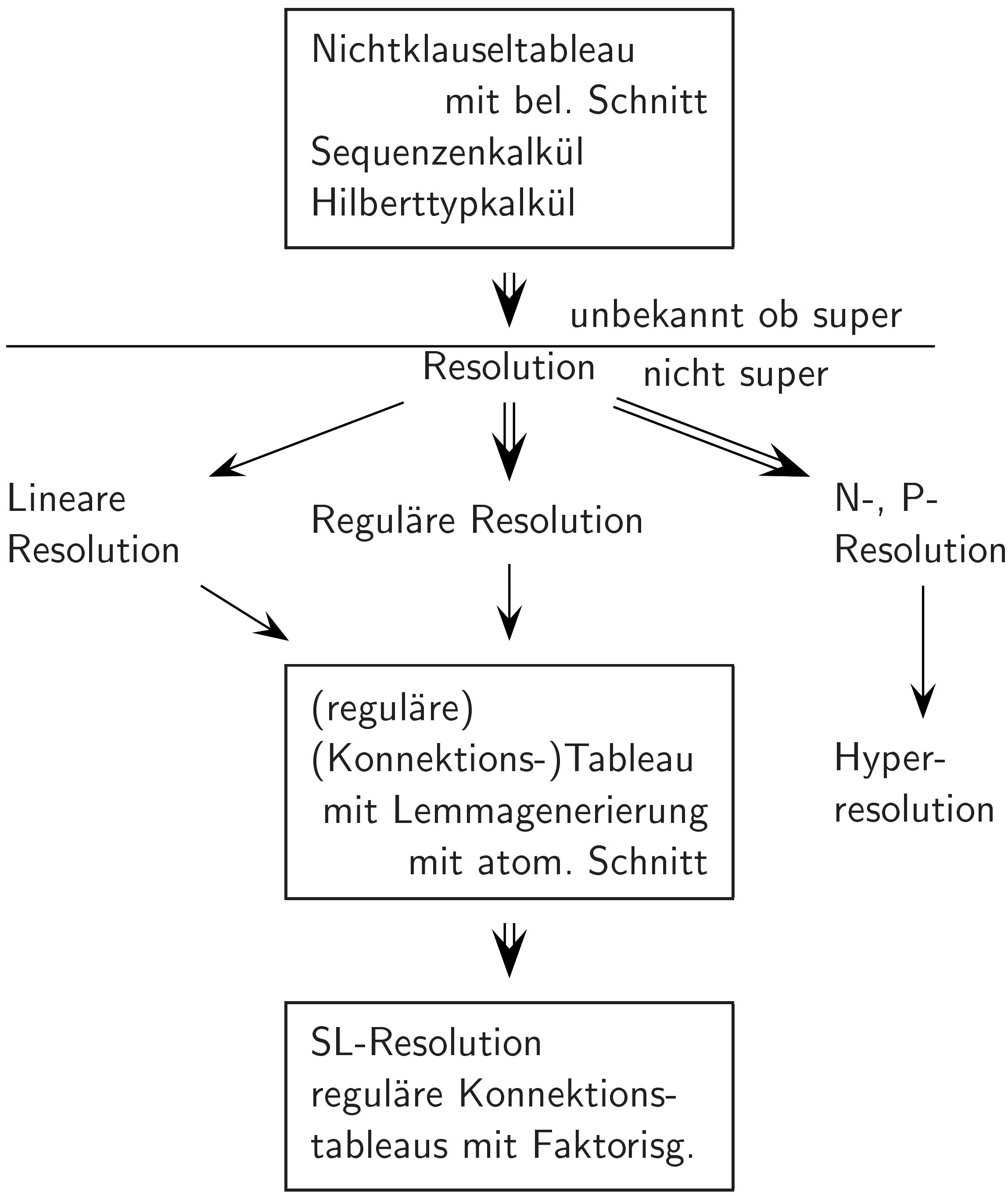


$\Uparrow$ kann p-simulieren, nicht umgekehrt

p-äquivalent

Größe des Suchraums von Beweisprozeduren

Definition. Eine **Beweisprozedur** ist ein Quintupel $\langle Z, V, i, E, u \rangle$, wobei:

- Z Menge von Zuständen
- E binäre Erreichbarkeitsrelation auf Z
- i : $\mathsf{For}_\Sigma \to 2^Z$ initiale Zustände
- u : $Z \to \{\mathsf{success}, \mathsf{no\text{-}success}\}$ Erfolg

Jedes $s \in Z$ ist mit label$(s) \subset V$ markiert:
Bäume, Klauselmengen, Ketten, . . . ■

Beispiel.

