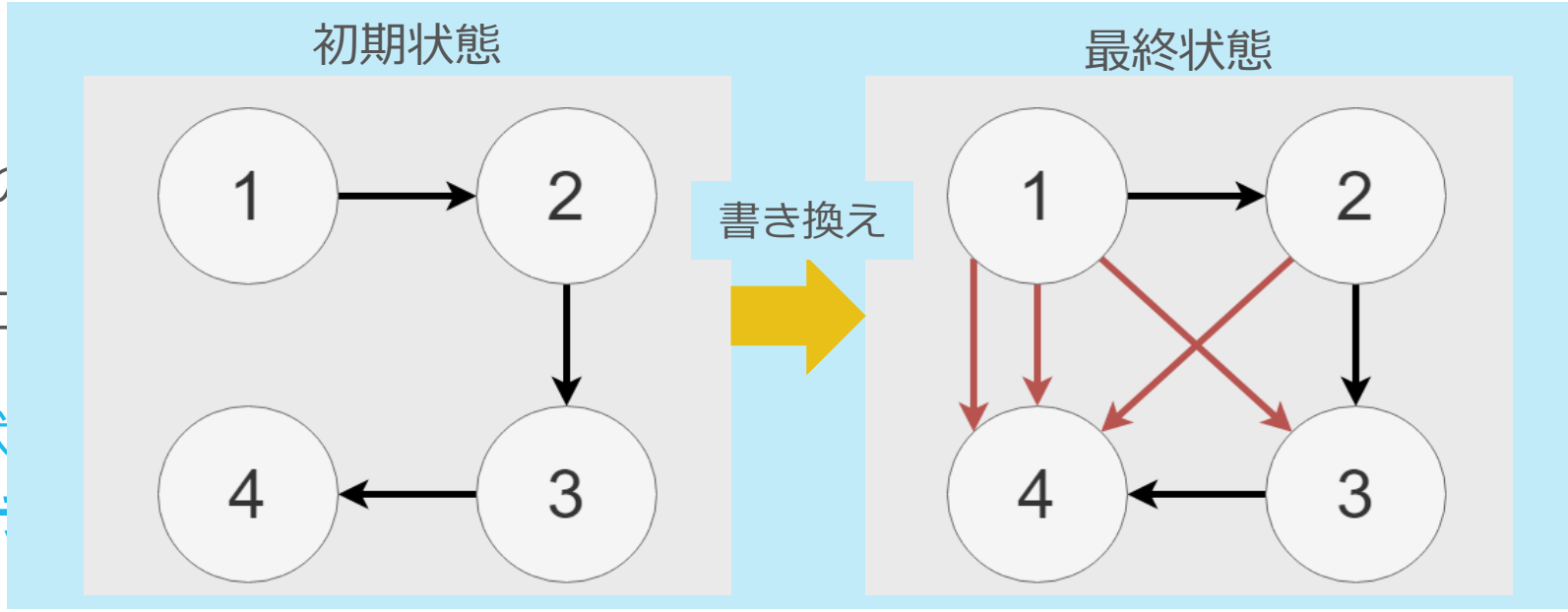
[1]上田和紀，加藤紀夫．言語モデル LMNtal．コンピュータソフトウェア，Vol. 21，No. 2，pp. 126-142, 2004.
[2]石川力 et al. 軽量な LMNtal 実行時処理系 SLIM の設計と実装. 情報処理学会第70回全国大会講演論文集. 153-154. 2008
[3]小川誠司. LMNtal 実行時処理系 SLIM への uniq 制約の導入. 卒業論文. 2009.

背景：閉包計算を行うプログラム

- **閉包**：与えられたタプルの多重集合と，そこからルールによって導出可能なタプルをすべて含む最小の多重集合
  - 右のプログラムのルールは
    与えられた初期状態の
    閉包を計算する

```
edge(1,2),edge(2,3),edge(3,4).
edge(X,Y),edge(Y2,Z)\:-
Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

  - 最終状態が初期状態の閉包になる



1. 元の
2. プロ

初期状
マッ

ムに

削減

ムは

背景：閉包計算を行うプログラム

- **閉包**：与えられたタプルの多重集合と，そこからルールによって導出可能なタプルをすべて含む最小の多重集合
  - 右のプログラムのルールは
    与えられた初期状態の
    閉包を計算する
  - 最終状態が初期状態の閉包になる

```
edge(1,2),edge(2,3),edge(3,4).
edge(X,Y),edge(Y2,Z)\:-
Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

1. 元のアトムは消さずに残す必要がある.
2. プログラムを停止させるため，uniq が必要.

