Theorem-proving strategies: a search-oriented taxonomy

MARIA PAOLA BONACINA
DEPT. OF COMPUTER SCIENCE
THE UNIVERSITY OF IOWA

Theorem proving

H: assumptions

φ: conjecture

$$H \overset{?}{\models} \varphi$$

H may be:

- a mathematical theory
  (e.g., algebra
  geometry
  analysis)
- a specification of a system
  (e.g., message-passing system)

Refutational theorem proving

$$H \cup \{\neg\varphi\}$$

either prove φ by generating a proof $H \cup \{\neg\varphi\} \vdash \bot$

or disprove φ by generating a model of $H \cup \{\neg\varphi\}$

In general: semi-decidable

However, TP works:

- Moufang identities in rings
  S. Anantharaman, J. Hsiang
  SBR2 1990

- Axiomatization of Lukasiewicz many-valued logic
  S. Anantharaman, M.P. Bonacina
  SBR 3 1989-90

- Single axioms for groups
  W. McCune OTTER 1993

- Robbins algebras are Boolean
  W. McCune EQP 1996

And not only in math:

- Deductive composition of sw from subroutine libraries
  (M. E. Stickel et al.
  SNARK 1994)

- Verification of cryptographic protocols
  (J. Schumann SETHEO 1997)

- Modelling + verification of message-passing systems
  (W. McCune IVY 1999)

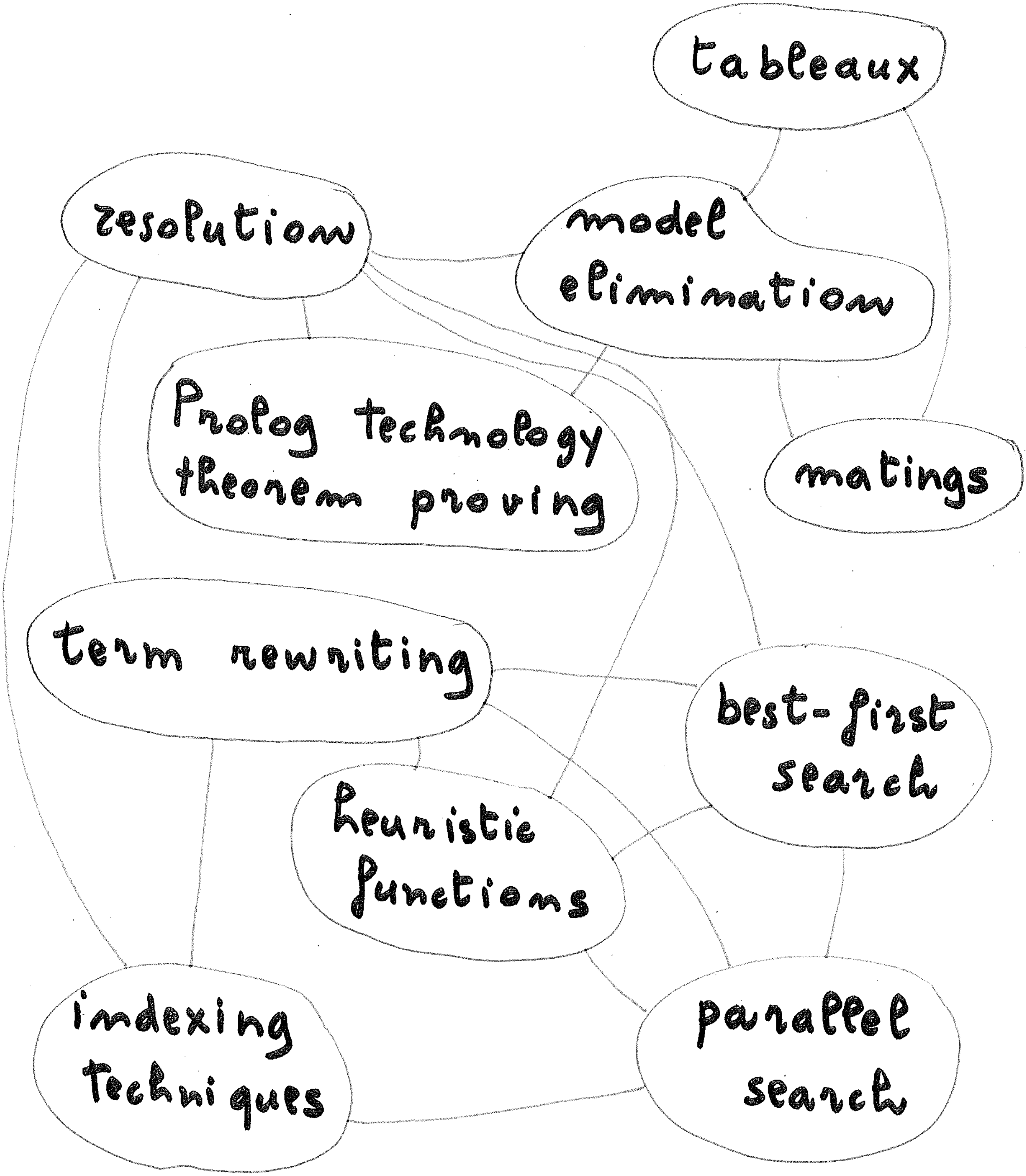
Many ___ systems:

- Fully automated T.P.
(OTTER, REVEAL, EQP, SETHEO, PROTEIN ...)

- Interactive T.P.
(ISABELLE, HOL, COQ, PVS ...)

- LIBRARIES of problems and proofs
(MIZAR, TPTP ...)

