モデル生成型定理証明系MGTPの要素技術

長谷川　隆三
藤田　博
越村　三幸

九州大学大学院システム情報科学研究院

MGTP

発端と経緯

- SATCHMOとPTTP
- ICOT拡大所議（1990.1）
- プルーバプロジェクト立上げ（1990.3）
- 淵コーディングの衝撃
  - インタプリタの開発
- Overbeekの来訪
  - Lukasievicz問題のチャレンジ
- インタプリタの改良
  - SOS, RAMS, MERC

SATCHMOのモデル生成法

- **モデル拡張** $A_1 \wedge \cdots \wedge A_n \rightarrow B_1 \vee \cdots \vee B_m$

$$\frac{M \models A_1 \wedge \cdots \wedge A_n \qquad M \not\models B_1 \vee \cdots \vee B_m}{M \cup \{B_1\} \mid \cdots \mid M \cup \{B_m\}}$$

- **モデル棄却** $A_1 \wedge \cdots \wedge A_n \rightarrow false$

$$\frac{M \models A_1 \wedge \cdots \wedge A_n}{\bot}$$

問題S1と証明木

正節

C1: true → p(a) ∨ q(b).

混合節

C2: q(X) → s(f(X)).

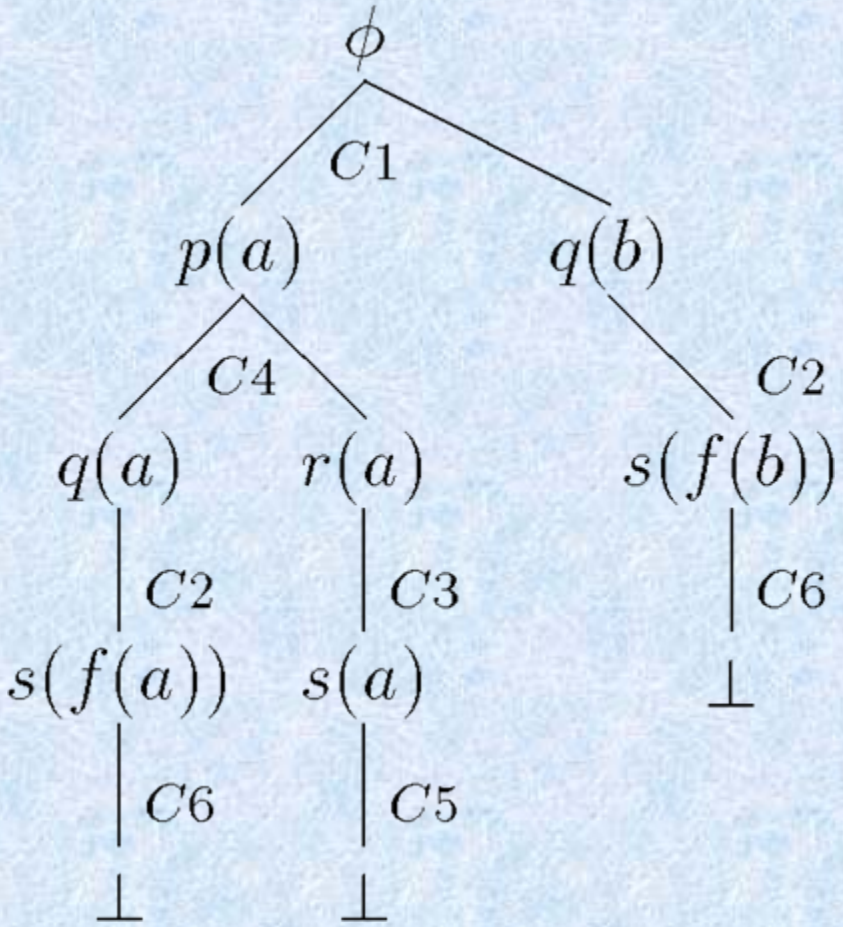
C3: r(X) → s(X).

C4: p(X) → q(X) ∨ r(X).

負節

C5: p(X) ∧ s(X) → false.

C6: q(X) ∧ s(Y) → false.



MGTP

Prolog版SATCHMO

```
satisfiable :- is_violated(C), !, satisfy(C),  satisfiable.
satisfiable.

is_violated(C) :- (A--->C), call(A), not(C).

satisfy(C) :- component(X,C), asserta(X),
              on_backtracking(retract(X)),
              not(false).

component(X,(Y;Z)) :- !, (X=Y; component(X,Z)).
component(X,X).

on_backtracking(X).
on_backtracking(X) :- call(X), !, fail.
```


MGTP

SATCHMOの特徴

- 対象変数とProlog変数を同一視
- (A--->C)の呼出しで新変数を得る
- メタ述語 assert/retract, call, not, ! の利用
- モデル候補要素はassert/retractされる
- call(A) で連言照合、not(C) で包摂検査

MGTP

KL1の困難

- メタ述語が使えない
  （…けど、対象変数とKL1変数を同一視したい）
  ⇒ 質問コンパイル法
  cf. ソフトウェア科学会全国大会1991，
  古川，藤田（正），藤田（博），長谷川，淵

- OR並列に伴う環境複製問題
  ⇒ モデル生成法の場合、環境（モデル候補）は基底なので問題は深刻でない
  OR並列のAND並列での代用
  cf. KL1プログラミングワークショップ1990，淵

モデル要素とリテラルとの照合

	Prolog	KL1
モデル要素 p(a)	プログラム(fact)節 として 待ち受ける	ゴール引数 として 持ち回される
リテラル p(X)	ゴール引数 として 持ち回される	プログラム節で 待ち受ける

MGTP

SATCHMOのKL1コーディング（問題特化版）by 淵

```
false(M,Z) :- true | c11(M,M,[],Z1), c21(M,M,Z1,Z2),
                     ... , c61(M,M,Z5,Z).

% C4: p(X) -> q(X); r(X).

c41(M1,M,[],Z) :- true |
  ( M1=[] -> Z=[];          % p(X) not in M: C4 satisfied
    M1=[p(X)|M2] -> c42(X,M,M,Z2), c41(M2,M,Z2,Z);
    otherwise; M1=[_|M2] -> c41(M2,M,Z1,Z) ).
otherwise.
c41(_,_,Z1,Z) :- true | Z=Z1.
```


MGTP

```
% C4: ... -> q(X); ...

c42(X,M1,M,Z) :- true |
  ( M1=[] -> c43(X,M,M,Z);  % q(X) not in M, try r(X)
    M1=[q(X)|_] -> Z=[];      % q(X) in M: C4 satisfied
    otherwise; M1=[_|M2] -> c42(X,M2,M,Z)   ).

% C4: ... -> ...; r(X)

c43(X,M1,M,Z) :- true |
  ( M1=[] -> c44(X,M,Z);    % r(X) not in M, extend M
    M1=[r(X)|_] -> Z=[];     % r(X) in M: C4 satisfied
    otherwise; M1=[_|M2] -> c43(X,M2,M,Z) ).

c44(X,M,Z) :- true | false([q(X)|M],Z1), false([r(X)|M],Z2),
                     both(Z1,Z2,Z).  % merge two results
```

```
**************************************

 FFFFF  U   U   CCC   H   H   III
 F      U   U  C   C  H   H    I
 F      U   U  C      H   H    I
 FFF    U   U  C      HHHHH    I
 F      U   U  C      H   H    I
 F      U   U  C   C  H   H    I
 F       UUU    CCC   H   H   III

**************************************

  File_name: icpsi152::>sys>spool>000          Identifier: >sys>net>spool>000
  System_name: SIMPOS_v5.22    Program_name: lbp_like_printer
  Date: 29-Mar-90  14:44:42        User_name: fuchi
```

```
:- module mg_s1.
:- public do/1.
%%  problem s1
do(N):-N>0|false([],T),N1:=N-1,do(N1).  otherwise.   do(_).
false(F,T):-true|c11(F,F,T),c21(F,F,T),c31(F,F,T),
                    c41(F,F,T),c51(F,F,T),c61(F,F,T).
%%      true=> (p(a);q(b));
c11(F1,F,t).   alternatively.
c11(F1,F,T):-true|(F1=[]   ->c12(F,F,T);
                   F1=[p(a)|F2] ->true ;
 otherwise;        F1=[Z|F2]    ->c11(F2,F,T)).
c12(F1,F,t).   alternatively.
c12(F1,F,T):-true|(F1=[]   ->c13(F,T);
                   F1=[q(b)|F2] ->true ;
 otherwise;        F1=[Z|F2]    ->c12(F2,F,T)).
c13(F,T)    :-true|false([p(a)|F],T1),false([q(b)|F],T2),
                   both(T1,T2,T).
both(T1,T2,T):-T1=t,T2=t          |T=t.
otherwise.   both(T1,T2,T):-true|T=abort.
%%      p(X)=> (q(X);r(X));
c21(F1,F,t).   alternatively.
c21(F1,F,T):-true|(F1=[]   ->true ;
                   F1=[p(X)|F2] ->c22(X,F,F,T),c21(F2,F,T);
 otherwise;        F1=[Z|F2]    ->c21(F2,F,T)).
c22(X,F1,F,t).   alternatively.
c22(X,F1,F,T):-true|(F1=[]->c23(X,F,F,T);
```

```
c21(F1,F,T):-true¦(F1=[]    ->true ;
                F1=[p(X)¦F2] ->c22(X,F,F,T),c21(F2,F,T);
 otherwise;     F1=[Z¦F2]    ->c21(F2,F,T)).
c22(X,F1,F,t).   alternatively.
c22(X,F1,F,T):-true¦(F1=[]->c23(X,F,F,T);
                F1=[q(X)¦F2] ->true ;
 otherwise;     F1=[Z¦F2]    ->c22(X,F2,F,T)).
c23(X,F1,F,t).   alternatively.
c23(X,F1,F,T):-true¦(F1=[]->c24(X,F,T);
                F1=[r(X)¦F2] ->true ;
 otherwise;     F1=[Z¦F2]    ->c23(X,F2,F,T)).
c24(X,F,T)  :-true¦false([q(X)¦F],T1),false([r(X)¦F],T2),
                both(T1,T2,T).
%%      q(X) => s(f(X));
c31(F1,F,t).   alternatively.
c31(F1,F,T):-true¦(F1=[]    ->true ;
                F1=[q(X)¦F2] ->c32(X,F,F,T),c31(F2,F,T);
 otherwise;     F1=[Z¦F2]    ->c31(F2,F,T)).
c32(X,F1,F,t).   alternatively.
c32(X,F1,F,T):-true¦(F1=[]->c33(X,F,T);
                F1=[s(f(X))¦F2]->true ;
 otherwise;     F1=[Z¦F2]    ->c32(X,F2,F,T)).
c33(X,F,T)   :-true          ¦false([s(f(X))¦F],T).
%%      r(X) => s(X);
c41(F1,F,t).   alternatively.
c41(F1,F,T):-true¦(F1=[]    ->true ;
                F1=[r(X)¦F2] ->c42(X,F,F,T),c41(F2,F,T);
 otherwise;     F1=[Z¦F2]    ->c41(F2,F,T)).
```

```
c33(X, F, T)    :-true                  !false([s(f(X))!F], T).
%%      r(X) => s(X);
c41(F1, F, t).   alternatively.
c41(F1, F, T):-true!(F1=[]    ->true ;
                     F1=[r(X)!F2]  ->c42(X, F, F, T), c41(F2, F, T);
 otherwise;          F1=[Z!F2]     ->c41(F2, F, T)).
c42(X, F1, F, t).   alternatively.
c42(X, F1, F, T):-true!(F1=[]->c43(X, F, T);
                     F1=[s(X)!F2]  ->true ;
 otherwise;          F1=[Z!F2]     ->c42(X, F2, F, T)).
c43(X, F, T)    :-true!false([s(X)!F], T).
%%      q(X), s(Y) => false;
c51(F1, F, t).   alternatively.
c51(F1, F, T):-true!(F1=[]    ->true ;
                     F1=[q(X)!F2]  ->c52(F, F, T), c51(F2, F, T);
 otherwise;          F1=[Z!F2]     ->c51(F2, F, T)).
c52(F1, F, t).   alternatively.
c52(F1, F, T):-true!(F1=[]    ->true ;
                     F1=[s(Y)!F2]  ->T=t;
 otherwise;          F1=[Z!F2]     ->c52(F2, F, T)).
%%      p(X), s(X) => false;
c61(F1, F, t).   alternatively.
```