初期状態が大きくなると**探索しなければいけない**
マッチングパターンが指数的に増加

このようなプログラムにおいて，
マッチングの無駄を削減したい
（ただし左辺のアトムは2つとする）

書き換えの様子と冗長なマッチング

プログラム

```
edge(1,2),edge(2,3),edge(3,4).
edge(X,Y),edge(Y2,Z)\:-
Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

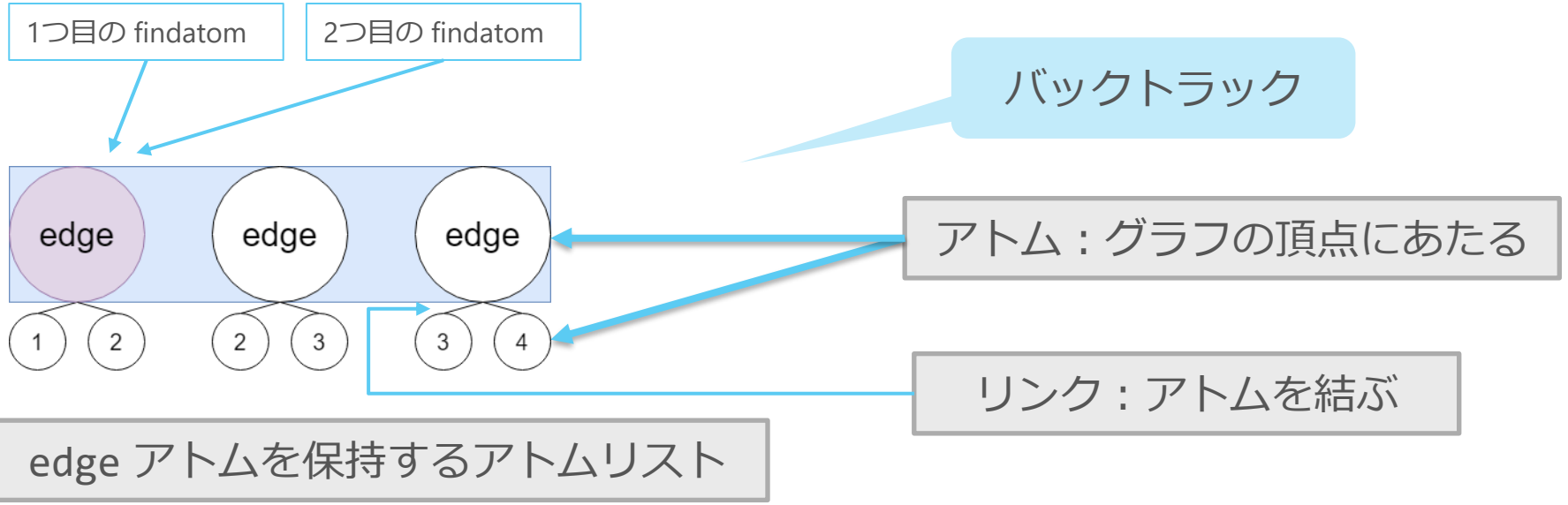
中間命令列

```
…
spec [1, 29]
findatom [1, 0, ‘edge’_2]
findatom [2, 0, ‘edge’_2]
…
```

findatom：
アトムリストの先頭からアトムを参照.
マッチング失敗したら次のアトムを参照する
バックトラックを繰り返す.

アトムリスト：
アトム名ごとに
アトムを保持するリスト

edge 探索中の edge(X,Y) を示すアトム　edge 探索中の edge(Y2,Z) を示すアトム　edge 両方が重なっているとき



書き換えの様子と冗長なマッチング

プログラム

```
edge(1,2),edge(2,3),edge(3,4).
edge(X,Y),edge(Y2,Z)\:-
Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

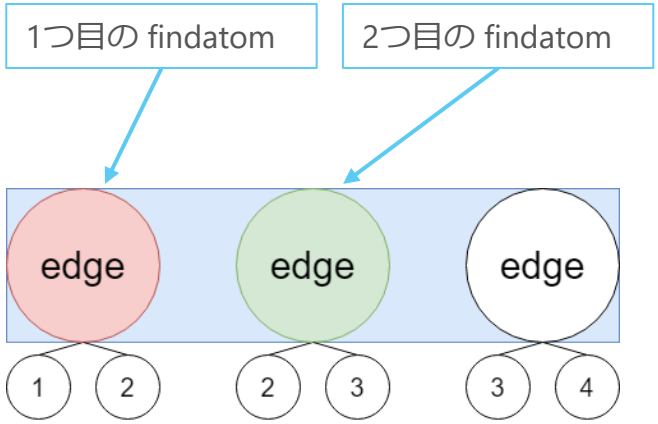
中間命令列

```
…
spec [1, 29]
findatom [1, 0, ‘edge’_2]
findatom [2, 0, ‘edge’_2]
…
```

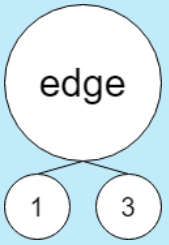
findatom：
アトムリストの先頭からアトムを参照.
マッチング失敗したら次のアトムを参照する
バックトラックを繰り返す.

アトムリスト：
アトム名ごとに
アトムを保持するリスト

edge 探索中の edge(X,Y) を示すアトム　edge 探索中の edge(Y2,Z) を示すアトム　edge 両方が重なっているとき



マッチング成功



を追加

書き換えの様子と冗長なマッチング

プログラム

```
edge(1,2),edge(2,3),edge(3,4).
edge(X,Y),edge(Y2,Z)\:-
Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

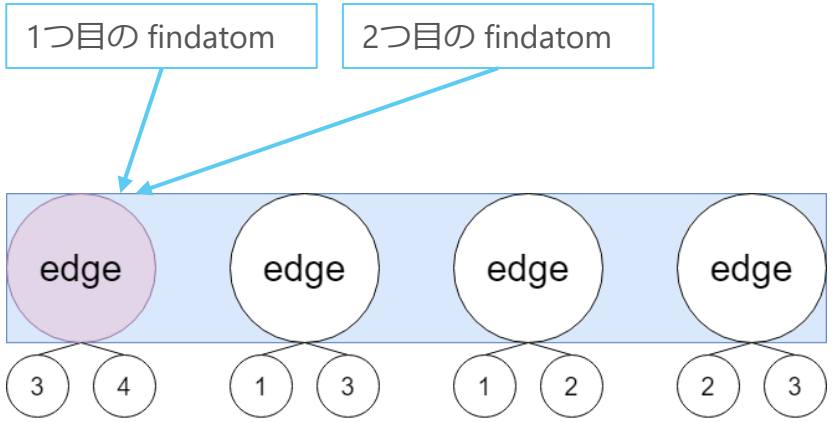
中間命令列

