Korrektheit d. Resolution in Prädikatenlogik

Satz

Sei R die Resolvente zweier Klauseln K_1 und K_2. Sei für eine Klausel K der **Allabschluss** $\forall K$ die Formel $\forall x_1. \ldots \forall x_n.K$, wobei $x_1, \ldots, x_n$ alle in K vorkommenden Variablen sind.

Dann gilt: $\forall R$ ist eine Folgerung aus $\forall K_1 \wedge \forall K_2$.

Beweisidee: Analog dem Beweis für die aussagenlogische Resolution: Ein Modell für $\forall K_1 \wedge \forall K_2$ muss nach Streichung der durch Resolution wegfallenden Literale mindestens einen der Allabschlüsse der Restklauseln $\forall K'_1$ oder $\forall K'_2$ erfüllen, und damit auch $\forall R$.

Folgerung

Insbesondere gilt: Ist $\square \in \mathrm{Res}^*(S)$, dann ist S inkonsistent.

Vollständigkeit der Resolution: Beweisgang

Jede Menge $\mathcal{F}$ von Formeln ist als Klauselmenge S darstellbar.
Sei S inkonsistent.

Satz von Herbrand

Endliche Menge S' von Grundklauseln ist inkonsistent.

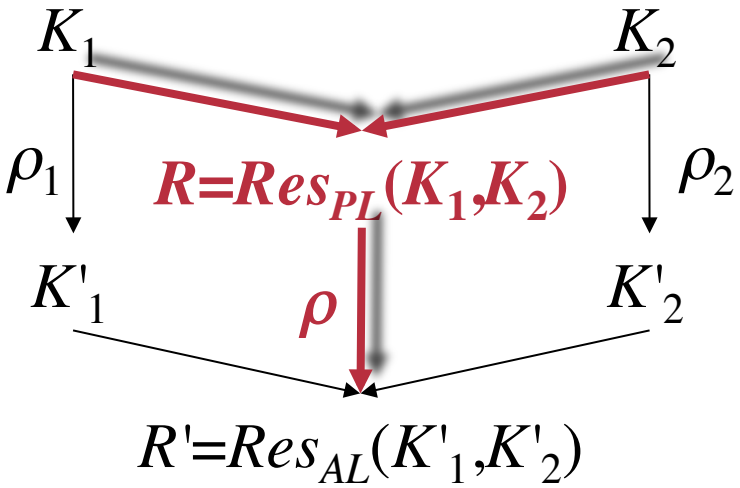
Resolutionssatz der AL

AL-Resolution leitet $\square$ aus S' ab.

„Lifting-Lemma“: noch zeigen!