Many ingredients:



2 main types of ingredients:

inference rules

search plans

inference system $\mathcal{I}$

$+$

search plan Σ

T. P. strategy $\mathcal{C}$

$$\mathcal{C} = \langle \mathcal{I} ; \Sigma \rangle$$

A search-oriented taxonomy

$\left.\begin{array}{l}\text{inference}\\\text{search}\end{array}\right\}$ equally important

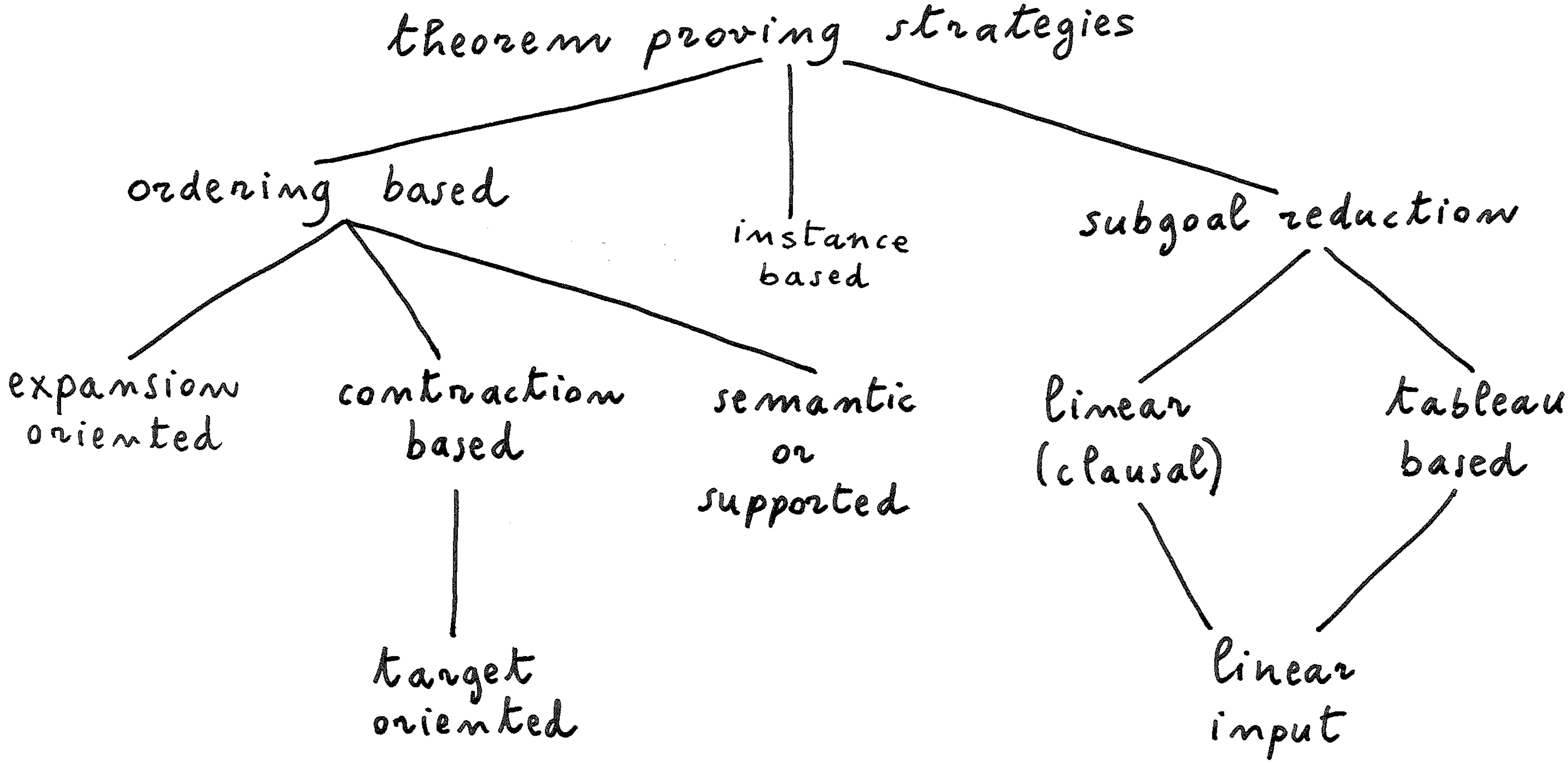
e.g.,

* Parallelization
* Machine-independent evaluation
* Engineering of T.P.

M.P. BONACINA
"A TAXONOMY OF THEOREM PROVING STRATEGIES" IN
"ARTIFICIAL INTELLIGENCE TODAY"
LNAI 1600, PP. 43-84, 1999

A taxonomy of strategies

First order, general purpose, fully automated



Ordering-based strategies

$H \cup \{\neg\varphi\} \longrightarrow S$: set of clauses

$>$: well-founded ordering on terms, atoms, literals, clauses, sets of clauses

Ex.: CSO

stable: $s > t \Rightarrow s\sigma > t\sigma$

monotonic: $s > t \Rightarrow c[s] > c[t]$

subterm property: $c[s] > s$

total on ground

[Nachum Dershowitz 1982]

Example: LRPO

$ack(0,y) = succ(y)$

$ack(succ(x),0) = ack(x, succ(0))$

$ack(succ(x), succ(y)) = ack(x, ack(succ(x),y))$

$ack(0,y) > succ(y)$

$ack(succ(x), 0) > ack(x, succ(0))$

$ack(succ(x), succ(y)) > ack(x, ack(succ(x), y))$

assuming $ack > succ > 0$

[LRPO: Kamin - Lévy 1980]

Ordering-based strategies

work on sets of clauses

Expansion inference rules:

$$f: \frac{S}{S'} \qquad S \subset S' \quad S \prec S'$$

e.g.: ordered resolution

$$\frac{S \cup \{L \vee C, \neg M \vee D\}}{S \cup \{L \vee C, M \vee D, (C \vee D)\sigma\}} \; L\sigma = M\sigma$$

$L\sigma \not\leq X\sigma \quad \forall X \in C$

$M\sigma \not\leq X\sigma \quad \forall X \in D$

also: hyperresolution,

paramodulation, superposition ...