```
…
spec [1, 29]
findatom [1, 0, ‘edge’_2]
findatom [2, 0, ‘edge’_2]
…
```

findatom：
アトムリストの先頭からアトムを参照.
マッチング失敗したら次のアトムを参照する
バックトラックを繰り返す.

アトムリスト：
アトム名ごとに
アトムを保持するリスト

edge 探索中の edge(X,Y) を示すアトム　edge 探索中の edge(Y2,Z) を示すアトム　edge 両方が重なっているとき



もう一度初めから
マッチングを試す

バックトラック

書き換えの様子と冗長なマッチング

プログラム

```
edge(1,2),edge(2,3),edge(3,4).
edge(X,Y),edge(Y2,Z)\:-
Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

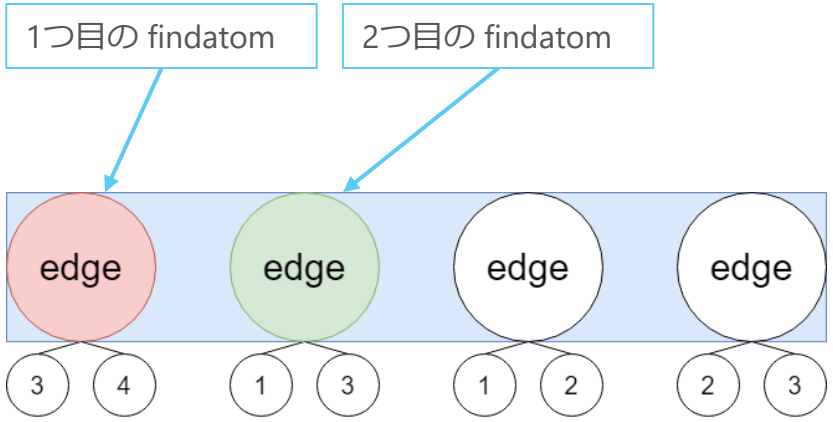
中間命令列

```
…
spec [1, 29]
findatom [1, 0, ‘edge’_2]
findatom [2, 0, ‘edge’_2]
…
```

findatom：
アトムリストの先頭からアトムを参照.
マッチング失敗したら次のアトムを参照する
バックトラックを繰り返す.

アトムリスト：
アトム名ごとに
アトムを保持するリスト

edge 探索中の edge(X,Y) を示すアトム　edge 探索中の edge(Y2,Z) を示すアトム　edge 両方が重なっているとき



バックトラック

書き換えの様子と冗長なマッチング

プログラム

```
edge(1,2),edge(2,3),edge(3,4).
edge(X,Y),edge(Y2,Z)\:-
Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

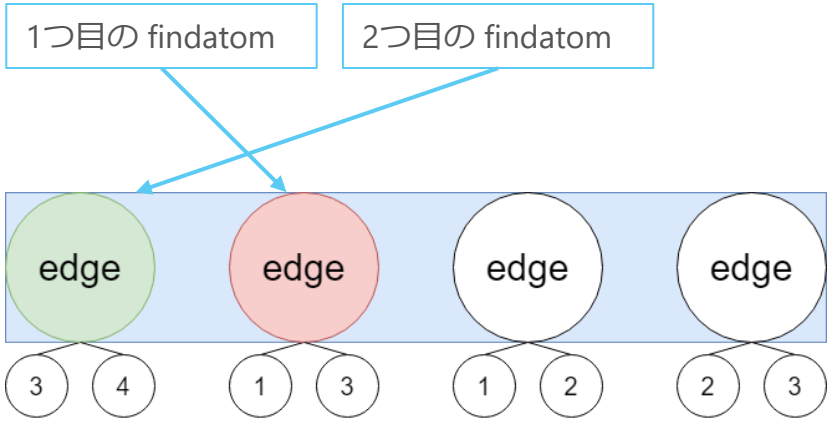
中間命令列

```
…
spec [1, 29]
findatom [1, 0, ‘edge’_2]
findatom [2, 0, ‘edge’_2]
…
```

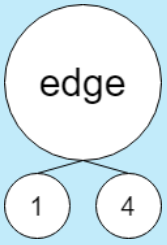
findatom：
アトムリストの先頭からアトムを参照.
マッチング失敗したら次のアトムを参照する
バックトラックを繰り返す.

アトムリスト：
アトム名ごとに
アトムを保持するリスト

edge 探索中の edge(X,Y) を示すアトム　edge 探索中の edge(Y2,Z) を示すアトム　edge 両方が重なっているとき



バックトラック
を繰り返し…
マッチング成功



を追加

書き換えの様子と冗長なマッチング

プログラム

```
edge(1,2),edge(2,3),edge(3,4).
edge(X,Y),edge(Y2,Z)\:-
Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

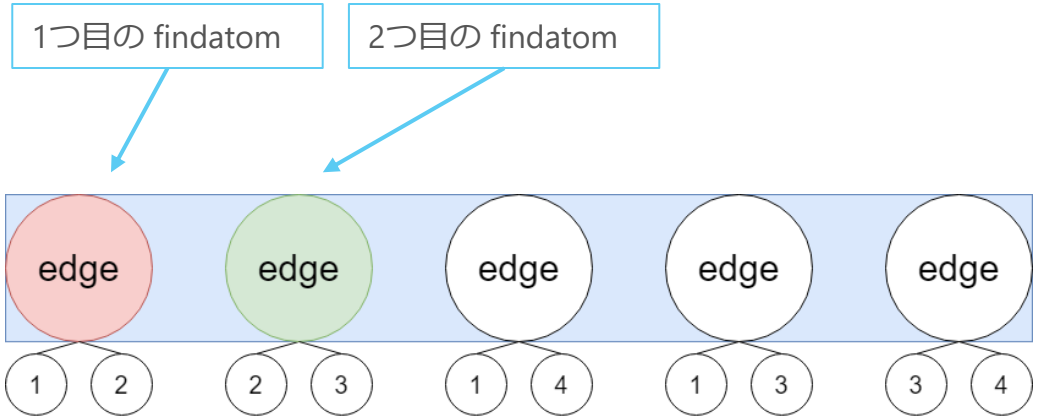
中間命令列

```
…
spec [1, 29]
findatom [1, 0, ‘edge’_2]
findatom [2, 0, ‘edge’_2]
…
```

findatom：
アトムリストの先頭からアトムを参照.
マッチング失敗したら次のアトムを参照する
バックトラックを繰り返す.