Es gibt eine Widerlegung von S mit Resolution in Präd.-Logik

Das Lifting Lemma

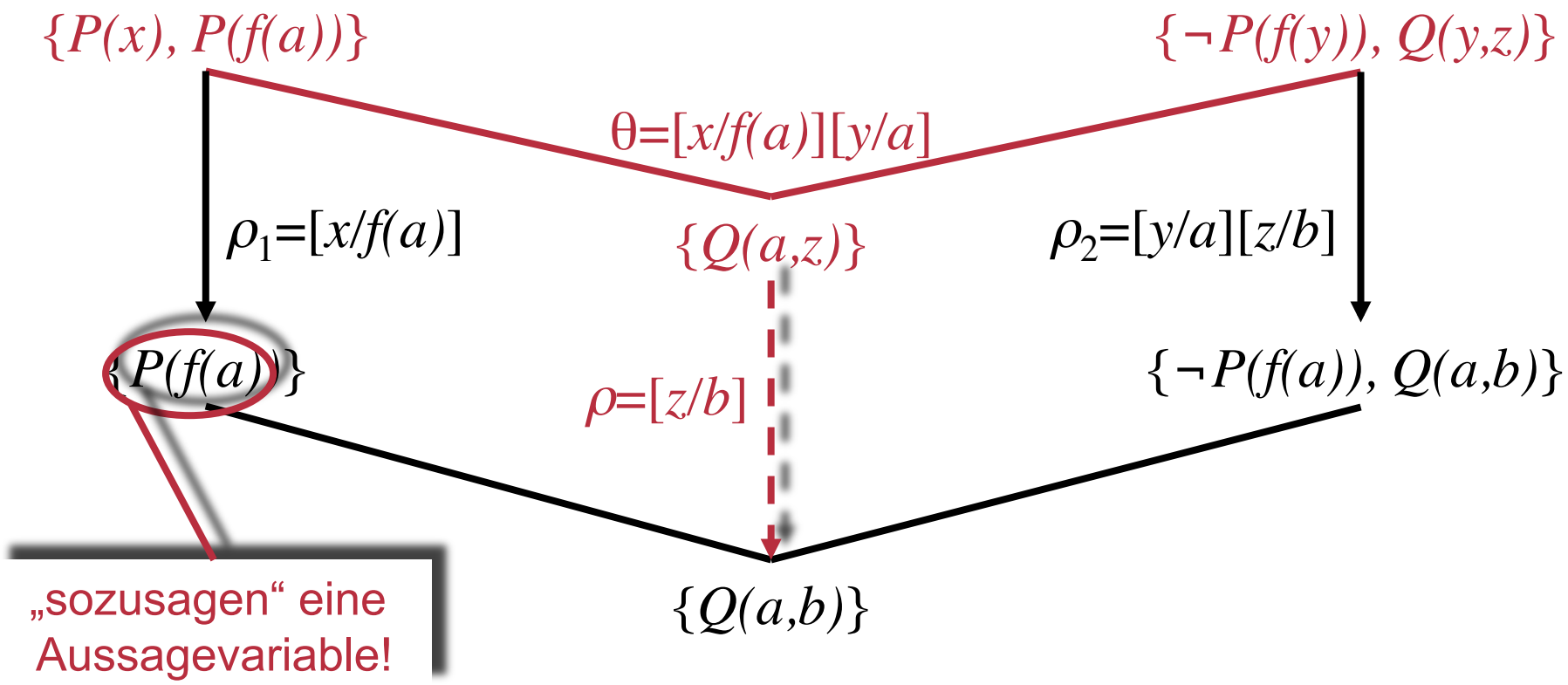


Seien K_1 und K_2 Klauseln mit disjunkten Variablen;
seien $K'_1=K_1\rho_1$ und $K'_2=K_2\rho_2$ beliebige Grundinstanzen davon, die aussagenlogisch resolvierbar sind mit Resolvente R'.
Dann gibt es eine prädikatenlogische Resolvente R von K_1 und K_2 und Substitution ρ, sodass $R'=R\rho$ ist.

Beweisidee: Verwende als Unifikator in der PL-Resolution diejenigen Substitutionen aus ρ_1, ρ_2, welche die entsprechenden Literale aus K_1 und K_2 unifizieren. Diese muss es geben.

Beispiel für Lifting-Lemma-Konstruktion

Prädikatenlogik-Ebene **rot**, Aussagenlogik schwarz



Vollständigkeits- und Resolutionssatz für PL

Vollständigkeitssatz

Sei $\mathcal{F}$ eine inkonsistente Formel in PL in Skolemform mit quantorenfreiem Teil $\mathcal{F}^*$ in KNF. Dann ist $\square \in \text{Res}^*(\mathcal{F}^*)$ für prädikatenlogische Resolution.

Beweisgang: s. vorvorletzte Folie.

Resolutionssatz der Prädikatenlogik

Sei $\mathcal{F}$ eine Formel mit Skolemform mit quantorenfreiem Teil $\mathcal{F}^*$ in KNF. $\mathcal{F}$ ist inkonsistent, gdw. $\square \in \text{Res}^*(\mathcal{F}^*)$.

Beweis: Zusammenfassung von Korrektheit und Vollständigkeit.

Bemerkung zur Faktorisierung

Resolution ohne Faktorisierung ist unvollständig!

Beispiel

1. $\{\neg P(x), \neg P(y)\}$
2. $\{P(u), P(v)\}$

Unterschiedliche Formulierungen von Resolution nehmen Faktorisierung entweder in die Resolutionsregel mit auf (wie in der Definition oben) oder definieren sie als eigene Inferenzregel (z.B. Ertel).

Spezialisierungen der Resolution

… sind **dieselben wie in der AL!**
Prinzip: Beschränkung der Auswahlmöglichkeiten
für die Elternklauseln K_i, K_j

Insbesondere gibt es:

- Stützmengen-Resolution (*set of support*, nicht in dieser Vorlesung)
- Einsklausel/Unit-Resolution
- Input-Resolution
- SLD-Resolution

Alle Spezialisierungen „erben“ Korrektheit!

Vollständigkeit (ggf. mit Einschränkungen)
beweise jeweils mit Hilfe des Lifting Lemmas!