V zulässige Ketten.

$|\mathsf{label}(s)| = 1$

$(s, t) \in E$ gdw label(t) zulässige Kette und durch ME-Regel aus label(s) ableitbar.

$i(S) = \{s \in Z \mid \mathsf{label}(s) \text{ Kette zu } C \in S\}$

$u(s) = \mathsf{success}$ gdw label$(s) = \Box$.

Definition. $Z(S)$ Menge der von $i(S)$ **erreichbaren Zustände** im Graph $\langle S, E\rangle$.

$E(S)$ die Paare $e \in E$, die auf einem Pfad von einem $s_0 \in i(S)$ zu $s_1 \in Z(S)$ liegen.

Graph $G(S) = \langle Z(S), E(S)\rangle$ **Suchraum** von P für S.

■

Definition. In $G(S)$ ist **Suchtiefe** die Länge eines maximalen Pfades, der in einem Erfolgsknoten endet.

Breite von $G(S)$ ist maximale Anzahl von Zuständen aus $Z(S)$, die nicht auf gleichem Pfad liegen.

Knotengröße ist $\max_{s \in Z(S)} |\text{label}(s)|$. ■

Suchtiefe, Breite, Knotengröße heißen konstant/linear/polynomiell/exponentiell, falls sie durch eine in $|S|$ konstante/lineare/polynomielle/exponentielle Funktion approximiert werden können.

Totale **Größe des Suchraums** $G(S)$ ist

$\sum_{s \in Z(S)} |\text{label}(s)| \leq$
Produkt aus Suchtiefe, Breite und Knotengröße.

Totale Größe von $G(S)$ für S Horn sollte linear oder polynomiell sein, ist es aber oft nicht.

Beispiel.

Modellelimination hat exponentielle Suchtiefe, konstante Knotengröße und exponentielle Breite.

Forschung steht am Anfang.

Interessante offene Fragestellungen:

- Plaisteds Fragestellung noch für etliche Kalküle offen, zum Beispiel diverse Tableauverfeinerungen.

- Suchraum für andere, polynomiell entscheidbare Formelklassen, zum Beispiel 2-SAT.

- Wie kann man die Betrachtung auf volle AL oder sogar PL erweitern?

  Möglicherweise: *relative Suchraumkomplexität*

- Vergleiche Suchraum für Klausel- und Nichtklauselkalküle

Qualitativer Vergleich

Konnektionstableaus (ME, SL-Res.)	Resolution
Beweisaufzählung	Aufzählung folgerbarer Klauseln (Saturierung)
Nicht beweiskonfluent	Beweiskonfluent
Keine Gegenbeispielerzeugung	Gegenbeispielerzeugung möglich
Globale Inferenz	Lokale Inferenz
Beweisobjekte nicht universell	Resolventen universelle Klauseln
Keine Umbenennung von Variablen	Bereinigung der Elternklauseln
Lange Beweise	Relativ kurze Beweise möglich
Kleiner lokaler Suchraum	Großer Suchraum
Keine Subsumtion	Subsumtion
Schnittfrei/analytische Schnitte	nicht analytische Schnitte

Konnektionstableaus (ME, SL-Res.)	Resolution
Zielorientiert	Nicht/schwach (SOS) zielorientiert
Lifting einfach	Lifting aufwendiger
Nichtklauselversionen erhältlich Beweis erhält Formelstruktur	Nichtklauselresolution ineffizient Beweis zerstört Formelstruktur

Beweiskonfluente Tableaus oft wie KT, aber:

– Erweiterung auf nicht-klassische Logiken einfach

– Gegenbeispiele

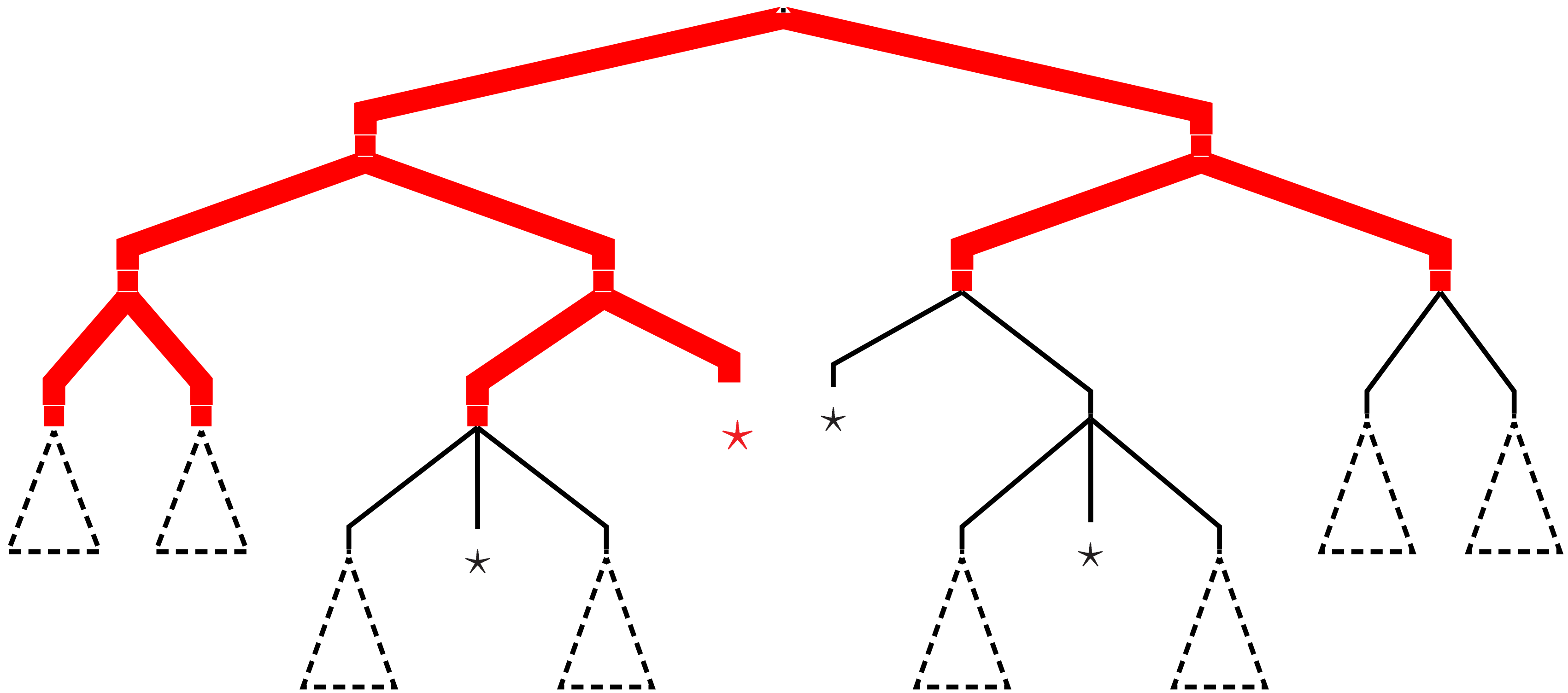
– Schwach zielorientiert

– Kürzere Beweise

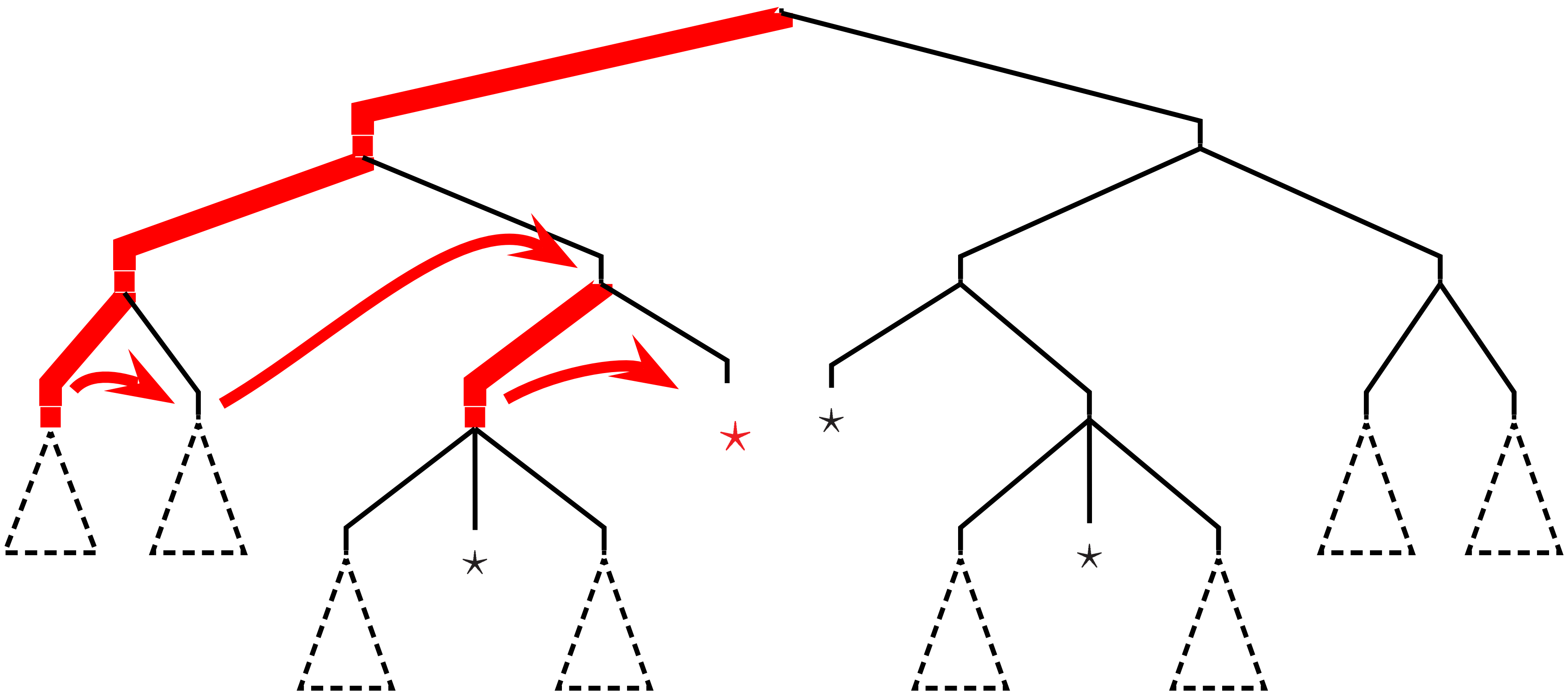
– Größerer lokaler Suchraum

Vergleich verschiedener Suchstrategien

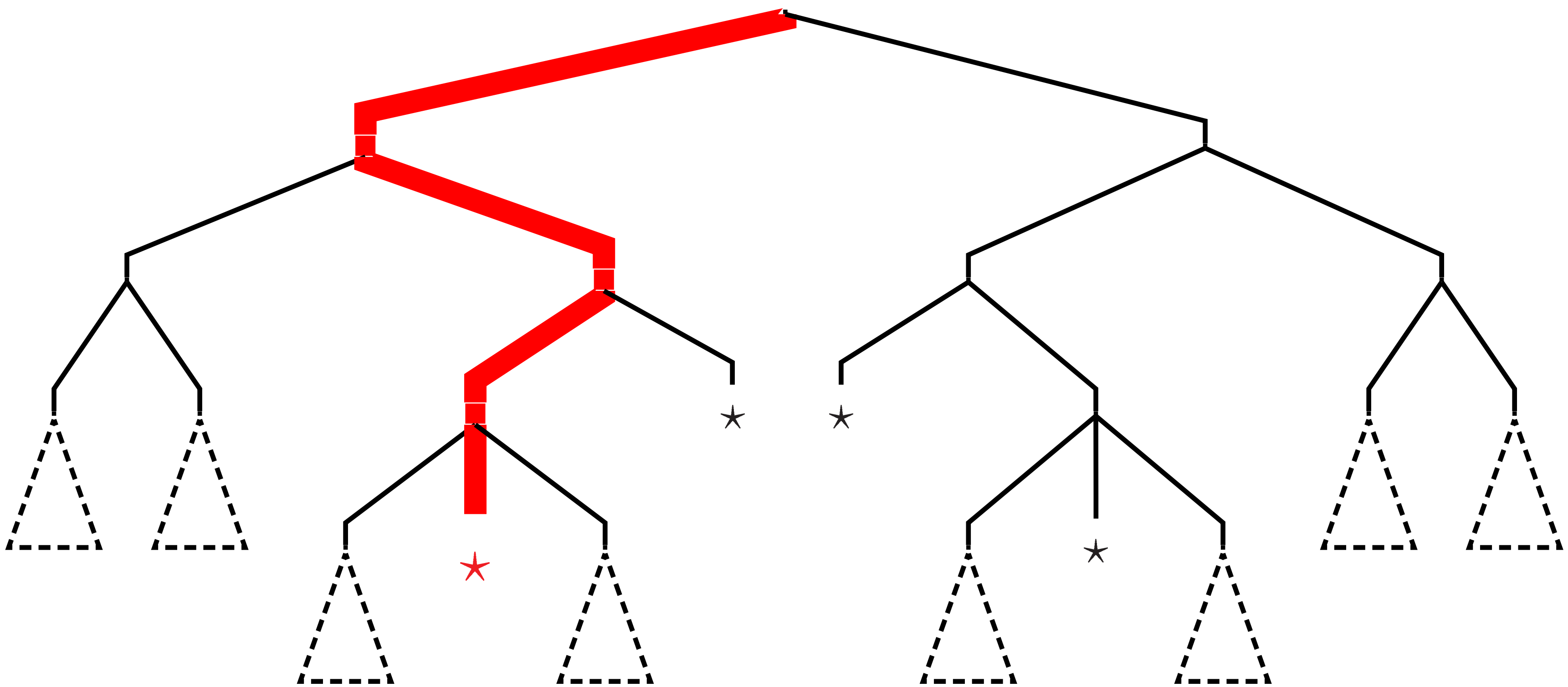
Breitensuche



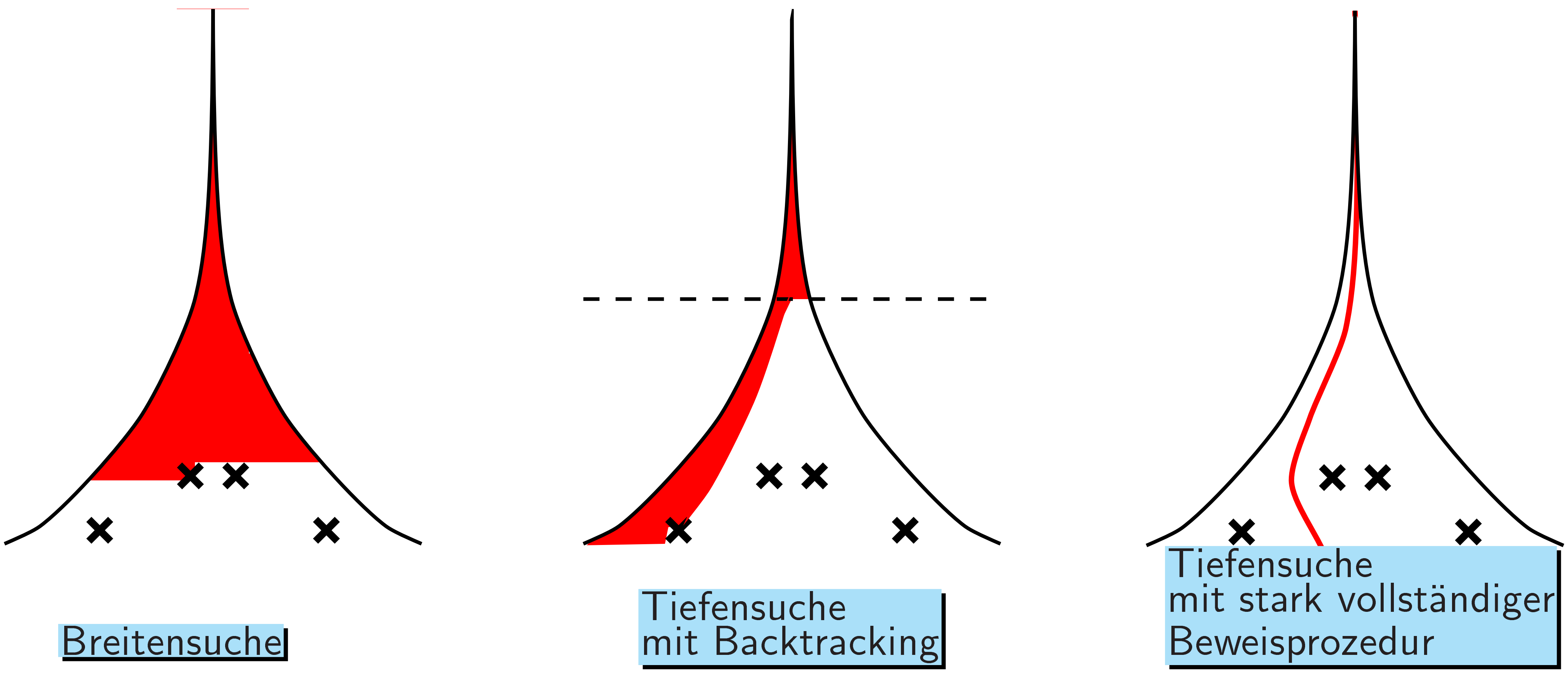
Tiefensuche mit Backtracking (Iterative Deepening)



Tiefensuche mit stark vollständiger Beweisprozedur



Suchstrategien: Vergleich



Testsuiten, Benchmarks