Ordering-based strategies

Contraction inference rules:

$$f: \frac{S}{S'} \qquad S \not\subseteq S' \qquad S > S'$$

e.g. simplification

$$\frac{S \cup \{ L[s] \vee C,\ l = r \}}{S \cup \{ L[r\sigma] \vee C,\ l = r \}} \quad s = l\sigma$$

$$l\sigma > r\sigma \quad \Longrightarrow \quad L[s] > L[r\sigma]$$

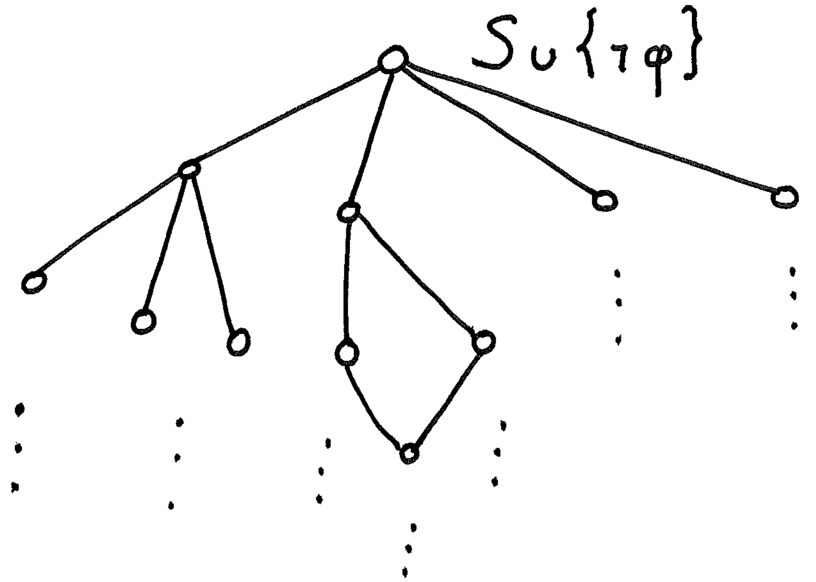
$L[s] \vee C$ is redundant

also: subsumption, taut. deletion, purity deletion, clausal simplification

Theorem proving as search problem

Inference system I

$S \cup \{\neg\varphi\} \overset{?}{\vdash}_{I} \bot$



Vertex : state   arc : inference

path : derivation

Search plan Σ : determines unique derivation

Refutationally complete I
Fair Σ
$\Rightarrow$ theorem-proving strategy $\mathcal{C} = \langle I, \Sigma \rangle$ complete

General scheme of search plan

$$\Sigma = < \zeta , \xi , \omega > \qquad \text{(at least)}$$

- rule-selecting function

  $\zeta : States^* \longrightarrow I$

- premise-selecting function

  $\xi : States^* \longrightarrow \mathcal{P}(\mathcal{L}_{\Theta})$

- termination-detecting function

  $\omega : States \longrightarrow Bool$

Search plan for ord-based strat.

$$\Sigma = \langle \zeta, \xi_1, \xi_2, \omega \rangle$$

- $\xi_1 : States^* \longrightarrow \mathcal{L}_\Theta$

  $\xi_1(S_0, \dots S_i) = \psi_1 \in S_i$

- $\zeta : States^* \times \mathcal{L}_\Theta \longrightarrow I$

  $\zeta(S_0, \dots S_i, \psi_1) = f$

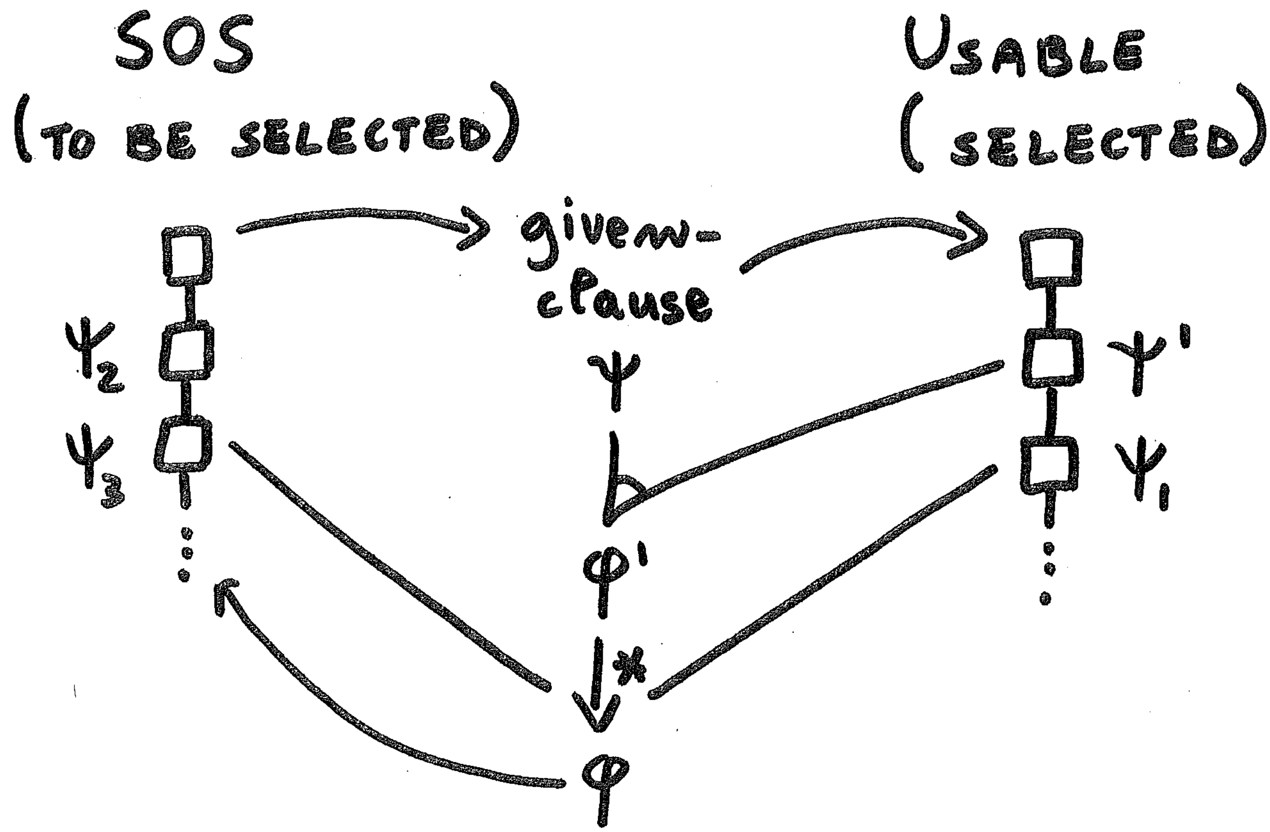
- $\xi_2 : States^* \times \mathcal{L}_\Theta \times I \longrightarrow \mathcal{P}(\mathcal{L}_\Theta)$