Maßnahmen zur Reduktion der Klauselmenge

- Wir hatten schon Tricks, die Klauselmenge um „Unnötiges“ zu bereinigen: Lösche Tautologien, lösche Doubletten
- Alle Beweiser-Programme nutzen weitere korrekte Regeln zur Verkleinerung der Klauselmenge

Pure Literal Regel

- Lösche alle Klauseln, in denen ein Literal pur vorkommt!
- korrekt in Widerlegungsbeweisen

Subsumption

- Lösche in Klauselmenge $\mathcal{F}$ alle Klauseln K, für die es eine Klausel $K' \subset K$ in $\mathcal{F}$ gibt! (K' **subsumiert** K in $\mathcal{F}$)
- korrekt in Widerlegungsbeweisen

Beispiel Resolution (1/3: Formalisierung)

Problembeschreibung

Prämisse: Auf verkaufte Ware erzielt man einen Gewinn.

Folgere: Wenn ich keinen Gewinn erzielt habe,
dann habe ich keine Ware verkauft.

Eine(!) mögliche Formalisierung

$V(x,y)$: x verkauft y; $W(x)$: x ist Ware; $G(x)$: x ist Gewinn;
$E(x,y)$: x erzielt y ; i: ich

Prämisse: $\forall x.[\ \exists y.(V(x,y) \wedge W(y)) \Rightarrow \exists z.(G(z) \wedge E(x,z))]$

Folgerung: $\neg \exists z.(E(i,z) \wedge G(z)) \Rightarrow \forall y.(V(i,y) \Rightarrow \neg W(y))$

Beispiel Resolution (2/3: Klauselform)

Prämisse: $\forall x.[\ \exists y.(V(x,y) \wedge W(y)) \Rightarrow \exists z.(G(z) \wedge E(x,z))\ ]$
Folgerung: $\neg \exists z.(E(i,z) \wedge G(z)) \Rightarrow \forall y.(V(i,y) \Rightarrow \neg W(y))$

Formulierung in Klauselform

1. $\{\neg V(x_1,y_1), \neg W(y_1), G(g(x_1,y_1))\}$
2. $\{\neg V(x_2,y_2), \neg W(y_2), E(x_2,g(x_2,y_2))\}$
3. $\{\neg E(i,z_3), \neg G(z_3)\}$
4. $\{V(i,a)\}$
5. $\{W(a)\}$