ＫＬ１プログラミング雑感－－ｐｒｏｖｅｒの並列化の体験より－－

１９９０．４．２３ 渕

１．いきさつ

ＫＬ１はこのプロジェクトの基本言語である。考え方については聞いているし了承しているものであるが、実作業については皆さんにお委せしてきた。今でもそれで良いと思っている。

ところで、昨年末、ＰＳＩをやっと手元において貰うことになった。しばらくＳＩＭＰＯＳ／ＥＳＰほかで遊んでいたが、２／１６（というのは実は私の誕生日）を期してＫＬ１を試みることにした。基本的には、老化度チェックの頭の体操のつもり。

題材としては、ｐｒｏｖｅｒを選んだ。正月の拡大所議で古川次長が紹介したＳａｔｃｈｍｏが大変コンパクトなｐｒｏｖｅｒ（の一種）で、私程度に手ごろかと思ったことが一つ。ｐｒｏｖｅｒ自体についての議論はここでは省くが、本来ＦＧＣＳの基本技術（の一つ）であるはずのもので、知識処理的テーマ（いわゆる山側テーマ）に直接、間接に関わっている。

ＫＬ１利用については、ＰＩＭＯＳなどシステム・プログラミングでは着実に進展しているようであるが、山側では（頑張ってくれている人達ももちろんいるが）私が期待したほどではないようである。ウワサでは、ＫＬ１は使い難いらしい。そういえば、昨年正月の拡大所議で、ＫＬ１のユニフィケーションについての議論（不用論？）が少しあった。議論を煮詰めて欲しいと要望してあったのだが（してくれたのかもしれないが）、その後

はどではないようである。ウワサでは、ＫＬ１は使い難いらしい。そうすれば、昨年並みの拡大所議で、ＫＬ１のユニフィケーションについての議論（不用論？）が少しあった。議論を煮詰めて欲しいと要望してあったのだが（してくれたのかもしれないが）、その後が判然としない。ウワサは本当なのか。自分でも実感を若干掴んでおきたいという気もあった。

２．やってみたこと（その１）

やったことの実体は、Ｓａｔｃｈｍｏ的動作のＫＬ１によるサンプル・コーディングである。（ここで本当はＳａｔｃｈｍｏの説明が必要だが、省略）　始める前の見通しとしては、

（１）ｏｒ並列的動作は、ａｎｄ並列で代用できる。（あまり問題はないはず。但し（その２）を参照のこと）

（２）ユニフィケーションにはなんらかの工夫を要する。それを丸々再コーディングするのでは遅くなりすぎるだろう。これは皆さん体験済み。出来ればＫｌ１の機能を流用したい。コンパイル方式の方が可能性がありそう。

ということで、「問題」をＫＬ１にコンパイルしたとして、想定すべきオブジェクト・コードを工夫することにした。

ところが幸いなことに、Ｓａｔｃｈｍｏには　ｒａｎｇｅ－ｒｅｓｔｒｉｃｔｅｄとい

う前提がある。その中で生成されるモデルは基底項だけで構成される。

そこで、実際に必要なのは、ユニフィケーションというより、マッチングである。また、コンパイル方式だから、新変数の導入が楽である。これで変数の依存関係を後送りに出来、また、ｏｒ並列の実現も簡単化できる。ということで第一の関門は抜けることが出来た。

速度は、現段階では、例えば、原論文　９．５秒（Ｓｅｔｈｅｏでは　４．５秒）の問題（それまでは分オーダ）が　２３０ｍｓほど。最適化をも少し入れると　１００ｍｓをきる。

コンパイラはまだ作っていない。本当の並列実行もこれからである。インタプリタは無理かと思っていたが、長谷川／藤田が巧みに実現した。

（３）やったこと（その２）

ところで、長谷川／藤田のアイデアに近いものとして、ＳｈａｐｉｒｏのＦＣＰによるｏｒ並列全解探索ｐｒｏｌｏｇインタプリタがある。少し制限がつくようだが、ＫＬ１にも移植できる。実効はともかく、ｏｒ並列＋ａｎｄ並列の実現もできる。ということでＫＬ１版を作った。

そのアイデアの基本は、ｔｒａｉｌを保存して、ｏｒ分岐のとき変数のコピーを作りなおすというものである。

（４）雑感

１．ＫＬ１は面白い。

（４）雑感

１．ＫＬ１は面白い。

並列実行はまだ実際には試みていないが、疑似でも面白い。横型探索の本格的言語の実体験は初めてなので大いに楽しめた。（皆さん、新しい玩具は楽しくないですか）

２．ｐｒｏｖｅｒの実現から見て。

横型探索（並列）は、ｆａｉｒｎｅｓｓを（ある程度）保証しているのでｐｒｏｖｅｒには有利である。

ＫＬ１変数（ユニフィケーション）も結構使える。使いこなす工夫を色々して、その上で拡張や改良の提案が出てくるとよいと思う。

これはもっと考えなければならないが、変数自体を扱う機能がやはり欲しい気がする。それは、すごく軽い（簡単だが有効な）ｆｒｅｅｚｅ－ｍｅｌｌのようなものだろうか。

３．もっとマクロな表現はないか。

各種のブルーバ、問題解決器を書くための上位言語（マクロ表現）はやはり欲しい気がする。ＫＬ１から素直に上昇するやり方があると良いと思うのだが。

PSATインタプリタ by 藤田・長谷川

```
false(M,Z) :- true | ante(1,[],M,M,sat,Z1), ante(2,[],M,M,Z1,Z2),
                     ... ,  ante(6,[],M,M,Z5,Z).

ante(Cn,A,M1,M,sat,Z) :- true |
  ( M1=[E|M2] -> c(Cn,E,A,S),  % 問題節の呼出し
     ( S=fail -> ante(Cn,A,M2,M,sat,Z);
       S=cont -> ante(Cn,[E|A],M,M,sat,Z1), ante(Cn,A,M2,M,Z1,Z);
       otherwise;
       true -> cnsq(S,S,M,Z1), ante(Cn,A,M2,M,Z1,Z) );
    M1=[] -> Z=sat ).
otherwise.
ante(_,_,_,_,Z1,Z) :- true | Z=Z1.

cnsq([E|D1] ,D,M,Z) :- true | /* Dの包摂検査 */.   % if E in M, Cl satisfied
cnsq([],D,M,Z) :- true | extend(D,M,Z).

extend([E|D] ,M,Z) :- true |
  false([E|M],Z1), extend(D,M,Z2), both(Z1,Z2,Z);
extend([],M,Z) :- true | Z=closed.
```


MGTP

S1問題のKL1表現

```
c(1,true,[],S):-true|S=[p(a),q(b)]. % true->p(a);p(b)
c(2,q(X),[],S):-true|S=[s(f(X))].   % q(X)->s(f(X))
c(3,r(X),[],S):-true|S=[s(X)].      % r(X)->s(X)
c(4,p(X),[],S):-true|S=[q(X),r(X)]. % p(X)->q(X);r(X)
c(5,p(X),[],S):-true|S=cont.        % p(X),s(X)->false
c(5,s(X),[p(X)],S):-true|S=[].
c(6,q(X),[],S):-true|S=cont.        % q(X),s(Y)->false
c(6,s(Y),[q(X)],S):-true|S=[].
otherwise.
c(_,_,_,S):-true|S=fail.
```


MGTP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%         PSAT:  Parallel SATCHMO Interpreter                                  %
%                version 5.5d   Mar. 2, '90  Am. 10:00                         %
%                                                                             %
%                by H. Fujita and R. Hasegawa                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