  $\xi_2(S_0, \dots S_i, \psi_1, f) = \{\psi_2 \dots \psi_n\} \subseteq S_i$

Eager contraction:
contraction-based strategies

Example: given-clause plan

(OTTER, SPASS, GANDALF, VAMPIRE...)



	ξ_1	ξ	ξ_2
Expansion	ψ	e-rule	ψ'
Forward contraction	φ'	c-rule	ψ_1 ψ_2
Backward contraction	φ	c-rule	ψ_3

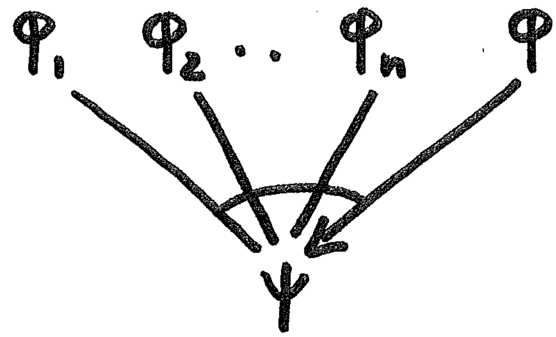
Search space

Closure S_I^*

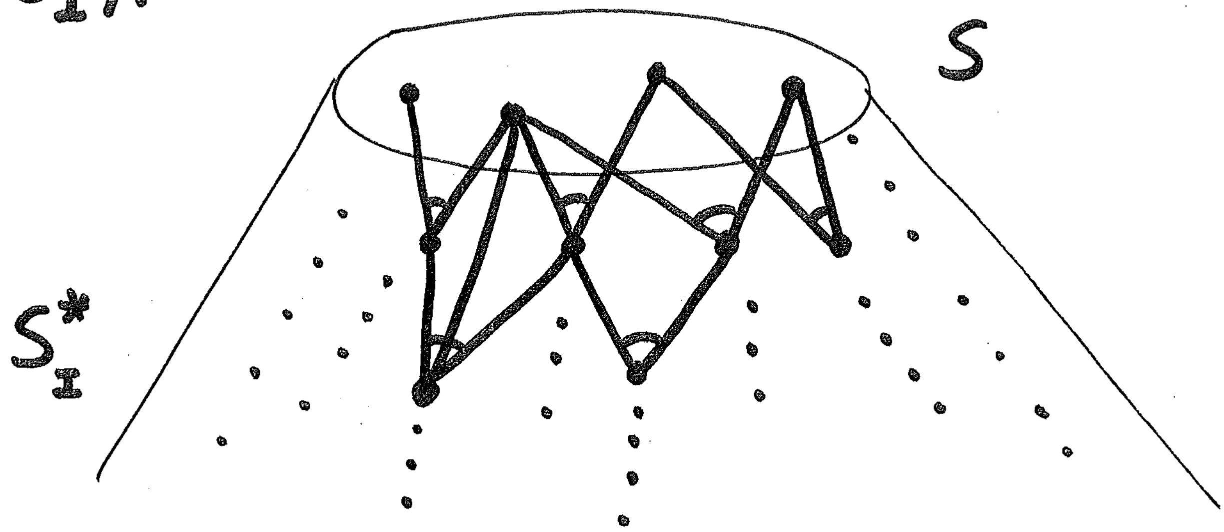
Search graph $G(S_I^*) = \langle V, E, \ell, h \rangle$

V: vertices : clauses
(equiv. classes of variants)