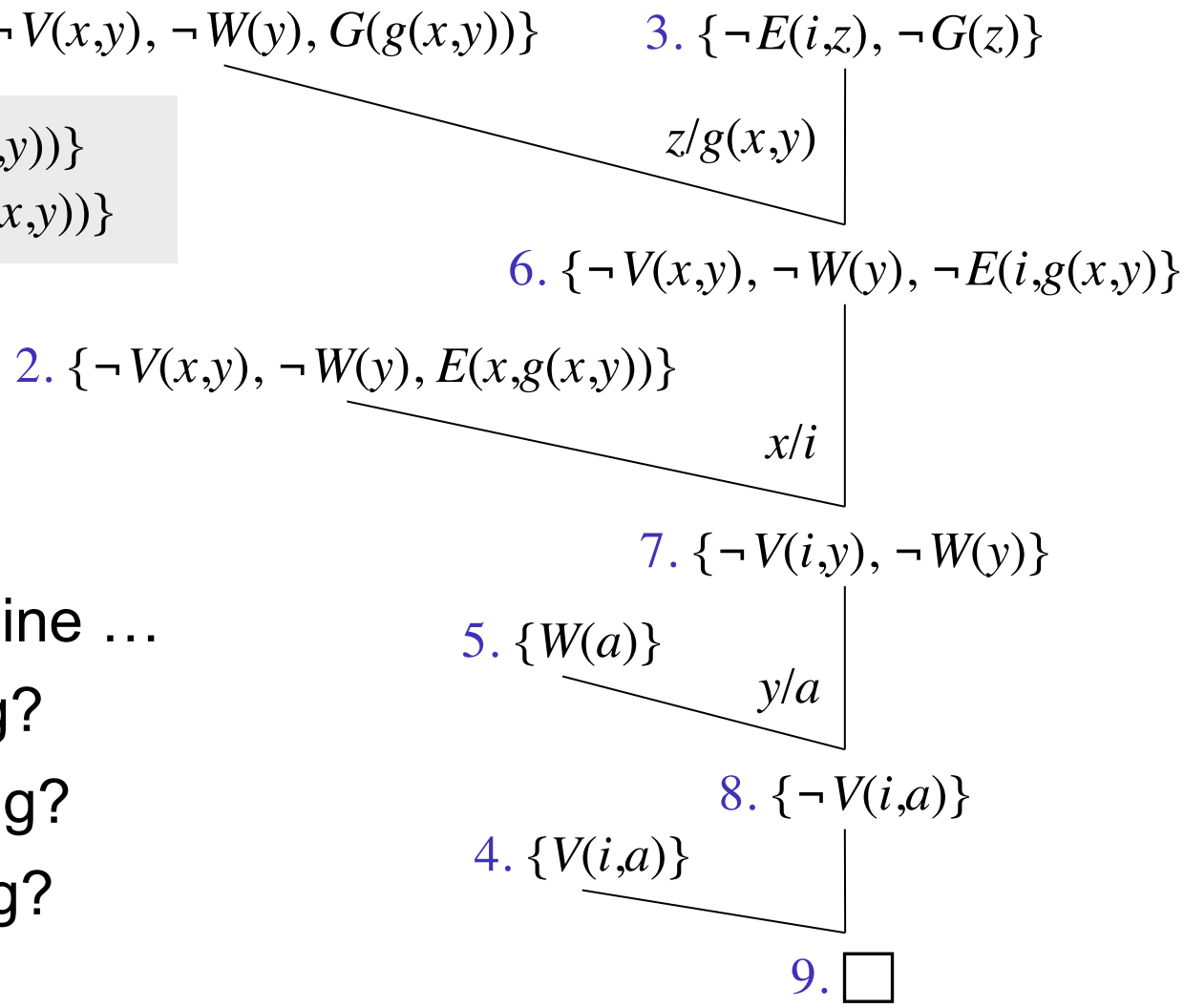
} Prämisse

} Negation der Folgerung
(Korrektheit nachprüfen!)

indiziere Variablen mit Klauselnummer,
um sie eindeutig zu machen

Beispiel Resolution (3/3: Widerlegung)

1. $\{\neg V(x,y), \neg W(y), G(g(x,y))\}$
2. $\{\neg V(x,y), \neg W(y), E(x,g(x,y))\}$
3. $\{\neg E(i,z), \neg G(z)\}$
4. $\{V(i,a)\}$
5. $\{W(a)\}$



Frage: Ist das eine …

- Unit-Ableitung?
- Input-Ableitung?
- SLD-Ableitung?

Anwendungen (Beispiele)

- Vierfarbensatz wurde 1976 mit Hilfe eines Spezial-Beweisers erstmals bewiesen
  (Beweiser behandelte umfangreiche Fallunterscheidung)
- Inferenz in Wissensbasierten Systemen
  (auch Semantic Web: Ontologien in Beschreibungslogik!)
- automatische Programmverifikation
- Korrektheitsbeweis für sicherheitskritische Software/ Software, bei der Fehler hohe Kosten verursachten
  (Kryptographie-Protokolle, Software für Raumfahrt, Software für medizinische Einsätze, …)

Anwendungsbeispiel PROLOG

Beispiel: Prozedur zur Vereinigung (ohne Doubletten!) zweier Listen (hier: nur erstes Argument auf Doubletten prüfen)

```
vng([],L) :- write(L).
vng([H|T],L) :- member(H,L), vng(T,L).
vng([H|T],L) :- not(member(H,L)), vng(T,[H|L]).
```

Beispiel:

```
?- vng([s,d,r,f,w,d,a,t,z,d,f,s],[q,w,e,r,t,z]).
[a, f, d, s, q, w, e, r, t, z]

Yes
```

- PROLOG-Antwortberechnung verwendet im Prinzip SLD-Resolution
- Außerlogische Konstrukte sind z.B. `write`, `not`, `!` („cut“)

Fazit Prädikatenlogik und Resolution

- Prädikatenlogik 1. Stufe hat deutlich höhere Ausdrucks-fähigkeit als Aussagenlogik. Sie ist als Basis für Formalismen der Wissensrepräsentation geeignet.
- Erfüllbarkeit in der Prädikatenlogik ist unentscheidbar.
- Resolution funktioniert wie in der AL, muss aber zusätzlich Unifikation und Faktorisierung verwenden.
- Resolution (+Faktorisierung) in PL ist korrekt&vollständig.
- Die Spezialisierungen von Resolution aus der AL können direkt weiter verwendet werden.
- Logikprogrammierung, CLP und Deduktionssysteme sind wichtige Anwendungen prädikatenlogischer Resolution.