:- module satchmo.
:- public do/1.

do(T) :- true | satchmo_problem:mi(M,IX), false(M,IX,T).

false(F,IX,T) :- vector_element(IX,0,Cn) | mk_cc(Cn,IX,F,[],T).

mk_cc(Cn,IX,F,T,T2) :- true |
    ( Cn > 0  ->  Cn1 := Cn - 1,
                  do_c(Cn,IX,F,T,T1),
                  mk_cc(Cn1,IX,F,T1,T2) ;
      Cn = 0  ->  T2=T ).

do_c(Cn,IX,F,T1,T) :- vector_element(IX,Cn,Ln) |
    ( Ln > 0  ->  do_c1(Cn,Ln,IX,_,F,F,T1,T) ;
      Ln = 0  ->  satchmo_problem:c(Cn,_,_,_,_,Facts0),
                  do_c0(Facts0,Facts0,IX,F,T1,T) ).

do_c0(Facts,Facts0,IX,F,[],T) :- true |
    ( Facts=[Fact|RF]  ->  do_c01(Fact,RF,Facts0,IX,F,F,T) ;
```

```
                    do_c0(Facts0,Facts0,IX,F,T1,T) ).

do_c0(Facts,Facts0,IX,F,[],T) :- true |
    ( Facts=[Fact|RF]  ->  do_c01(Fact,RF,Facts0,IX,F,F,T) ;
      Facts=[]         ->  do_c02(Facts0,IX,F,T) ).
otherwise.
do_c0(_,_,_,_,T1,T) :- true | T=T1.

do_c01(Fact,RF,Facts0,IX,F1,F,T) :- true |
    ( F1=[Fact|F2]     ->  T=[] ;
      F1=[]            ->  do_c0(RF,Facts0,IX,F,[],T) ;
      otherwise ;
      F1=[_|F2]        ->  do_c01(Fact,RF,Facts0,IX,F2,F,T) ).

do_c02(Facts,IX,F,T) :- true |
    ( Facts=[Fact|RF]  ->  false([Fact|F],IX,T1),
                           do_c02(RF,IX,F,T2),
                           both(T1,T2,T) ;
      Facts=[]         ->  T=t ).

both(t,t,T) :- true | T=t.
otherwise.
both(_,_,T) :- true | T=abort.

do_c1(Cn,Ln,IX,Vi,F1,F,[],T) :- true |
    ( F1=[P|F2]  ->  satchmo_problem:c(Cn,Ln,P,Vi,Vi1,Ans),
                     do_c11(Ans,Cn,Ln,IX,Vi,Vi1,F2,F,T) ;
      F1=[]      ->  T=[] ).
otherwise.
do_c1(_,_,_,_,_,_,T1,T) :- true | T=T1.
```

```
      otherwise ;
      F1=[_|F2]          ->  do_c01(Fact,RF,Facts0,IX,F2,F,T) ).

do_c02(Facts,IX,F,T) :- true |
    ( Facts=[Fact|RF]  ->  false([Fact|F],IX,T1),
                           do_c02(RF,IX,F,T2),
                           both(T1,T2,T) ;
      Facts=[]         ->  T=t ).

both(t,t,T) :- true | T=t.
otherwise.
both(_,_,T) :- true | T=abort.

do_c1(Cn,Ln,IX,Vi,F1,F,[],T) :- true |
    ( F1=[P|F2]  ->  satchmo_problem:c(Cn,Ln,P,Vi,Vi1,Ans),
                     do_c11(Ans,Cn,Ln,IX,Vi,Vi1,F2,F,T) ;
      F1=[]      ->  T=[] ).
otherwise.
do_c1(_,_,_,_,_,_,T1,T) :- true | T=T1.

do_c11(Ans,Cn,Ln,IX,Vi,Vi1,F2,F,T) :- true |
    ( Ans=f      ->  do_c1(Cn,Ln,IX,Vi,F2,F,[],T) ;
      Ans=t      ->  Ln1 := Ln-1,
                     do_c1(Cn,Ln1,IX,Vi1,F,F,[],T2),
                     do_c1(Cn,Ln,IX,Vi,F2,F,T2,T) ;
      Ans=[]     ->  T=t ;
      Ans=[_|_]  ->  do_c0(Ans,Ans,IX,F,[],T2),
                     do_c1(Cn,Ln,IX,Vi,F2,F,T2,T) ).
```

```
%
%       S1
%

:- module satchmo_problem.
:- public mi/2, c/6.

mi(M,IX) :- true | M=[], IX={6,1,1,1,0,2,2}.

c(6,2,p(X),  _,V,A) :- true | A=t, V={X}.     %  p(X),
c(6,1,s(X),{X},_,A) :- true | A=[].            %        s(X) ---> false
c(5,2,q(X),  _,_,A) :- true | A=t.             %  q(X),
c(5,1,s(Y),  _,_,A) :- true | A=[].            %        s(Y) ---> false
c(4,_,   _,  _,_,A) :- true | A=[p(a),q(b)].   %  true ---> p(a) ; q(b)
c(3,1,p(X),  _,_,A) :- true | A=[q(X),r(X)].   %  p(X) ---> q(X) ; r(X)
c(2,1,q(X),  _,_,A) :- true | A=[s(g(X))].     %  p(X) ---> s(g(X))
c(1,1,r(X),  _,_,A) :- true | A=[s(X)].        %  r(X) ---> s(X)
otherwise.
c(_,_,_,_,_,A) :- true | A=f.

%
%   mi(M,IX)
%
%       M  : initial model (list of facts)
%       IX : clause information (Number of clauses, Number of literals)
%
%   c(Rule_no,Literal_no,Literal,Vin,Vout,Answer)
%
%       Rule_no    : identifies the rule
%       Literal_no : identifies literal position in an antecedent
```

```
%
%   mi(M,IX)
%
%       M  : initial model (list of facts)
%       IX : clause information (Number of clauses, Number of literals)
%
%   c(Rule_no,Literal_no,Literal,Vin,Vout,Answer)
%
%       Rule_no    : identifies the rule
%       Literal_no : identifies literal position in an antecedent
%       Literal    : literal in an antecedent
%       Vin        : vector of variables propagated from previous litera
%       Vout       : vector of variables to be propagted to the next lit
%       Answer     : f   if matching the specified literal fails
%                    t   if matching the specified literal succeeds
%                    a list of succeedent literals
%                        if the antecedent has been solved
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Performance results of PSAT interpreter on ICPSI265                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  Overhead

?- true.
 13 reductions
 169 msec
 yes.

%  S1

?- satchmo:do(T).
 T = t
 680 reductions
 168 msec
 yes.

%  S2 : Shubert's Steamroller

   satchmo:do(T).
 T = t
 14155 reductions
 167 msec
 yes.

%  S3
```

```
 yes.

%  S1

?- satchmo:do(T).
 T = t
 680 reductions
 168 msec
 yes.

%  S2 : Shubert's Steamroller

   satchmo:do(T).
 T = t
 14155 reductions
 167 msec
 yes.