- akademisch, realitätsfern
- heterogen, uninterpretiert, syntaktisch
- minimal unerfüllbar, irredundant
- nur allgemeingültige Formeln
- nur klassische Logik, überwiegend SNF

TPTP (2.4.1, Mai 2001)

Sammlung von mehr als 5000 AL und PL Problemen mit/ohne Gleichheit, Horn/Nicht-Horn in SNF, sowie ca. 200 Probleme ohne Normalform.

- einfache, uniforme Syntax, Übersetzer in diverse andere Fornmate mitgeliefert;
- verschiedene Gebiete und Schwierigkeit;
- dokumentiert, Bug Fixes, Updates;
- unterscheidet Axiome, Hypothesen, Theoreme

Entwickler:
C. Suttner (TU München)
G. Sutcliffe (James Cook University, Australien)

`http://wwwjessen.informatik.tu-muenchen.de/~tptp/`

Doku: Sutcliffe et al. (1994); Suttner & Sutcliffe (1997)

Anwendungsstudien

- Reine Mathematik: Algebra, Logik, Kombinatorik
- Formale Hard- und Softwareverifikation

In Anwendungen auftretende Deduktionsprobleme oft spezifischer ...

- keine volle PL, nur endliche Modelle
- viele Probleme mit ähnlicher Struktur

... aber auch allgemeiner als bisher betrachtet:

- nicht-klassische Logik, Sorten
- Gleichheit, eingebaute Theorien

Zusätzliche Anforderungen:

- Viele Axiome, Redundanz
- Formelstruktur muß erhalten bleiben
- Simplifikation, Normalisierung, Gegenbeispielerzeugung gebraucht anstatt Widerlegung
- nicht-logische Informationen
- Benutzerinteraktion, Parametrisierung durch Benutzerwissen
- „Re-Use“ von Beweisen

Konkrete Implementierungen

SETHEO 3.3 (Januar 1996)

PL reguläre Konnektionstableaus mit inkrementeller Tiefensuche und Backtracking.