アトムリスト：
アトム名ごとに
アトムを保持するリスト

edge 探索中の edge(X,Y) を示すアトム　edge 探索中の edge(Y2,Z) を示すアトム　edge 両方が重なっているとき



バックトラックを繰り返すと
既に試したマッチングが出現

冗長なマッチング！
*最適化手法に履歴管理[4]がある

[4] 中田昌輝. グラフ書き換え言語 LMNtal における非連結サブパターンマッチング高速化手法. 卒業論文. 2021

書き換えの様子と冗長なマッチング

プログラム

```
edge(1,2),edge(2,3),edge(3,4).
edge(X,Y),edge(Y2,Z)\:-
Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

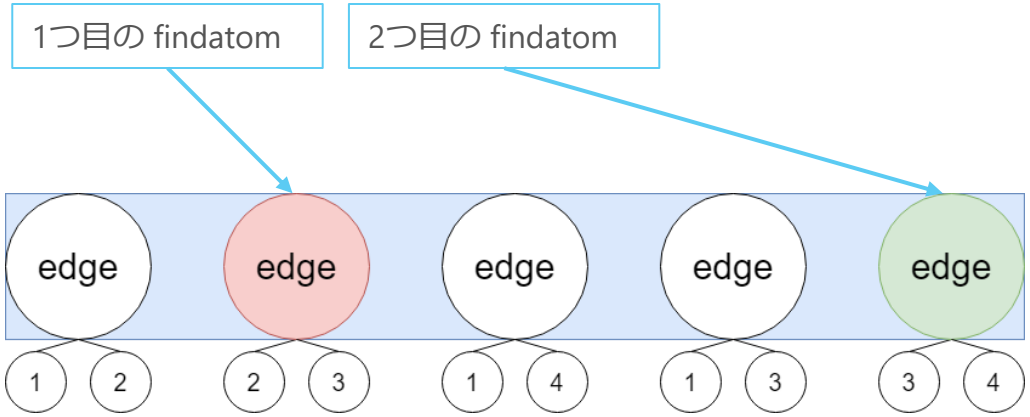
中間命令列

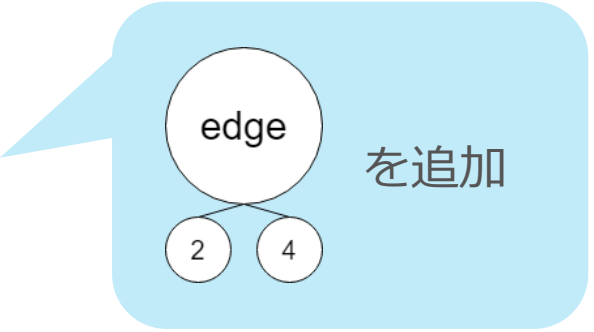
```
…
spec [1, 29]
findatom [1, 0, ‘edge’_2]
findatom [2, 0, ‘edge’_2]
…
```

findatom：
アトムリストの先頭からアトムを参照.
マッチング失敗したら次のアトムを参照する
バックトラックを繰り返す.

アトムリスト：
アトム名ごとに
アトムを保持するリスト

edge 探索中の edge(X,Y) を示すアトム　edge 探索中の edge(Y2,Z) を示すアトム　edge 両方が重なっているとき





書き換えの様子と冗長なマッチング

プログラム

```
edge(1,2),edge(2,3),edge(3,4).
edge(X,Y),edge(Y2,Z)\:-
Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