%  S3

?- satchmo:do(T).
 T = t
 591192 reductions
 6877 msec
 yes.
```

Mar 2, '90

(淵, 石川, 長谷川, 藤田)

① depth First + Breadth First
(スタック + キュー)
{スタック ~~fst~~ true → A ; B ; C
キュー その他の後件.

② X-X . global counter

③ Priority
⊙ Clause Number を利用.

④ Configuration
⊙ MR-cc を変更.

⑤ G — falsify された Model を記憶 ~~(Top)~~ T = C(F)

⑥ satisfiable Model を返す
→ abort(F)

⑦ Subsumption をみる

○ Subsumption を入れる

↳ Memo function (Model に対して)

Most specific generalization?

$g(x) \wedge s(y) \rightarrow false$

$g(b) \wedge s(f(b)) \rightarrow false$ ①

$g(x) \wedge s(f(x)) \rightarrow false$ ②

+ $g(x) \rightarrow false$ ③ をレンマとして追加

integrity constraint を dynamic に 経済的な lemma を作る.

△ ユーザが lemma を作る際のガイダンスを与える

が含まれば

$g(x) \rightarrow$

$\rightarrow s(f x)$

淵コードとPSATインタプリタ

淵コード	PSATインタプリタ
問題節(C)と推論手続き(P)が融合(Pc) コードサイズ大	問題節(C)と推論手続き(P)が分離 コンパクト
Cは簡単なマクロでPcへ	もし、PをCについて部分評価すればPc相当を得られるはず
節解釈のオーバヘッドなし	節解釈のオーバヘッドは小さく 実行時間で3倍弱以下

MGTP

PSATからMGTPへ

- Hardなホーン問題への挑戦

  Condensed detachment問題

  → p(i(i(i(X,Y),Z),i(i(Z,X),i(U,X)))).

  p(X), p(i(X,Y)) → p(Y).

  p(i(i(i(a,b),a),a))) → .

  - 冗長計算の回避
  - Occur check 付きunification
  - Demodulation/paramodulation

MGTPにおけるnew features-1

- 連言照合の冗長性回避

p(X), p(i(X,Y)) → ...

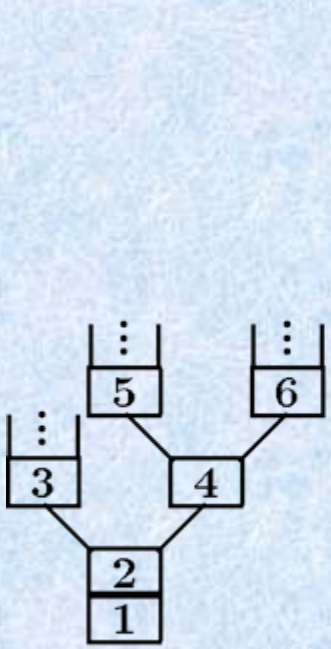
Δ＋M Δ＋M

＝Δ×Δ ＋Δ×M ＋ M×Δ ＋ M×M

冗長！

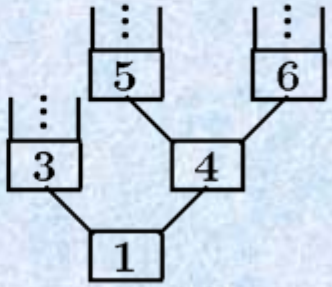
  - Ramified stack方式 (照合結果の記憶)
  - Merc方式(差分計算)

Ramified Stack 方式



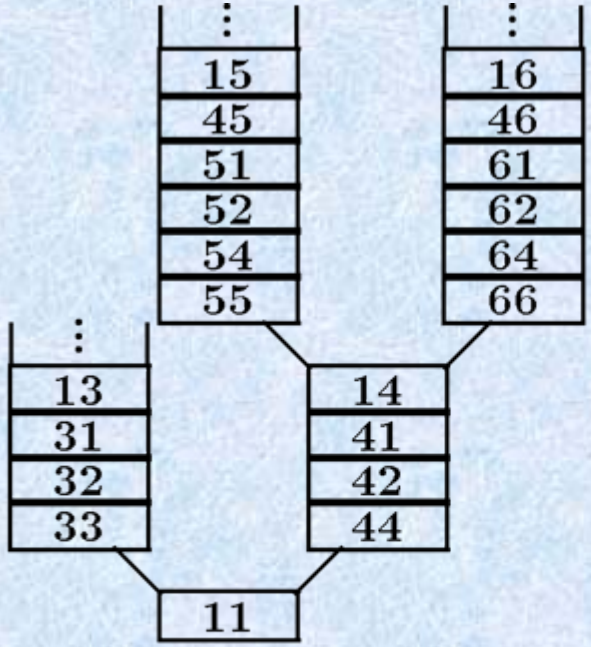
S_0

M



S_1

$L_1,$



S_2

$L_2, \quad L_3 \rightarrow L_4; L_5$

MGTP

Merc 方式

$$A_1 \wedge \quad A_2 \quad \wedge \quad A_3 \quad \wedge \cdots \wedge \quad A_n \quad \rightarrow \cdots$$

$$\begin{array}{llllll}
A_1 \mid & A_2 & \wedge & A_3 & \wedge \cdots \wedge & A_n \quad \rightarrow \cdots \\
A_2 \mid & A_1 & \wedge & A_3 & \wedge \cdots \wedge & A_n \quad \rightarrow \cdots \\
 & & \cdots & & & \\