„Folding Up“ (Lemma-Generierung), Tautologielöschung, Löschung purer Klauseln; in beschränktem Maß: Subsumtion, Anti-Lemmata

Implementierung:
Klauselmengen in Maschinencode einer erweiterterten WAM (SAM) übersetzt. Übersetzer und SAM Interpretierer in C. Regularität, Anti-Lemmata, Tautologielöschung mit syntaktischen Constraints.

Plattformen: SunOS, Solaris, HP

Entwickler:
R. Letz, G. Stenz, A. Wolf
(TU München)

`http://wwwjessen.informatik.tu-muenchen.de/`

Doku: Letz et al. (1992); Goller et al. (1994)

Zur Zeit: Re-Implementierung in SCHEME: SETHEO II

$_3T^AP$ 4.0 (Juli 1997)

PL Nichtklauseltableaus mit inkrementeller Tiefensuche und Backtracking.

Schwache Konnektionen, lokale Lemmata, Löschung purer Klauseln, universelle Formeln, eingebaute Gleichheit, mehrwertige Logik, Sorten, Pruning

Implementierung: Prolog (95 %) und C

Plattformen: Sicstus Prolog 3.0.x oder Eclipse Prolog 3.5.x sowie ANSI C und Unix. Ports auf SunOS, Solaris, Linux, andere kein Problem.

Entwickler:
B. Beckert, R. Hähnle, C. Pape (Uni Karlsruhe)

`http://i12www.ira.uka.de/~threetap/`

Doku: Beckert et al. (1996a,b)

Otter 3.0.6 (Februar 1999)

PL Resolution mit Faktorisierung

Berechnung von SNF, Löschregeln, SOS, Unit-Resolution, Hyper-Resolution, UR-Resolution, gewichtete Klauseln und Literale, universelle Theorien; in beschränktem Maß: Gleichheit, eingebaute Prädikate

Implementierung: in C

Plattformen: viele Unix Dialekte, DOS, Mac.

Entwickler:
William McCune (Argonne National Labs, USA)

`http://www.mcs.anl.gov/home/mccune/ar/otter/`

Dokumentation: McCune (1994); Wos (1996)

Waldmeister 3.0.5

Beweiser für reine Gleichheits-Logik

Basiert auf Rewriting,
sehr effiziente Implementierung

Plattformen: SunOS, Solaris, Linux

Entwickler:
Thomas Hillenbrand, Andreas Jaeger, Bernd Loechner,
Arnim Buch (Universitt Kaiserslautern)

`http://agent.informatik.uni-kl.de/waldmeister`

Weitere Deduktionsformalismen

Klauselgraphresolution Kowalski (1975)

Mache die Links einer Klauselmenge explizit.

Werden nach jedem Resolutionsschritt neu berechnet und für Steuerung des nächsten Schrittes verwendet.

Links, über die schon resolviert wurde, werden gelöscht.

Nichtklauselresolution Murray (1982)

ϕ, ψ AL Formeln und $p \in \mathsf{Var}(\phi) \cap \mathsf{Var}(\psi)$.

Wenn ϕ, ψ erfüllbar, dann auch $\phi(p/\underline{\mathsf{true}}) \vee \psi(p/\underline{\mathsf{false}})$

ϕ, ψ Klauseln und p positiv in ψ: AL Resolution

Mit Vereinfachungsregeln ($\underline{\mathsf{true}} \wedge \underline{\mathsf{true}} \Rightarrow \underline{\mathsf{true}}, \ldots$):

Nichtklauselversion von Resolution, kann auf PL gehoben werden

Inverse Methode Maslov (1964)

Ähnlich zu Hyper-Resolution

Dissolution Murray & Rosenthal (1993)

ϕ AL Formel in NNF in Matrixnotation.

M komplementärer Pfad durch ϕ.