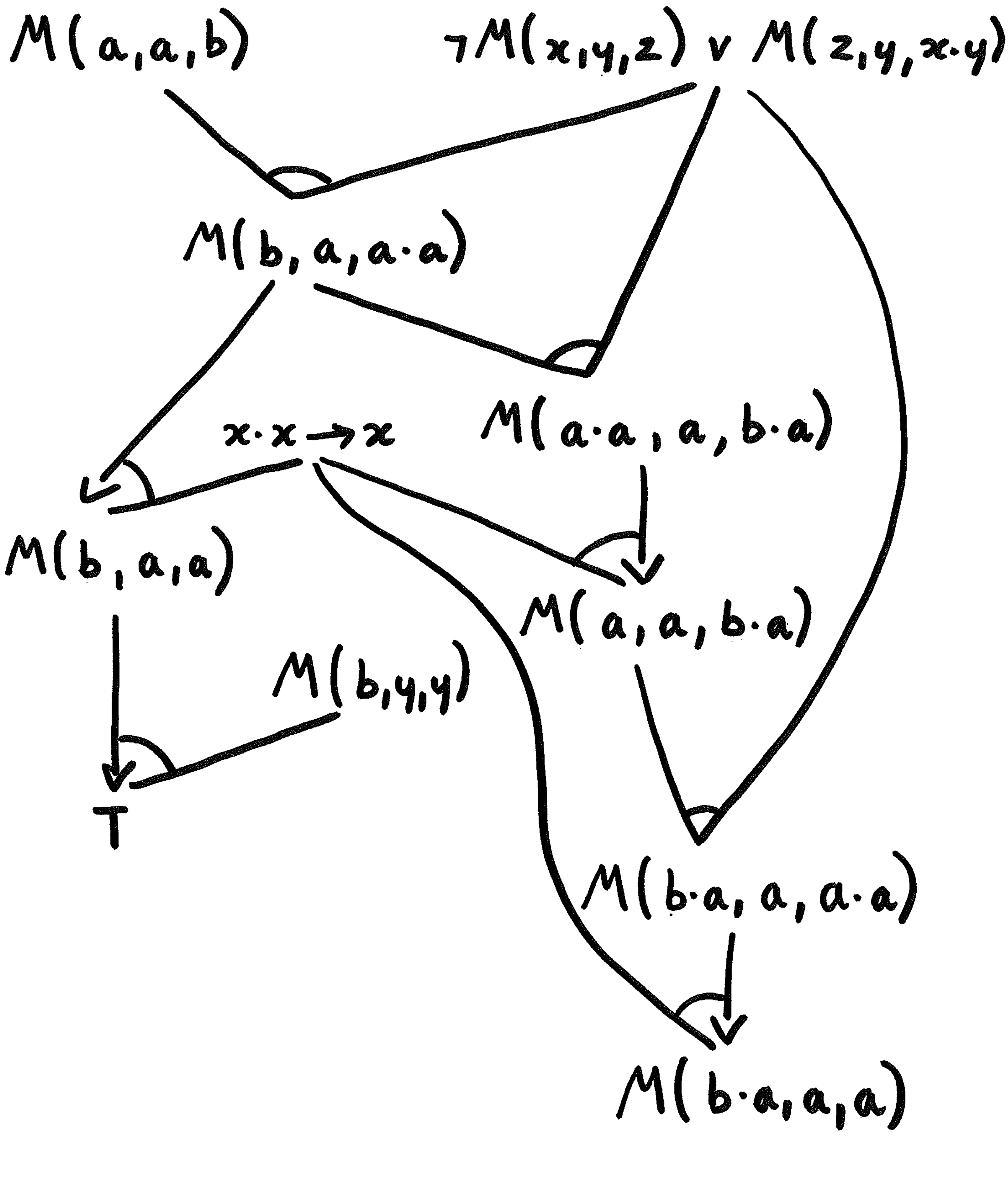
E: hyperarcs : inferences e.g.



$G(S_I^*)$:



Example:



M(a,a,b)
¬M(x,y,z) ∨ M(z,y,x·y)
M(b,a,a·a)
x·x → x
M(a·a,a,b·a)
M(b,a,a)
M(a,a,b·a)
M(b,y,y)
T
M(b·a,a,a·a)
M(b·a,a,a)

Dynamic search space

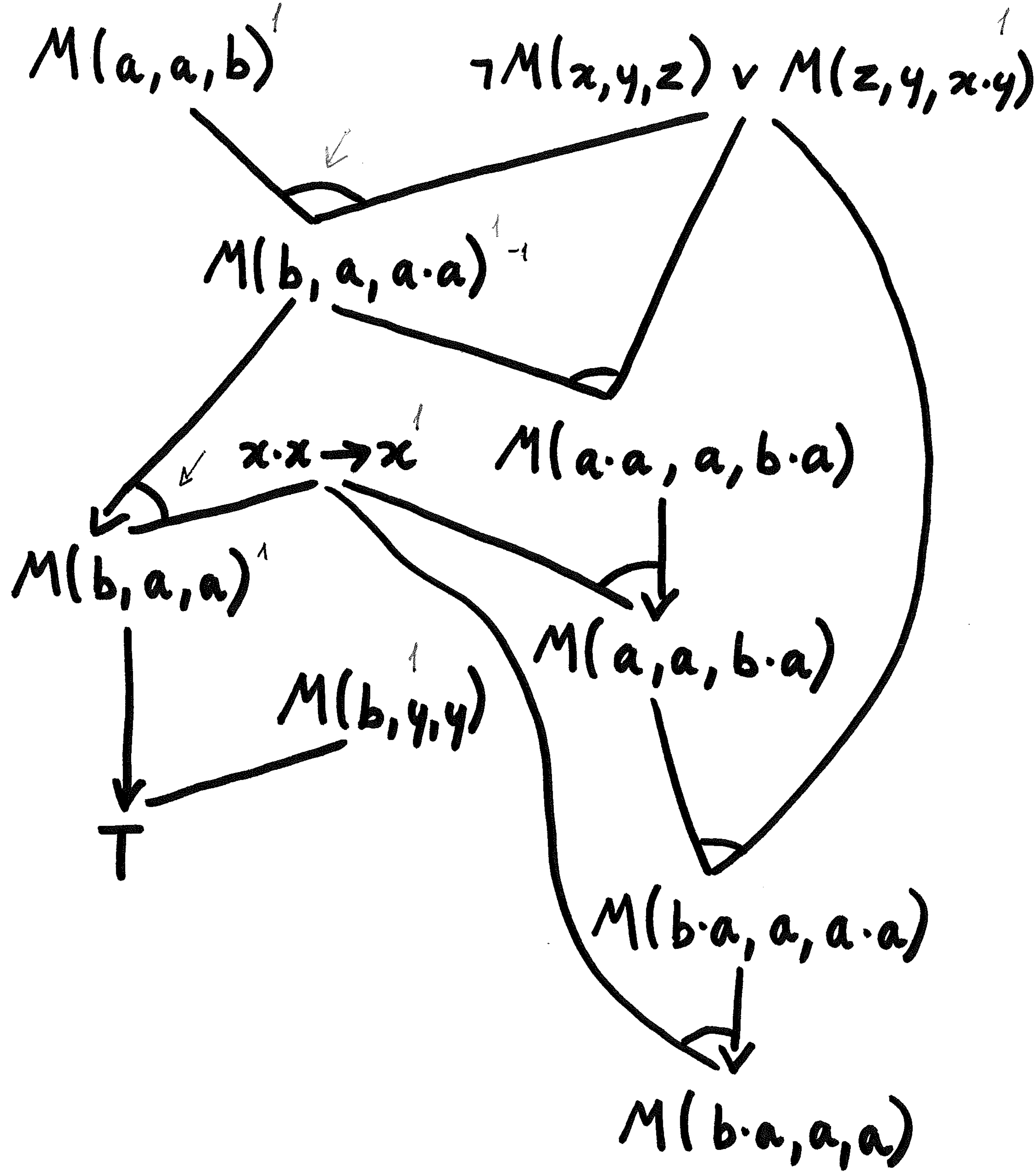
Which clauses are generated / deleted ?