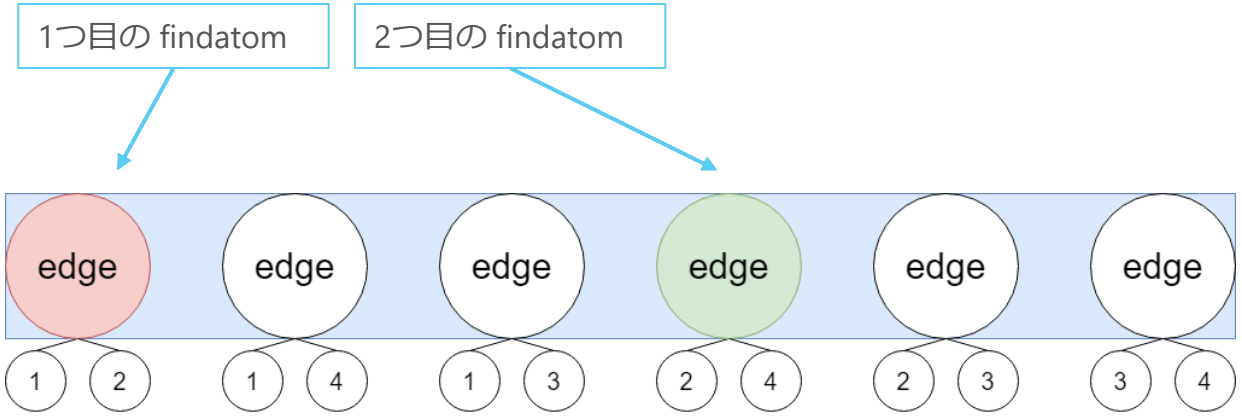
中間命令列

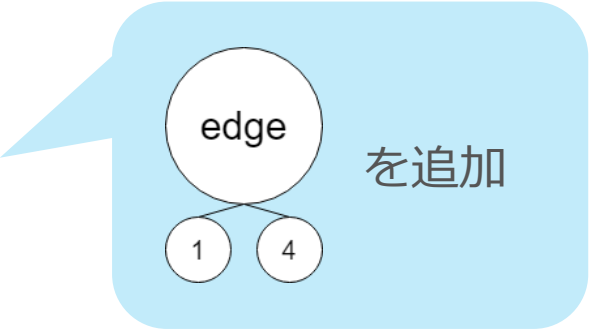
```
…
spec [1, 29]
findatom [1, 0, ‘edge’_2]
findatom [2, 0, ‘edge’_2]
…
```

findatom：
アトムリストの先頭からアトムを参照.
マッチング失敗したら次のアトムを参照する
バックトラックを繰り返す.

アトムリスト：
アトム名ごとに
アトムを保持するリスト

edge 探索中の edge(X,Y) を示すアトム　edge 探索中の edge(Y2,Z) を示すアトム　edge 両方が重なっているとき





書き換えの様子と冗長なマッチング

プログラム

```
edge(1,2),edge(2,3),edge(3,4).
edge(X,Y),edge(Y2,Z)\:-
Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

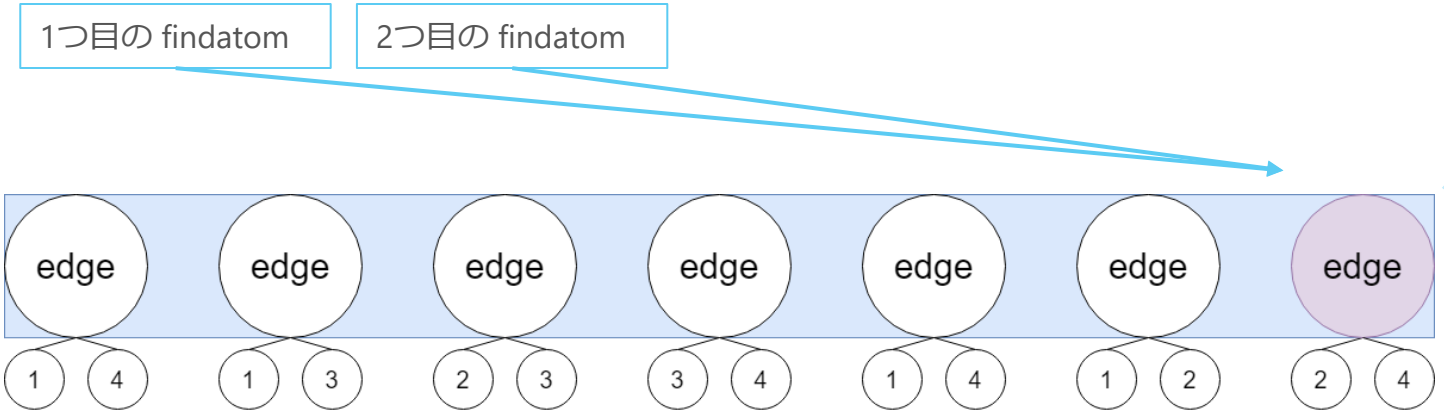
中間命令列

```
…
spec [1, 29]
findatom [1, 0, ‘edge’_2]
findatom [2, 0, ‘edge’_2]
…
```

findatom：
アトムリストの先頭からアトムを参照.
マッチング失敗したら次のアトムを参照する
バックトラックを繰り返す.

アトムリスト：
アトム名ごとに
アトムを保持するリスト

edge 探索中の edge(X,Y) を示すアトム　edge 探索中の edge(Y2,Z) を示すアトム　edge 両方が重なっているとき



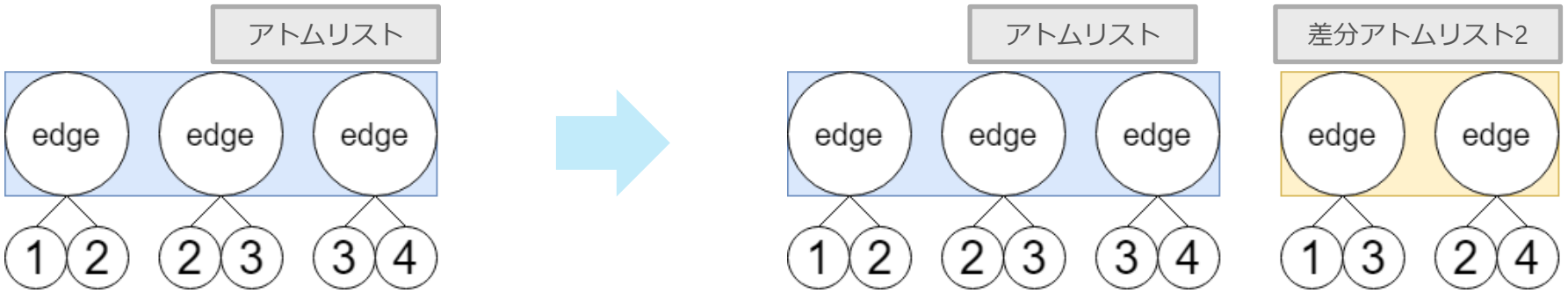
これ以上
マッチング成功
しないため
ルール失敗

貢献：差分アトムリストを用いたマッチング手法

- **差分アトムリスト**：本手法のキーアイデア．新たに生成されるアトムを保持しておくアトムリスト．データ構造はアトムリストと同じ．
- さらなるマッチングに使用する差分アトムリスト1と，生成したアトムを追加するための差分アトムリスト2がある．

ルール例