Dissolutionsregel ist Transformation D : For$0_\Sigma \to$ For0_Σ, so daß in $D(\phi)$ genau so ein Pfad M von ϕ entfernt wird.

Wenn ϕ unerfüllbar, dann $D^n(\phi) = \Box$ für $n < \omega$

Dissolution ist genuin nicht-klausale Prozedur.

$D(\phi)$ i.a. nicht in KNF, selbst wenn ϕ es war.

Es gibt Version für PL.

Konsolution Eder (1991)

Verallgemeinerung von Resolution und Konnektionsmethode vor allem für theoretische Untersuchungen.

BDD (Binary Decision Diagram) Bryant (1986)

`if-then-else` Normalform für AL Formeln:

$\models p \land q$ gdw
$\models$ if p then (if q then <u>true</u> else <u>false</u>) else <u>false</u>

Eng verwandt zu Tableaus mit Lemmata.
Ordnung auf Σ, vereinfacht, DAG.
Bedingt auf PL hebbar Posegga (1993).

Davis-Putnam-Resolution Davis & Putnam (1960)

BDDs für KNF bzw.:

Fallunterscheidung und Vereinfachung durch Purity und Unit-Resolution.

Hyperlinking Plaisted & Lee (1990)

Iterative Berechnung von Instanzen einer Klauselmenge bis AL Widerlegung gefunden.

Automatisches Beweisen in Nicht-klassischen Logiken und Erweiterungen der PL

Sortierte Logik: „typisierte Prädikatenlogik“

HOL: Quantifikation über Funktions-/Prädikatensymbole

Mehrwertige Logik:
weitere Wahrheitswerte wie $\{\underline{\text{false}}, \underline{\text{undefined}}, \underline{\text{true}}\}$

Modal- und Temporallogik:
$\phi \rightarrow \diamond\psi$ wenn jetzt ϕ gilt, dann gilt irgendwann ψ

Nicht-Monotone Logik:
$\Phi \models \psi$, aber nicht $\Phi \cup \Theta \models \psi$

Nicht-deduktive Methoden

Algebraische Methoden

AL Formeln ϕ und ψ logisch äquivalent gdw ϕ und ψ dasselbe Element in Boolescher Algebra repräsentieren

Erweiterbar auf diverse nicht-klassische Methoden

Lineare Optimierung Hooker (1988)

$\{p_1, \ldots, p_i, \neg p_{i+1}, \ldots, \neg p_n\}$

ist erfüllbar gdw die lineare Ungleichung

$p_1 + \cdots + p_i + (1 - p_{i+1}) + \cdots + (1 - p_n) \geq 1$

über $\{0, 1\}$ lösbar ist.

Reduktion von AL Erfüllbarkeit auf Lösbarkeit von linearen, ganzzahligen Optimierungsproblemen (engl.: *integer programming*)

Evolutionäre Algorithmen und Verwandte
Gent & Walsh (1993); Selman et al. (1992)

Gütefunktion f für AL Interpretationen definiert, z.B.:

$f(I, \phi) = \frac{1}{2^{u_I}}$, wobei $u_I = |\{C \in \phi \mid \mathsf{val}_I(C) = \underline{\mathsf{false}}\}|$

Falls ϕ erfüllbar, finden evolutionäre Algorithmen schnell eine erfüllende Interpretation

Taktisches Theorembeweisen

Taktik ist eine Makroregel zusammen mit einer Konstruktion, wie ein Beweis dafür (mittels einfacheren Regeln) zu finden ist.

Evtl. Benutzerinteraktion erforderlich.

Komplexe Beweisaufgaben (wie SW-Verifikation) können nicht durch blinde Suche gelöst werden.

Entwurf geschickter Taktiken ist anspruchsvolle Aufgabe.

Zusammenspiel von automatischem und interaktivem Beweisen.

Spezialbeweiser

Geometrie.

Fragmente der Arithmetik wie **Presburger Arithmetik:**

PL mit Signatur

$\langle\{0/0, s/1\}, \{plus/2, \doteq /2\}\rangle$

über der Theorie der natürlichen Zahlen ist entscheidbar.