A_n \mid & A_1 & \wedge & A_2 & \wedge \cdots \wedge & A_{n-1} \quad \rightarrow \cdots
\end{array}$$

$$\begin{array}{cccccc}
\Uparrow & \Uparrow & & \Uparrow & \cdots & \Uparrow \\
\Delta & \Delta + M & & \Delta + M & \cdots & \Delta + M
\end{array}$$

M^nの重複のみ回避

MGTP

MGTPにおけるnew features-2

- 支持集合として働く選言バッファの導入

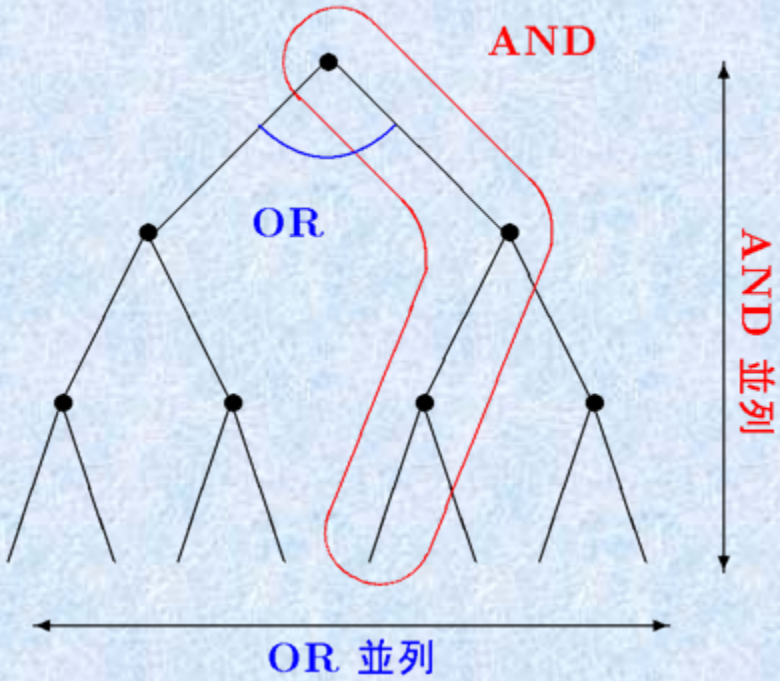
  入力節集合Sを支持集合(T)と補助情報(S-T)に分割し、S-Tどうしの導出を避ける。但し、S-Tは充足可能。
  ここでは、支持集合として全単位節をとる。

  - M: モデル候補　　has-been-given-list
  - D: モデル拡張候補　to-be-given-list

  ⇒ 重複計算の回避,
  　Dの簡約による枝刈り

並列化

- AND並列化　一つの枝の探索を並列に行う.
  - 連言照合と包摂検査が主な対象
  遅延モデル生成:　負節の要求に基づき生成
- OR並列化　各枝の探索を並列に行う.

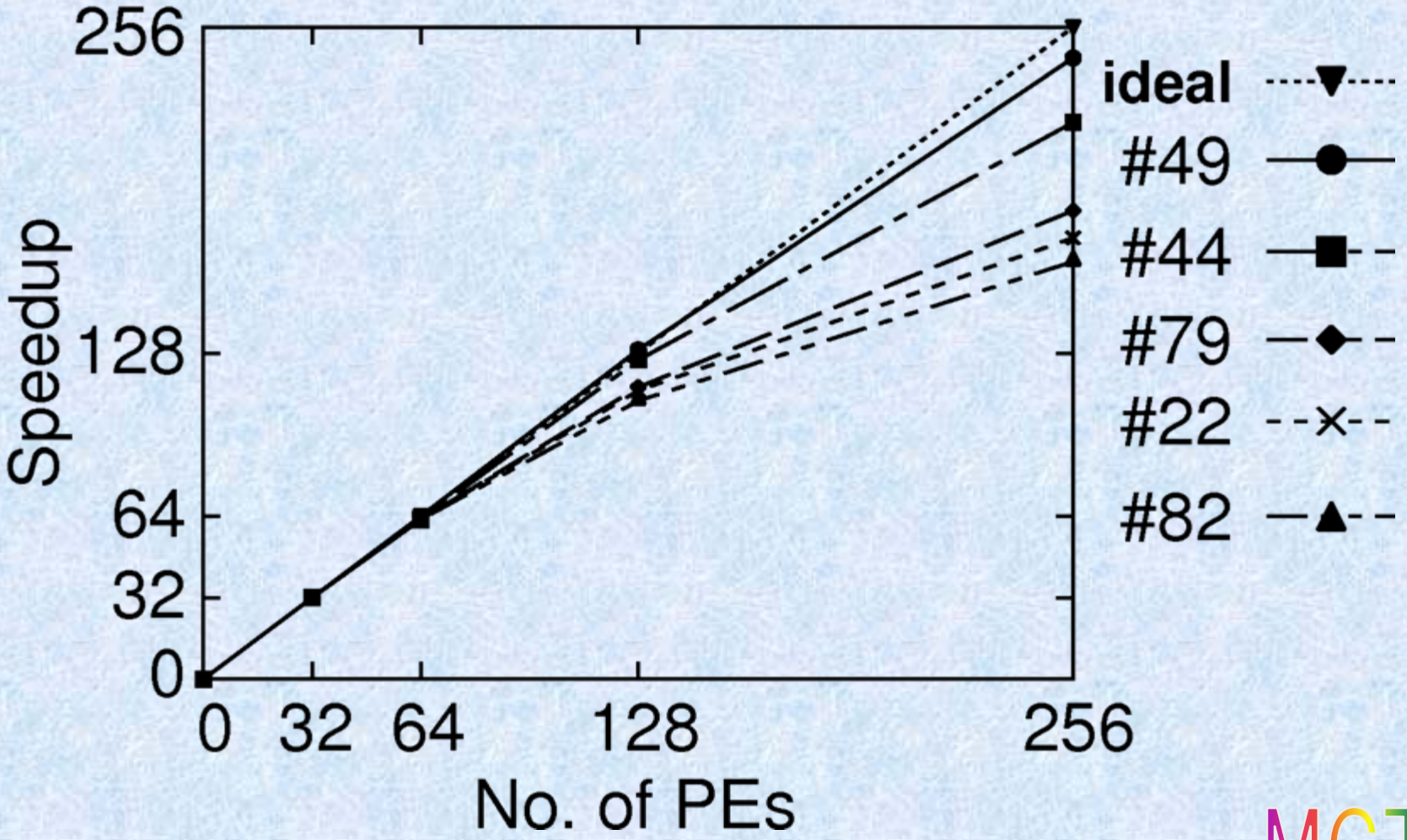


循環割付け方式:分枝を他PEに循環的に割当てる

N-逐次方式:分枝を自PE内ではスタックで、
他PEへ割当てる際はキューで取り出す

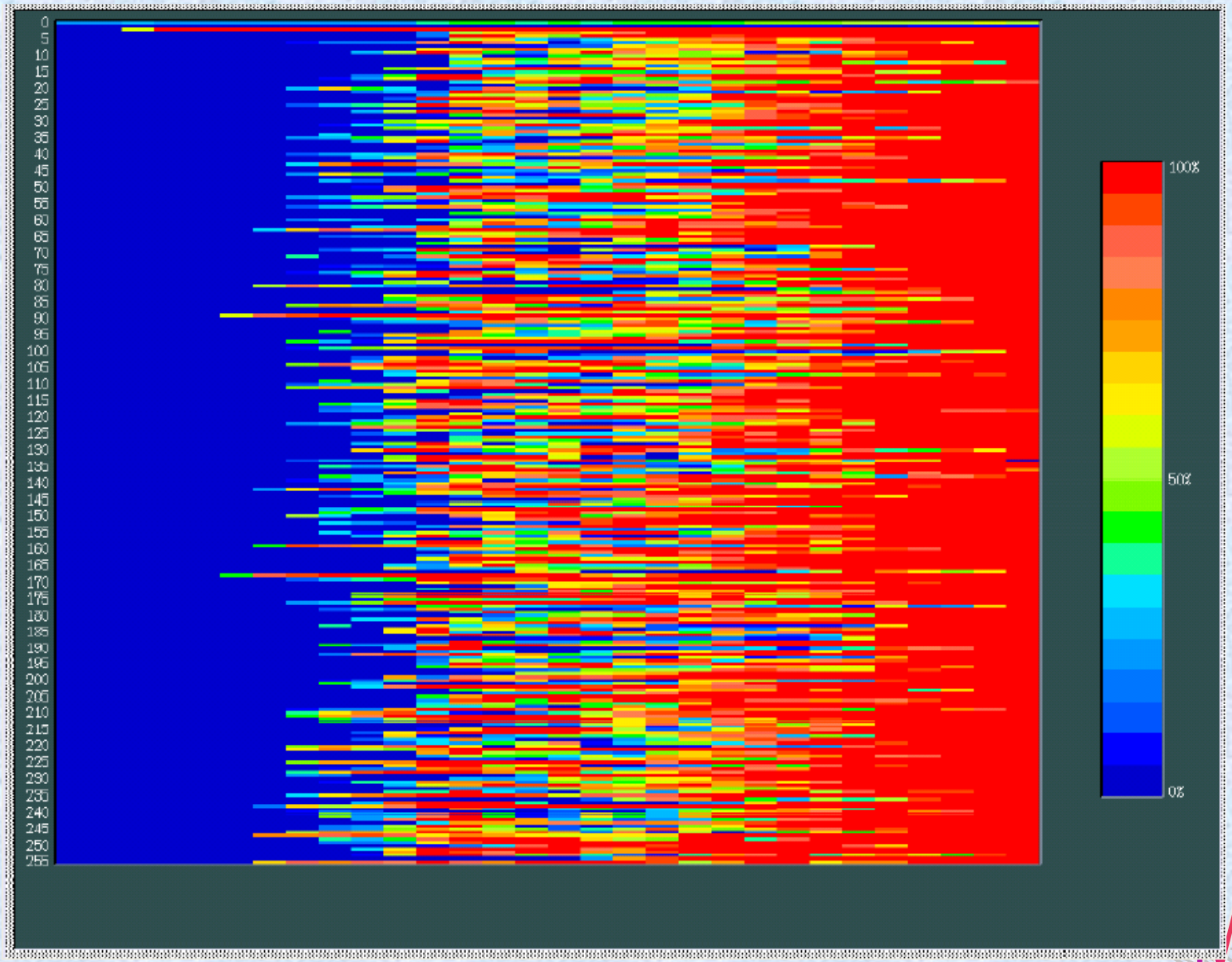
MGTP

AND並列化(台数効果)

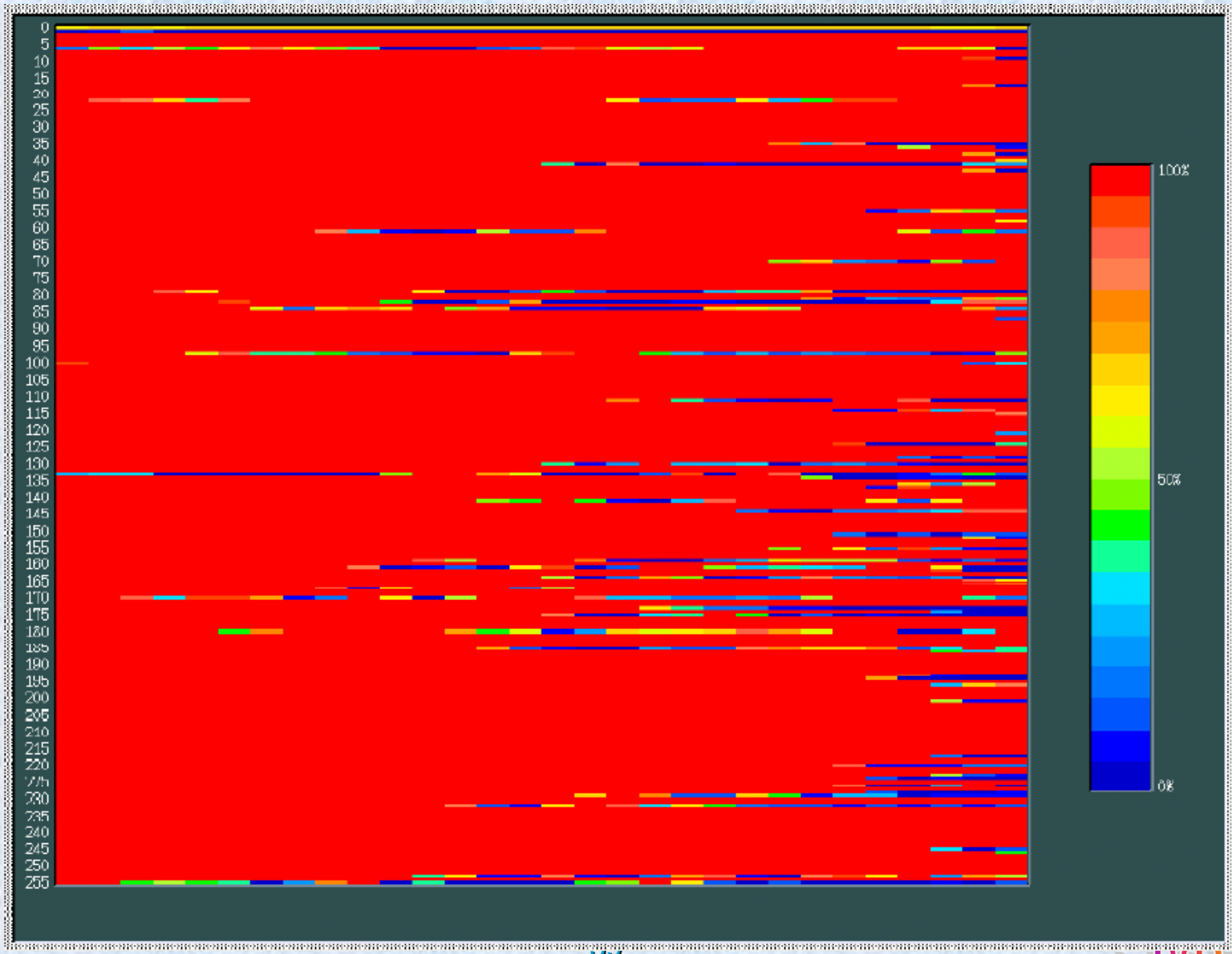


MGTP

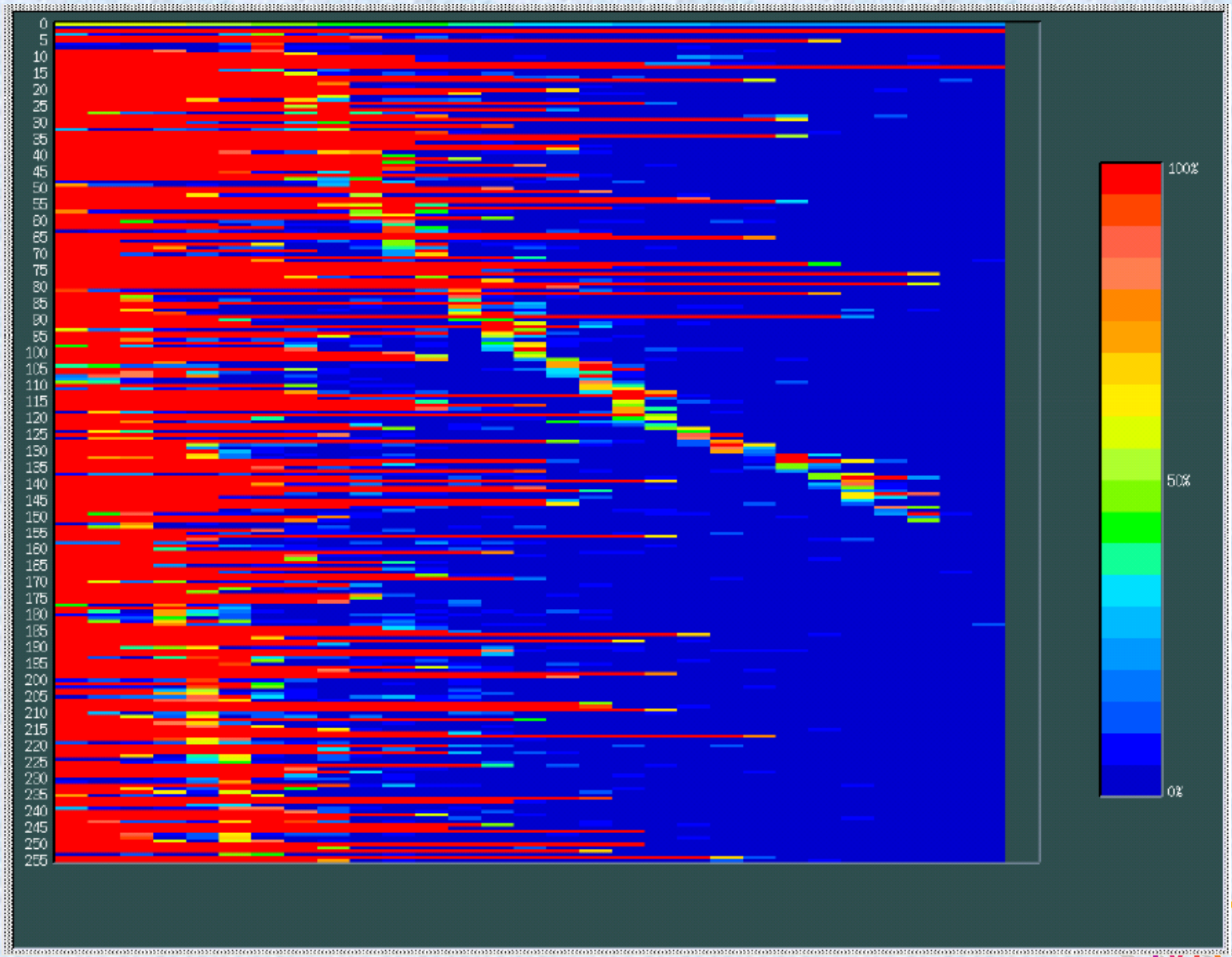
循環割付け方式(開始時)


0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
100%
50%
0%

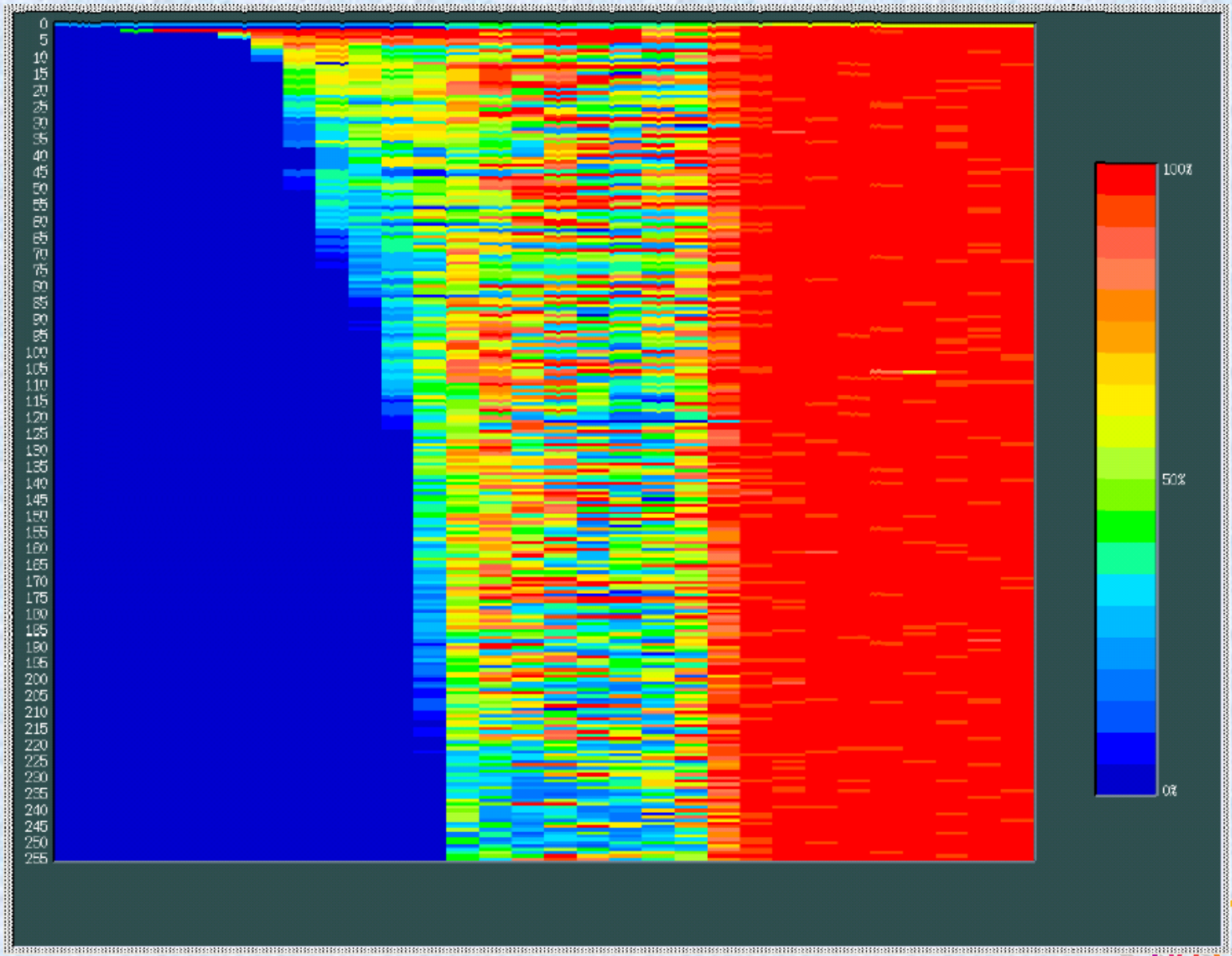
循環割付け方式(途中)


0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
100%
50%
0%

循環割付け方式(終了時)


0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
100%
50%
0%

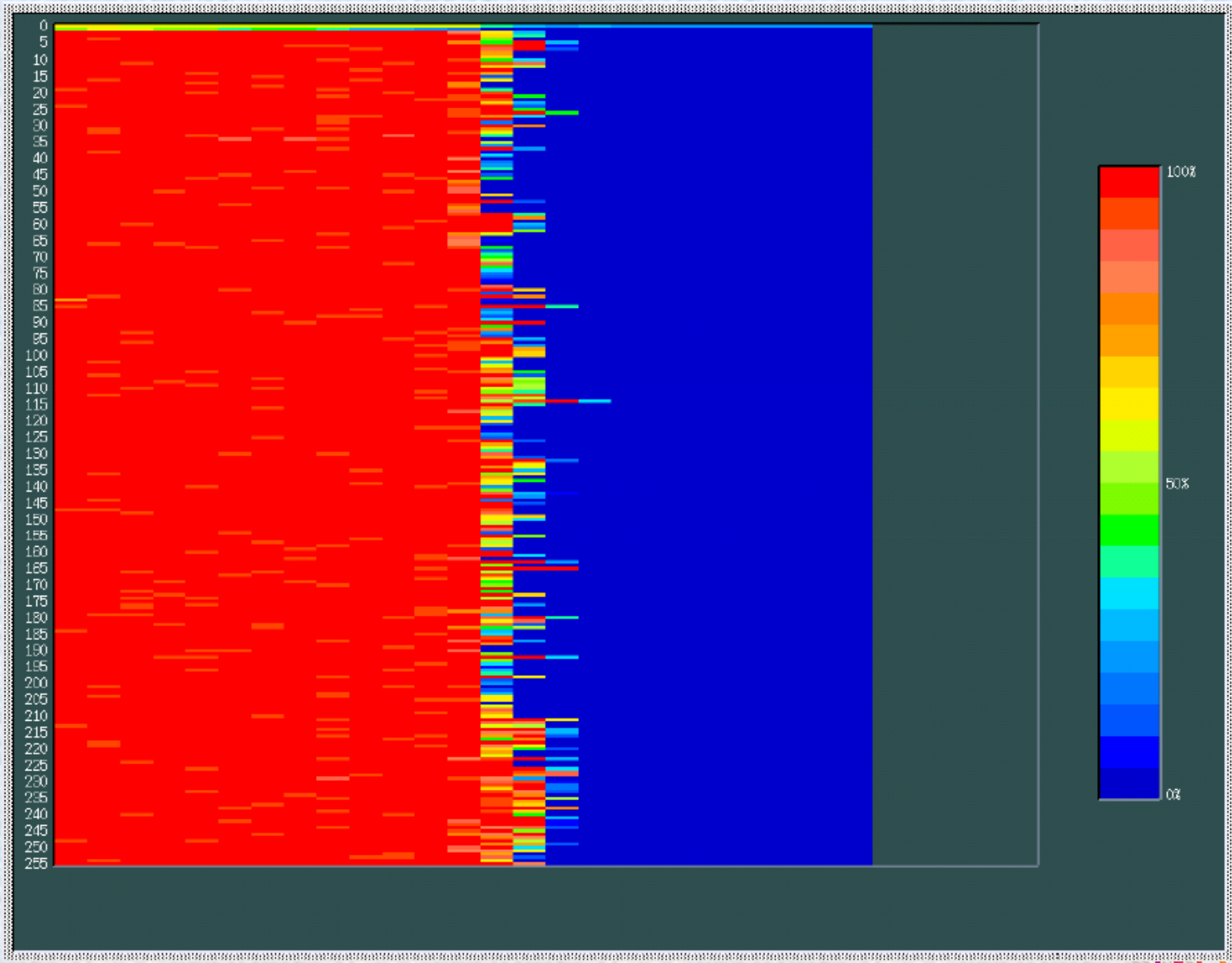
N-逐次方式(開始時)


0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
100%
50%
0%

N-逐次方式(途中)


0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
100%
50%
0%

N-逐次方式(終了時)


0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
100%
50%
0%

機能拡張と探索空間の削減

1. 制約解消

   CMGTP: 負(否定)アトムの導入

   IV-MGTP: 区間制約の導入

2. ノンホーンマジックセット法(上昇型証明＋下降型証明)

   ゴールに無関連な探索を削除

   幅優先NHMと深さ優先NHM

3. 依存性解析による冗長探索の削除

   証明濃縮(Proof condensation)： 無駄な分枝の削除

   畳み込み(Folding-up)： 重複証明の回避

4. 極小モデル生成法

   分岐仮定と分岐補題

   ⇒非極小モデルの刈り込みと極小性検査の削減

人工知能への応用等

1. 失敗による否定の実装
   notを含む論理式を様相MG節へ変換

$$A :- \ B, not \ C \Rightarrow B \rightarrow A \vee KC$$

＋ 一貫性制約、安定条件

2. MM-MGTPを利用した解集合計算