```
edge(X,Y),edge(Y2,Z)\:- Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

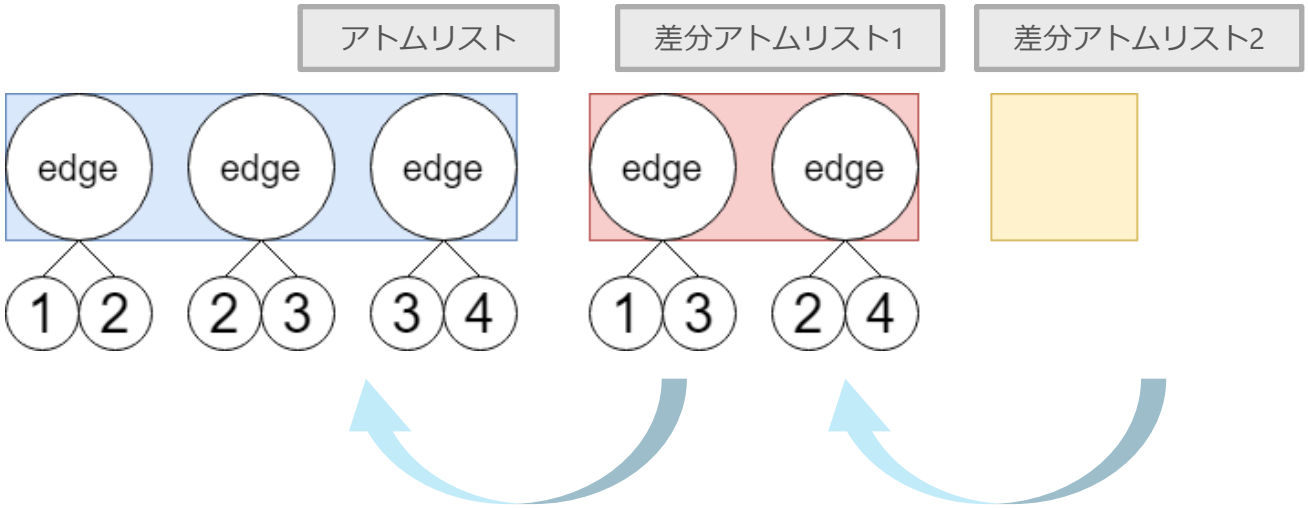


1. マッチング成功で終了せず，バックトラックによって全パターンマッチングを試す
2. 生成されるアトムは差分アトムリスト2へ追加

貢献：差分アトムリストを用いたマッチング手法

ルール例

```
edge(X,Y),edge(Y2,Z)\:- Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

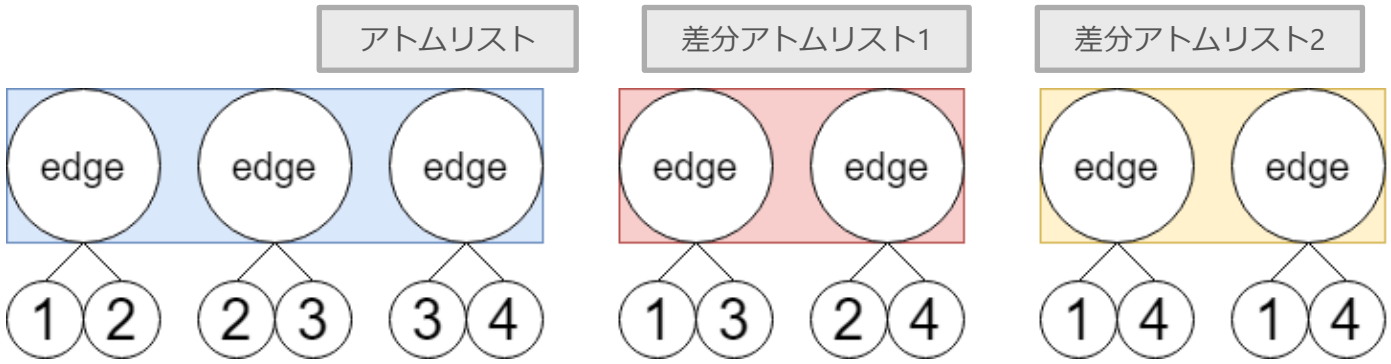


3. 差分アトムリスト1のアトムをアトムリストの最後尾に移動（最初は空なので実質的には何もしない）
4. 差分アトムリスト2のアトムを差分アトムリスト1の最後尾に移動
5. 差分アトムリスト1が空ならルール失敗（アトムが1つも生成されていない）

貢献：差分アトムリストを用いたマッチング手法

ルール例

```
edge(X,Y),edge(Y2,Z)\:- Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```



edge(X,Y) に対応する findatom が参照するアトムリストとedge(Y2,Z) に対応する findatom が参照するアトムリストの組を(アトムリスト,差分アトムリスト1)のように表す.