Marked search graph:

$$G = \langle V, E, l, h, s \rangle$$

$$s(\varphi) = \begin{cases} m > 0 & \text{if } m \text{ variants of } \varphi \text{ present} \\ -1 & \text{if all variants of } \varphi \text{ deleted} \\ 0 & \text{otherwise} \end{cases}$$

Example:


M(a,a,b)
¬M(x,y,z) v M(z,y,x·y)
M(b,a,a·a)
x·x→x
M(a·a,a,b·a)
M(b,a,a)
M(a,a,b·a)
M(b,y,y)
T
M(b·a,a,a·a)
M(b·a,a,a)

Evolution of search space

$S_0 \vdash S_1 \vdash \ \ldots \quad S_i \vdash S_{i+1} \vdash \ldots$

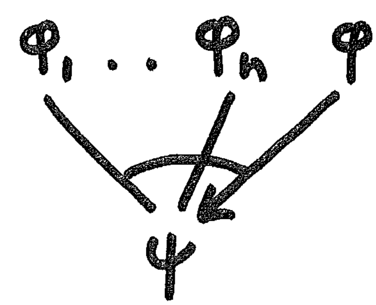
$G_0\,,\, G_1 \quad \ldots \quad G_i\,,\, G_{i+1} \ \ldots$

At stage 0:

$$s_0(\varphi) = \begin{cases} 1 & \text{if } \varphi \in S_0 \\ 0 & \text{otherwise} \end{cases}$$

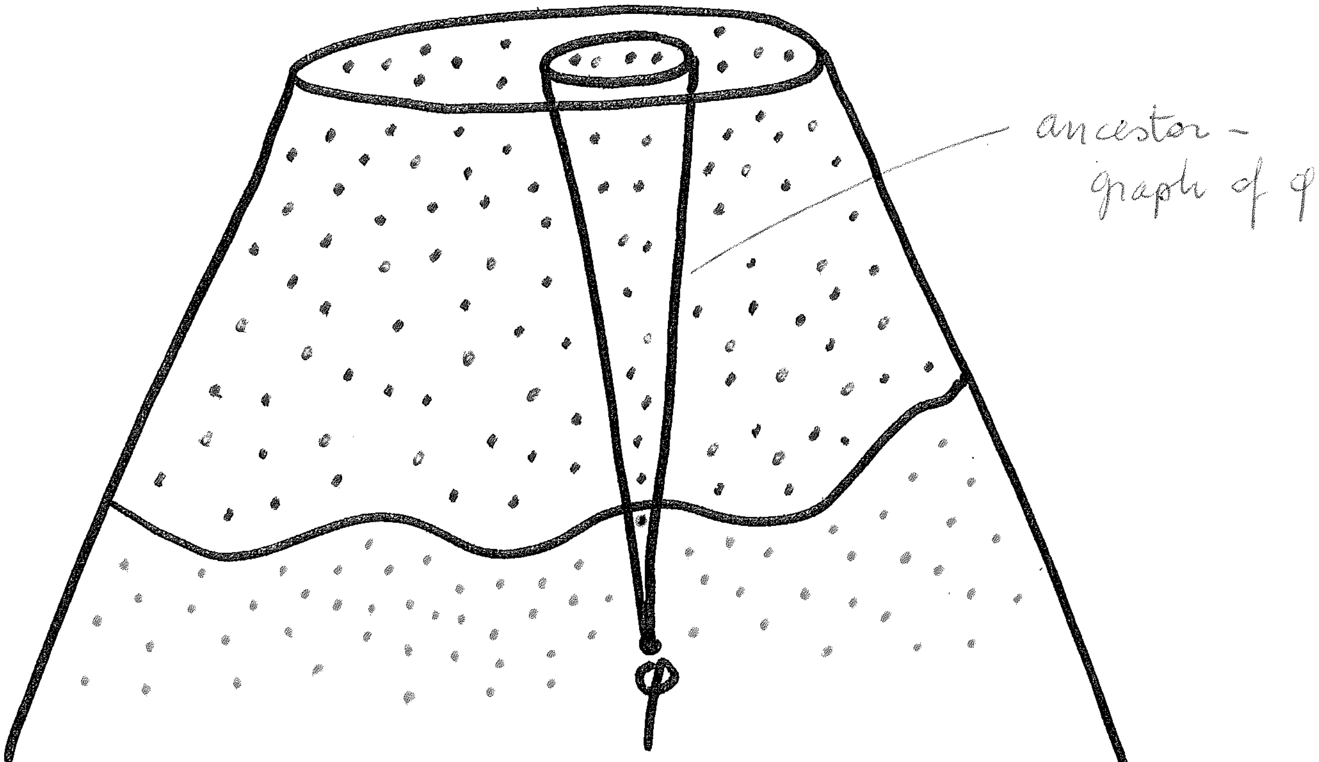
At stage i:

select hyperarc



$$s_{i+1}(x) = \begin{cases} s_i(x) + 1 & \text{if } x = \psi \ \wedge \ s_i(x) \geq 0 \\ 1 & \text{if } x = \psi \ \wedge \ s_i(x) < 0 \\ s_i(x) - 1 & \text{if } x = \varphi \ \wedge \ s_i(x) > 1 \\ -1 & \text{if } x = \varphi \ \wedge \ s_i(x) = 1 \\ s_i(x) & \text{otherwise} \end{cases}$$

Search space and proof



Active search space $(s(\varphi) > 0)$.

Generated search space $(s(\varphi) \neq 0)$..