   様相MG節および極小モデル生成法を利用

3. 法的推論への適用
   NAFを用いた非単調推論の導入
   ルール間優先機能の導入

4. FPGA上のSATソルバーの開発
   PCMGTP: MGTPのハードウェア化 ⇒ ソフトの100倍高速化
   ソフト（補題、ヒューリスティクス）とハードのハイブリッド化

MGTPのモデル生成法

- **モデル拡張** $A_1 \wedge \cdots \wedge A_n \rightarrow B_1 \vee \cdots \vee B_m$

$$\frac{M \models A_1 \wedge \cdots \wedge A_n \qquad M \not\models B_1 \vee \cdots \vee B_m}{M \cup \{B_1\} \quad | \quad \cdots \quad | \quad M \cup \{B_m\}}$$

- **モデル棄却** $A_1 \wedge \cdots \wedge A_n \rightarrow false$

$$\frac{M \models A_1 \wedge \cdots \wedge A_n}{\bot}$$

- **単位反駁　および　単位簡約**

$$\frac{L \in M \quad \overline{L} \in M}{\bot} \qquad \frac{\overline{B_j} \in M \quad B_1 \vee \ldots \vee B_j \vee \ldots \vee B_m}{B_1 \vee \ldots \vee B_{j-1} \vee B_{j+1} \vee \ldots \vee B_m}$$

準群(QG)の存在問題

∘	1	2	3	4	5
1	1	3	2	5	4
2	5	2	4	3	1
3	4	5	3	1	2
4	2	1	5	4	3
5	3	4	1	2	5

$$\rightarrow dom(1) \wedge dom(2) \wedge dom(3) \wedge dom(4) \wedge dom(5).$$

$$dom(M) \wedge dom(N) \rightarrow p(M,N,1) \vee p(M,N,2) \vee p(M,N,3) \vee p(M,N,4) \vee p(M,N,5).$$

$$p(Y,X,A) \wedge p(A,Y,B) \wedge p(B,Y,C) \wedge \{C \neq X\} \rightarrow .$$

$$p(X,X,A) \wedge \{X \neq A\} \rightarrow .$$

$$p(X,A,Y) \wedge p(X,B,Y) \wedge \{A \neq B\} \rightarrow .$$

$$p(A,X,Y) \wedge p(B,X,Y) \wedge \{A \neq B\} \rightarrow .$$

QG5: $x \circ x = x \wedge$
$y \circ x \circ y \circ y = x$

MGTP

CMGTPの記述例

$$p(Y,X,A) \wedge p(A,Y,B) \wedge p(B,Y,C) \wedge \{C \neq X\} \rightarrow .$$

⇓　CMGTP用に書換え

$$p(Y,X,A) \wedge p(A,Y,B) \rightarrow p(B,Y,X).$$
$$p(Y,X,A) \wedge p(B,Y,X) \rightarrow p(A,Y,B).$$
$$p(A,Y,B) \wedge p(B,Y,X) \rightarrow p(Y,X,A).$$
$$p(Y,X,A) \wedge \neg p(B,Y,X) \rightarrow \neg p(A,Y,B).$$
$$p(Y,X,A) \wedge \neg p(A,Y,B) \rightarrow \neg p(B,Y,X).$$
$$p(A,Y,B) \wedge \neg p(Y,X,A) \rightarrow \neg p(B,Y,X).$$
$$\neg p(B,Y,X) \wedge p(A,Y,B) \rightarrow \neg p(Y,X,A).$$