6. (アトムリスト，差分アトムリスト1)，（差分アトムリスト1，アトムリスト），（差分アトムリスト1，差分アトムリスト1）について全パターンマッチングを試す

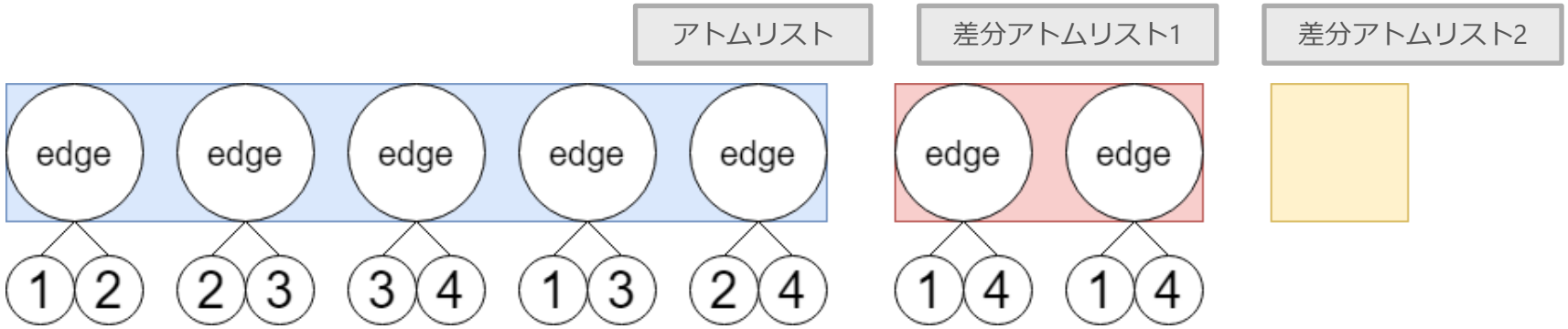
7. 生成されるアトムは差分アトムリスト2に追加する

このマッチング手法により，すべてのアトムの組について
ちょうど1回マッチングを試すことができる

貢献：差分アトムリストを用いたマッチング手法

ルール例

```
edge(X,Y),edge(Y2,Z)\:- Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```



8. 差分アトムリスト1のアトムをアトムリストの最後尾に移動

9. 差分アトムリスト2のアトムを差分アトムリスト1の最後尾に移動

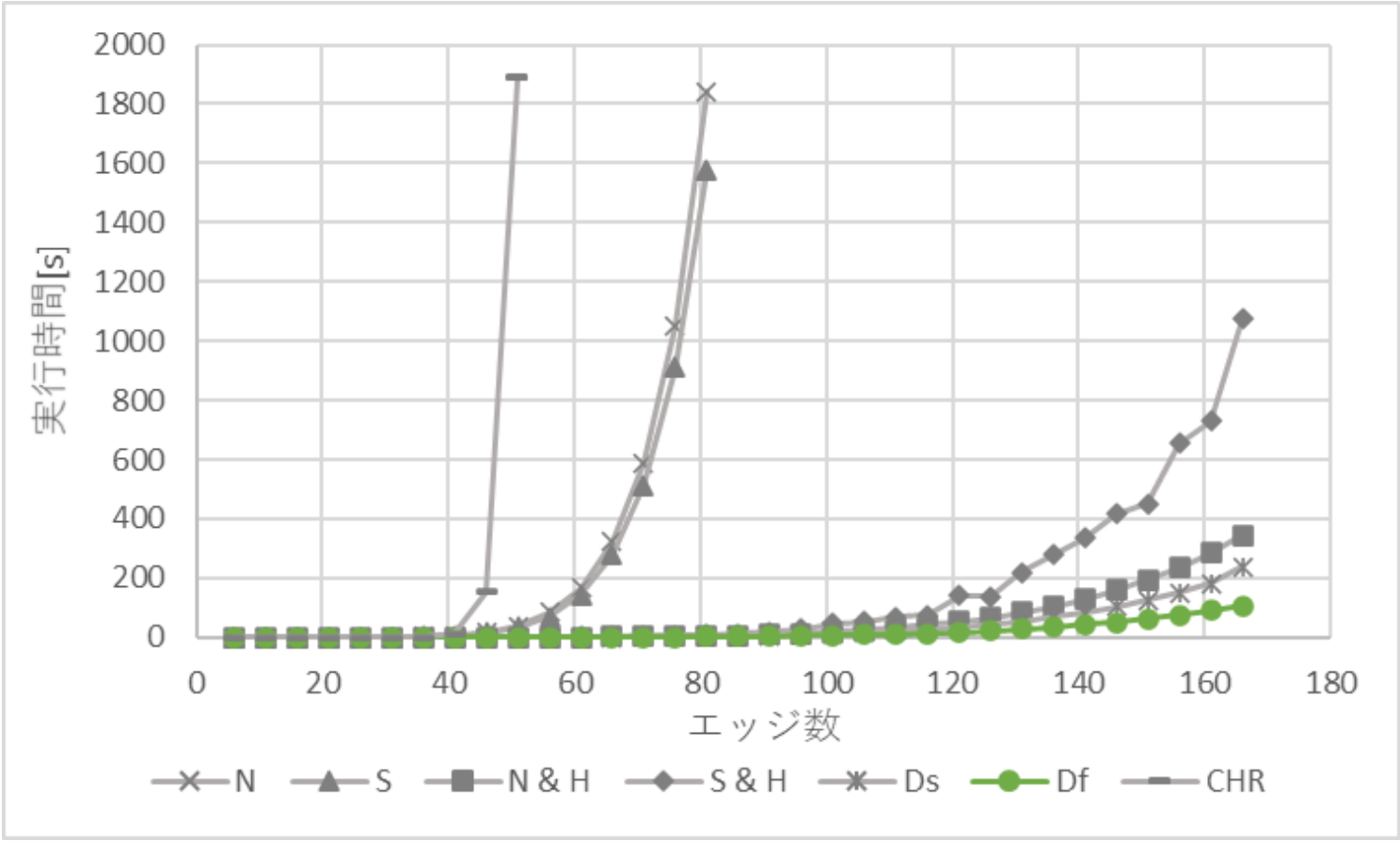
10. 差分アトムリスト1が空ならルール失敗で終了
    空でないなら 6 に戻る

プログラムは，すべてのルールが失敗したら終了する．
本手法ではすべてのアトムの組にマッチングを試していることが保証されているので，
このプログラムでは**ルール失敗で終了しても良い**
（実験では、ルール成功で終了する場合についても測定した）

評価実験