Ancestor-graph of φ: proof of φ

Ancestor-graph of $\square$: proof
(of unsatisfiability)

Proof reconstruction

Marked search graph

Advantages:

- Graph does not change
  Marking changes

- Allows to represent contraction

- Extended to parallel search
  (one marking per process)

- Used as basis of strategy
  analysis:

  - contraction
    [Information and Computation, 1998]

  - distributed search
    [Annals of Math and AI, 1999]

Ordering-based Strategies

Work on sets of clauses

e.g., $S_0 \vdash S_1 \vdash \dots S_i \vdash \dots$

Build many proof attempts implicitly

No backtracking

Redundancy: too many clauses

Remedies: contraction

orderings

semantic refinements

Subgoal-reduction strategies

Synthetic:

e.g. Linear Resolution

generate clauses (like ord-based)

search for linear ancestor-graph of □

Analytic:

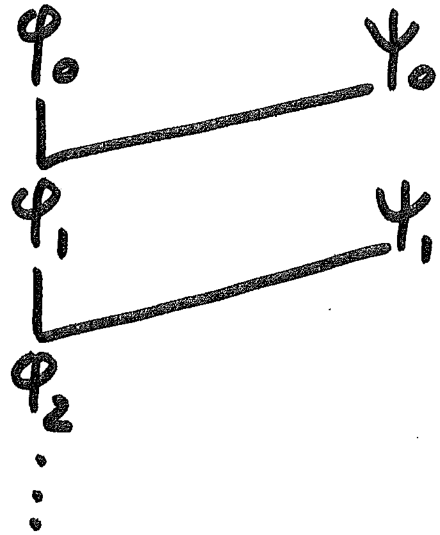
e.g. ME-tableaux

decompose clauses

survey interpretations to show none is a model

Linear Resolution

$S = T \cup \{\varphi_0\}$



input clause
or
ancestor

State: $(T\,;\ \varphi\,;\ A)$

$\Sigma = \langle\ \zeta,\ \xi_1,\ \xi_2,\ \omega\ \rangle$

$\xi_1((T; \varphi_0; A_0) \ldots (T; \varphi_i; A_i)) = L \in \varphi_i$

$\zeta : \text{States}^* \times \mathcal{L}_{H} \longrightarrow I \cup \{\text{backtrack}\}$

$\xi_2 : ((T; \varphi_0; A_0) \ldots (T; \varphi_i; A_i), L, \rho) = \psi \in T \cup A_i$

DFID

Search space

S_I^* : all subgoals of φ_0

Marked search graph:

$$G = \langle V, E, \ell, h, q \rangle$$

where marking keeps track of backtracking / failure:

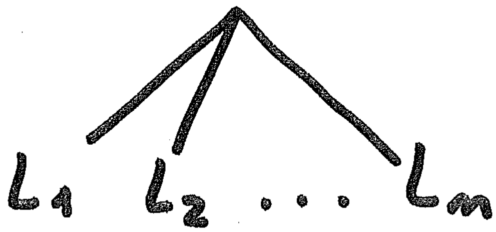
$$q(\varphi) = \begin{cases} m+1 & \text{if } \varphi \text{ has } m \text{ active goal ancestors} \\ -1 & \text{if } \varphi \text{ failed} \\ 0 & \text{otherwise} \end{cases}$$

Active search space $(q(\varphi) > 0)$

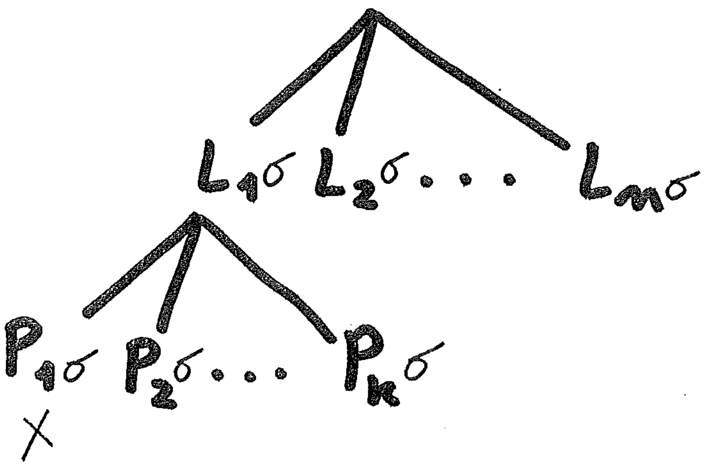
Generated search space $(q(\varphi) \neq 0)$

Model Elimination Tableaux

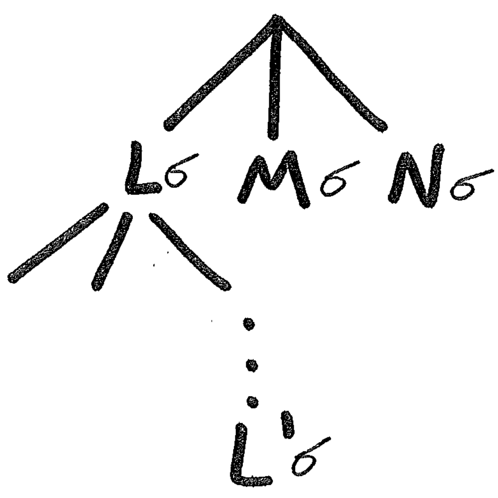
$S = T \cup \{\varphi_0\}$ $\qquad \varphi_0 = L_1 \vee .. \vee L_m$