$$\neg p(A,Y,B) \wedge p(B,Y,X) \rightarrow \neg p(Y,X,A).$$
$$\neg p(Y,X,A) \wedge p(B,Y,X) \rightarrow \neg p(A,Y,B).$$

MGTP

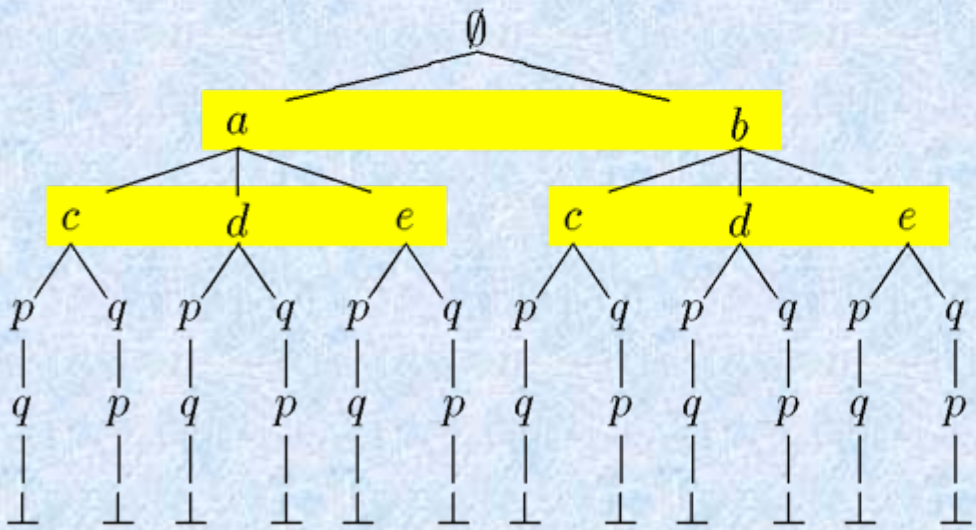
他システムとの比較(失敗枝数)

QG5	DDPP	FINDER	MGTP	CMGTP
9	15	40	239	15
10	50	356	7026	38
11	136	1845	51904	117
12	443	13527	2749676	372
13				13914
14				64541
15				151250
16				19382469

DDPP: Davis&Putnum with Discrimination Tree
FINDER: Finite-domain solver
QG5-11: No. of models = 5
Others: No. of models = 0

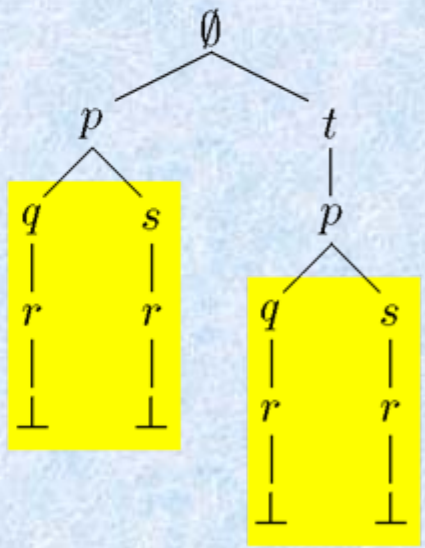
モデル生成法の冗長性

C1: $\rightarrow\ a \vee b.$
C2: $\rightarrow\ c \vee d \vee e.$
C3: $\rightarrow\ p \vee q.$
C4: $p \rightarrow q.$
C5: $q \rightarrow p.$
C6: $p \wedge q \rightarrow\ .$



証明に無関連なモデル拡張

C1: $\rightarrow\ p \vee t.$
C2: $p \rightarrow\ q \vee s.$
C3: $q \rightarrow\ r.$
C4: $s \rightarrow\ r.$
C5: $t \rightarrow\ p.$
C6: $p \wedge r \rightarrow\ .$

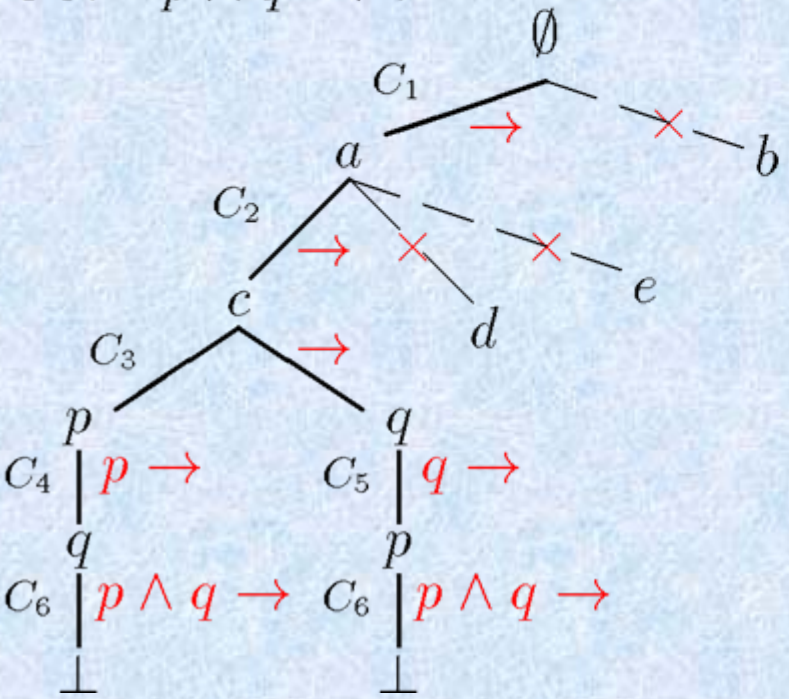


分岐後の重複証明

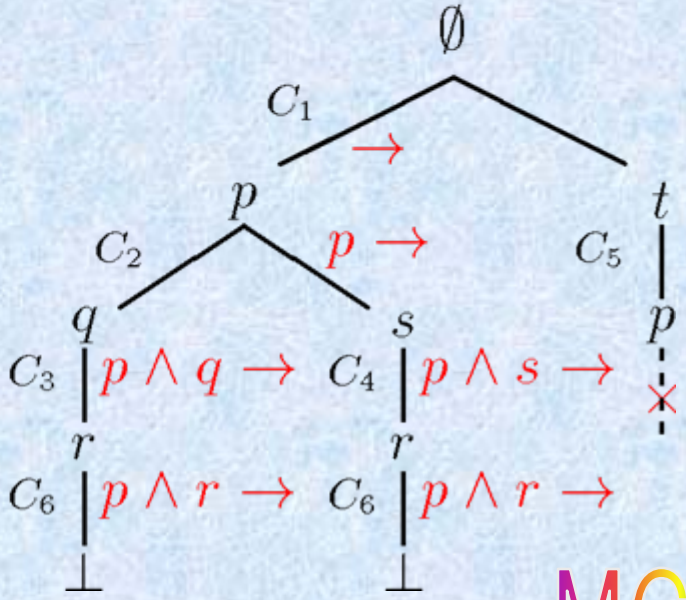
MGTP

学習節による冗長探索の削除

C1: $\rightarrow\ a \vee b.$
C2: $\rightarrow\ c \vee d \vee e.$
C3: $\rightarrow\ p \vee q.$
C4: $p \rightarrow\ q.$
C5: $q \rightarrow\ p.$
C6: $p \wedge q \rightarrow\ .$



C1: $\rightarrow\ p \vee t.$
C2: $p \rightarrow\ q \vee s.$
C3: $q \rightarrow\ r.$
C4: $s \rightarrow\ r.$
C5: $t \rightarrow\ p.$
C6: $p \wedge r \rightarrow\ .$



MGTP

Implicationによる刈り込み

C1: $\rightarrow r.$

C2: $r \rightarrow a \vee b.$

C3: $r \rightarrow p \vee c \vee d.$

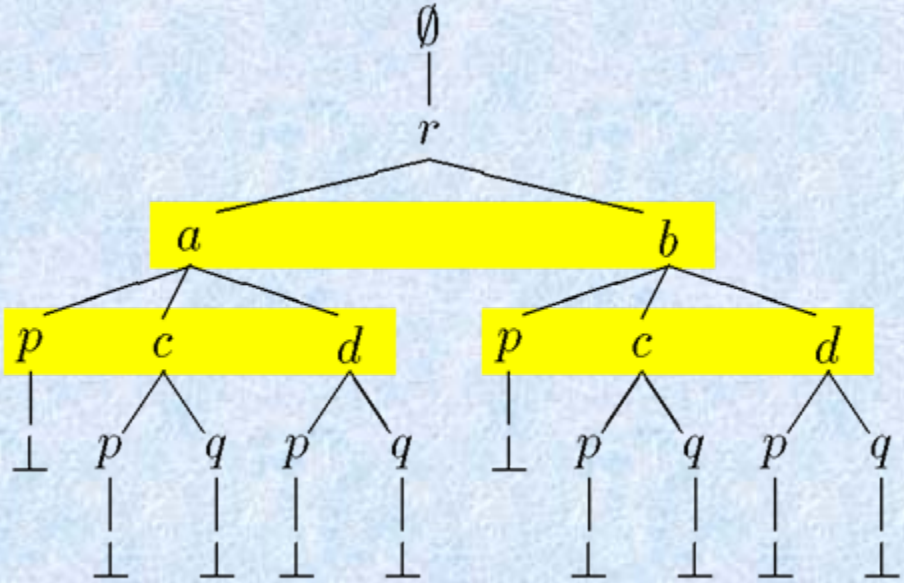
C4: $r \rightarrow p \vee q.$

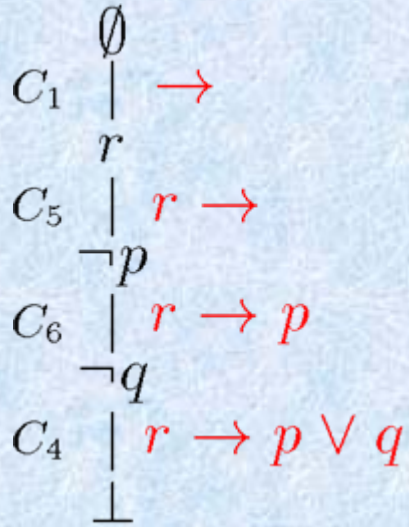
C5: $p \rightarrow .$

C6: $q \rightarrow .$

Implication (UR-resolution)

$$\frac{M = \{a, \neg c\} \quad a \wedge b \rightarrow c}{\neg b}$$

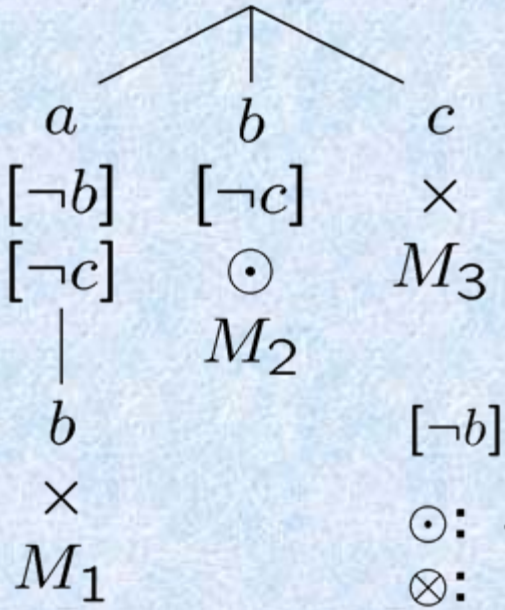




MGTP

分岐仮定による非極小モデルの棄却

$$S_1 = \left\{ \begin{array}{l} \rightarrow a \vee b \vee c \\ a \rightarrow b \\ c \rightarrow \end{array} \right\}$$