```
edge(X,Y),edge(Y2,Z)\:- Y=:=Y2,uniq(X,Y,Z)|edge(X,Z).
```

提案手法は,
この例題において
現存する手法の中で
最も速い履歴管理の
約3分の1の実行時間



N(normal) : 通常のコンパイルと実行,　S(swaplink) : --use-swaplink をつけてコンパイルし実行,
H(history) : 履歴管理を適用,　Ds(diffatom succeed) : 提案手法(ルール成功で終了), Df(diffatom fail) : 提案手法, CHR[5][6] : 関連言語

[5] Thom Frühwirth. Constraint Handling Rules. Cambridge University Press. 2009.
[6] Thom Frühwirth, Frank Raiser. Constraint Handling Rules: Compilation, Execution, and Analysis. 2011.

まとめと今後の課題

- まとめ
  - 差分アトムリストを用いたマッチング手法により，閉包計算を行うプログラムの冗長なマッチングをすべて削減した
- 今後の課題
  - 例題の拡充
  - コンパイラの実装

参考文献

[1] 上田和紀，加藤紀夫．言語モデル LMNtal．
コンピュータソフトウェア，Vol. 21，No. 2，pp. 126-142, 2004.

[2] 石川力，堀 泰祐，村山 敬，岡部 亮，上田 和紀.
軽量な LMNtal 実行時処理系 SLIM の設計と実装.
情報処理学会第70回全国大会講演論文集. 153-154. 2008.

[3] 小川誠司. LMNtal 実行時処理系 SLIM への uniq 制約の導入. 卒業論文. 2009.

[4] 中田昌輝.
グラフ書き換え言語 LMNtal における非連結サブパターンマッチング高速化手法.
卒業論文. 2021

[5] Thom Frühwirth. Constraint Handling Rules. Cambridge University Press. 2009.

[6] Thom Frühwirth, Frank Raiser.
Constraint Handling Rules: Compilation, Execution, and Analysis.
Books On Demand GmbH2011.

補足：実験環境

OS	Ubuntu18.04
CPU	AMD Ryzen Threadripper 2990WX 32-Core Processor
Memory	64Gbyte
LMNtal Compiler	version 1.51
LMNtal コンパイルオプション	`--slimcode -O3` (`--use-swaplink`)
SLIM	version 2.5.0
SLIM オプション	`-p2`
SWI-Prolog	version 7.6.4 for amd64

補足：初期状態の生成プログラム

- 引数の num にエッジ数を与える
- 初期状態を標準出力

```
void test(int num){
    int i=1, j=2, k=0;
    int count=1;
    printf("edge(%d,%d)",i,j);
    i++; j++;
    while(count < num){
        if(j < i+6){
            count++;

printf(",edge(%d,%d)",i,j);
            j++;
        }else{
            i=j-1;
        }
    }
}
```

補足：CHRで記述したプログラム

- 時間計測のプログラム
- CHR のルールは
  最後の行のもの
- edge(X,Y) 等を chr 制約と呼び
  同じ制約について高々1回
  右辺の制約を追加する

```
:-module(test, [main/0]).
:-use_module(library(chr)).
:-chr_constraint edge/2.

main:-statistics(runtime, [T1|_]),
 VAR ,
 statistics(runtime,[T2|_]),
 Time is T2 - T1,
write(Time),nl,write(edge).

edge(X,Y), edge(Y,Z) ==> edge(X,Z).
```

補足：例題のバックトラック計測結果（履歴管理と提案手法）

エッジ数	N&H のバックトラック数	Ds のバックトラック数	Df のバックトラック数	N&H / Ds	N&H / Df
6	383	275	143	1.39	2.68
11	3009	2036	1044	1.48	2.88
16	13441	8997	4575	1.49	2.94
21	44877	29767	15005	1.51	2.99
26	123375	81753	41151	1.51	3.00
31	294746	194838	97806	1.51	3.01
36	632327	417823	209431	1.51	3.02
41	1246110	824388	412866	1.51	3.02
46	2299600	1521580	762067	1.51	3.02
51	4013030	2655110	1329160	1.51	3.02
56	6684900	4423300	2213620	1.51	3.02
61	10692700	7084830	3544790	1.51	3.02
66	16565000	10970900	5488260	1.51	3.02
71	24899000	16497600	8252120	1.51	3.02
76	36483900	24179800	12093700	1.51	3.02
81	52264600	34644900	17326900	1.51	3.02
86	73384200	48654300	24334600	1.51	3.02
91	101162000	67100000	33558500	1.51	3.01
96	137271000	91049900	45534600	1.51	3.01
101	183557000	121749000	60885200	1.51	3.01
106	242157000	160641000	80332900	1.51	3.01
111	315569000	209394000	104711000	1.51	3.01
116	406760000	269916000	134973000	1.51	3.01
121	518830000	344381000	172207000	1.51	3.01
126	655650000	435250000	217643000	1.51	3.01
131	821313000	545299000	272670000	1.51	3.01
136	1020530000	677642000	338843000	1.51	3.01
141	1258450000	835757000	417902000	1.51	3.01
146	1540830000	1023520000	511783000	1.51	3.01
151	1874490000	1245210000	622633000	1.51	3.01

N&H : 通常のコンパイル+履歴管理, Ds : 提案手法（ルール成功で終了）, Df : 提案手法