Extension: $P_1 \vee .. \vee P_k \in T$ $\qquad P_1\sigma = \neg L_1\sigma$



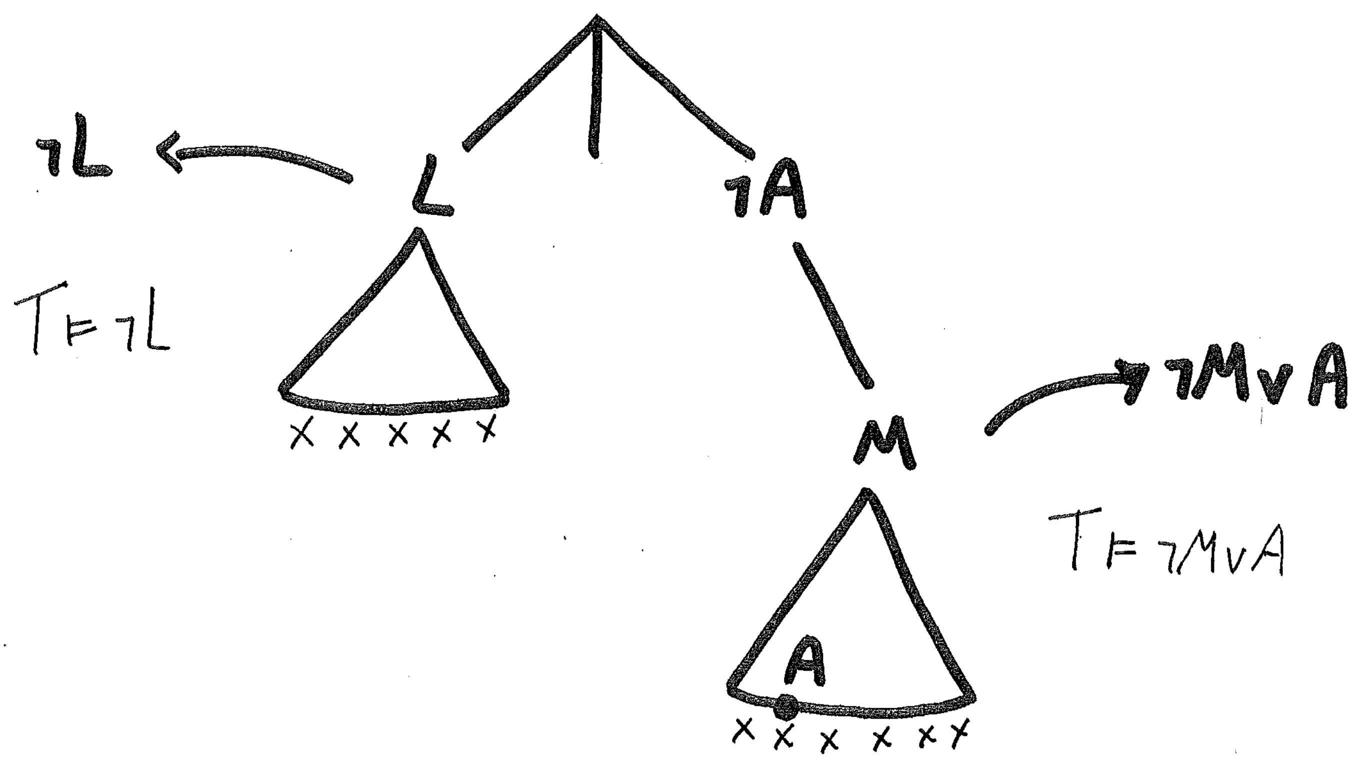
Reduction: $L\sigma = \neg L'\sigma$



All closed: proof

Model Elimination Tableaux

Lemmatization:



Also: regular tableaux only,
taut. - free

Pre-process T:

UR- resolution

contraction

Search plan for ME-tableaux

State: $(T; \mathcal{X})$

$(T_0; \mathcal{X}_0) \vdash (T_1; \mathcal{X}_1) \vdash \ldots (T_i; \mathcal{X}_i) \vdash \ldots$

$$\Sigma = \langle \zeta, \xi_1, \xi_2, \omega \rangle$$

ξ_1 selects open leaf $L \in \mathcal{X}_i$

ζ selects inference / backtrack

ξ_2 selects other premise in T_i

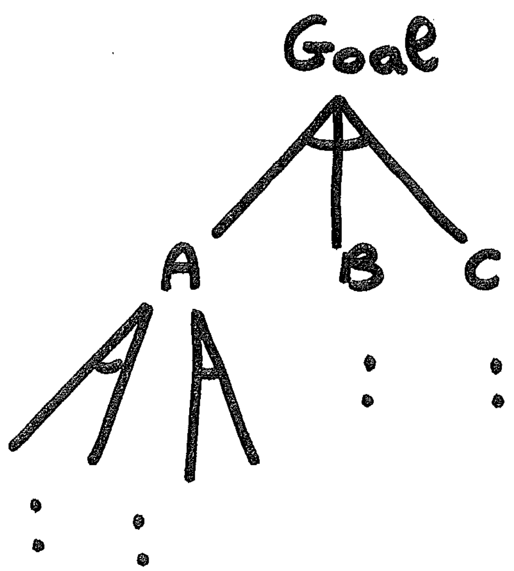
ω returns true if $\mathcal{X}_i$ closed

DFID

Search space

State space:
graph of tableaux

Analytic marked search graph:
AND-OR-graph like



with marking to keep track of:
backtracking / failure
open / closed

Subgoal-reduction strategies

Work on a goal

e.g., $\varphi_0 \vdash \varphi_1 \; \cdots \; \varphi_i \; \cdots$

$\quad\quad\;\; X_0 \vdash X_1 \; \cdots \; X_i \; \cdots$

Build explicitly one proof attempt at a time

e.g., linear deduction

tableau

Use backtracking to go to next

Redundancy: too much repetition

Remedies: lemmatization

pre-processing

Summary

	Ord- based	Subgoal- red
Gen. search space	all generated clauses	all tried tableaux
Active search space	all kept clauses	current tableau
Gen. proof	ancestor-graph of $\square$	closed tableau
Goal sensitive	NO	YES
Proof confluent	YES	NO

Frontier of the field

Integration of:

T.P. + decision procedures

Auto T.P. + interactive T.P. (proof checkers)

T.P. + Symbolic Computation (Deduction + Computation)

T.P. + model checking (verification)

Applications: PROBLEM FORMULATION

New: machine-independent eval.