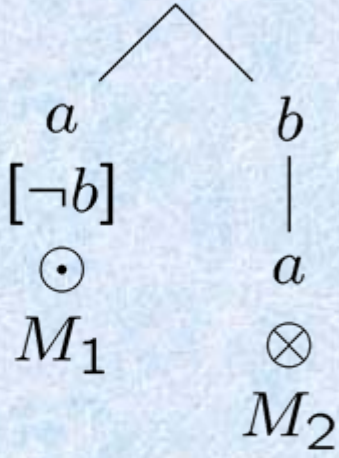


$[\neg b], [\neg c]$: 分岐仮定

⊙: 極小モデル
⊗: 非極小モデル
×: 棄却

(a) 有効な分岐仮定

$$S_2 = \left\{ \begin{array}{l} \rightarrow a \vee b \\ b \rightarrow a \end{array} \right\}$$



(b) 無効な分岐仮定

⇓ (Bryらの方法)
モデル制約$\{a \rightarrow\}$
をS_2に付加

MGTP

分岐補題による刈り込み

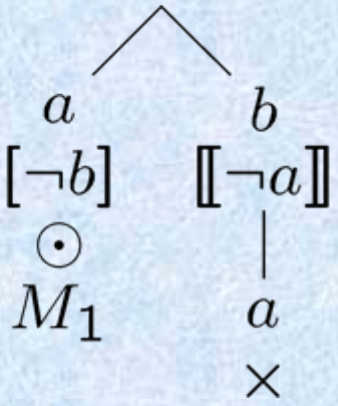
対称的分岐仮定の使用

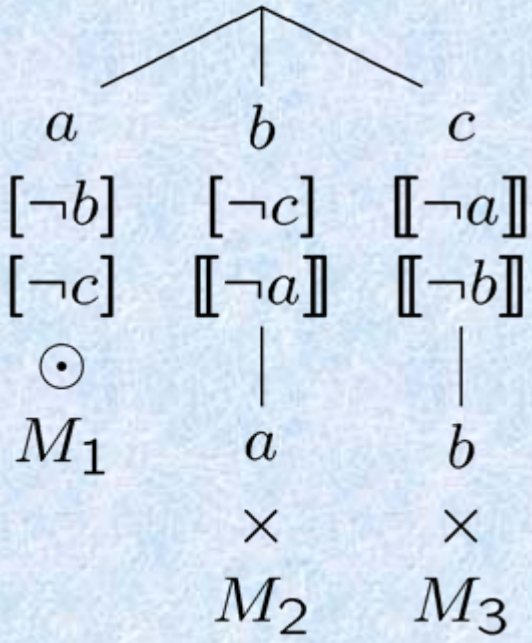
$$\begin{array}{c|c} A \vee B & \\ \hline A & B \\ [\neg B] & [\![\neg A]\!] \end{array}$$

$$S_3 = \left\{ \begin{array}{l} \rightarrow a \vee b \vee c \\ b \rightarrow a \\ c \rightarrow b \end{array} \right\}$$

$[\![\neg A]\!]$: 分岐補題

Aの証明で，分岐仮定$[\neg B]$を使用していないとき生成.





MGTP

SAT ベンチマークの結果1

Prob	Vars	Cls	MMs	Conf Minisat	Conf MGTP	Time Minisat	Time MGTP
Industrial							
depots1	161	496	2	13	**8**	0.02	0.02
ferry7	1946	22336	5040	**28375**	50943	**9.97**	15.13
driver.is	128	493	149184	**18185**	110167	381.59	**3.50**
driver.ar	128	489	676269	---	**3**	T.O.	**5.56**
Crafted							
qg4-08	512	9685	0	735	**598**	**0.13**	0.16
qg4-09	729	15580	194	37660	**26866**	**7.92**	10.67
qg5-09	729	28540	0	19	**15**	0.11	**0.03**
qg7-09	729	22060	4	28	**4**	0.11	**0.02**

MGTP

SAT ベンチマークの結果2

Prob	Vars	Cls	MMs	Conf Minisat	Conf MGTP	Time Minisat	Time MGTP
Random							
uf100-06	100	430	4730	3811	18391	0.61	0.39
uf200-01	200	860	3	14992	92611	0.53	6.09
uuf100-01	100	430	0	371	588	0.05	0.05
uuf150-01	150	645	0	3203	12498	0.13	0.70
MM-Crafted							
ex1	50	5	10000	20475	0	119.78	0.22
ex3	65	78	65536	13597	1	66.36	0.69
ex4	20	7	341	0	0	0	0.05
ex5	20	16	17	5	9	0.02	0.02

MGTP

まとめ・雑感

- KL1・PIM

  KL1はやはり並列のラピッドプロトタイピングに適している。
  並列定理証明は計算量を低減する訳ではないが、それでも
  未解決問題を含む難問の解決に役立つ。

- 値域限定モデル生成型証明

  値域限定の制約のため無限領域の問題には無力だが、
  有限領域である人工知能の多くの問題に適用できる。

- MGTP vs DPLL

  MGTPはDPLLにない1階表現能力を有すが、双方向推論能力
  はない。これを補うため、7月からMGTPの改良に着手した
  （単位負節による連言照合、implication、学習節の利用）。
  現在では、Minisatと遜色のない推論能力を有す。