2.5 Grenzen der Logik (Beispiele)

Zum Beispiel: Normalfallschließen (*default reasoning*)

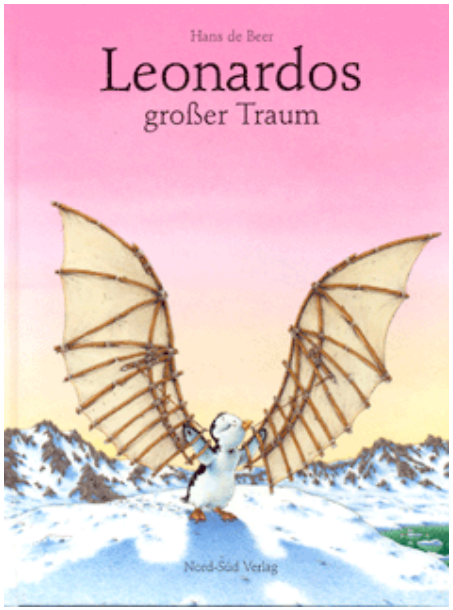
Problem klassischer Logik bei vielen „natürlichen“ Schlüssen:

- Im Normalfall gelten gewisse Konsequenzen;
- unter bekannten Ausnahmen gelten andere Konsequenzen;
- solange nicht bekannt ist, dass ein Ausnahmefall vorliegt, soll der Normalfall angenommen werden (kann zurückgenommen werden, wenn später Ausnahme-Information nachkommt!)

Beispiele

- Mangels Information übers Gegenteil nimm an, dass der Bundespräsident am Nachmittag derselbe ist wie vormittags
- Mangels Information übers Gegenteil nimm an, dass ein Vogel, von dem die Rede ist, fliegen kann

Normalfälle in klassischer Logik?



Beispiel: „Alle/Typische Vögel können fliegen“

- $\forall x.\, Vogel(x) \Rightarrow Kann\_fliegen(x)$
- modelliert nicht das „typisch“, insbesondere folgt aus $\forall x.\, Pinguin(x) \Rightarrow Vogel(x)$ dass $\forall x.\, Pinguin(x) \Rightarrow Kann\_fliegen(x)$!
- Zweiter Versuch: $\forall x.\, Vogel(x) \wedge \neg Untypisch_{Vogel}(x) \Rightarrow Kann\_fliegen(x)$ und $\forall x.\, Pinguin(x) \vee Strauß(x) \vee Brathuhn(x) \Rightarrow Untypisch_{Vogel}(x)$.
- … funktioniert, aber erfordert ausdrückliche Aufzählung aller Ausnahmen (ausgenommen die Ausnahmen der Ausnahmen, …)
- … und bei jeder Ableitung (Amsel, Drossel, Fink & Star) Nachweis nötig, dass jeweils keine Atypie vorliegt!

Vogelflug in PROLOG

```
kannfliegen(X) :- vogel(X), not(untypischV(X)).
vogel(tweety).
vogel(hansi).
vogel(X) :- pinguin(X).
untypischV(X) :- pinguin(X).
pinguin(leonardo).

?- kannfliegen(X).
X = tweety ;
X = hansi ;
No

pinguin(tweety).

?- kannfliegen(X).
X = hansi ;
No
```

Prolog verwendet *negation by finite failure*, was hier das Gesuchte leistet! Aber:

- die logische Semantik davon ist nicht offensichtlich
- das geht so mit Formeln in Hornklausellogik – wie sähe es im Allgemeinen aus?

nicht monoton! (vgl. Folie 36)

Interpretation eines Vorlesungsverzeichnisses

Beispiel aus dem Vorlesungsverzeichnis:

Projektgruppen SS08 für Masterprogramm Informatik

- PG(SS08, 6.772, humanoide_Roboter, Riedmiller)
- PG(SS08, 6.774, Social_Network_Applications, Vornberger)

Frage:

Welche Projektgruppen laufen im SS08?

Mögliche Antworten (≈Modelle) in reiner Prädikatenlogik:

- 2! (6.772, 6.774)
- 1! (6.772, 6.774 sind untersch. Namen desselben „Dings“)
- 5! (6.772, 6.774, und dann noch 3 andere, deren Namen wir nicht kennen)

↪ **Antwort**: Zwischen 1 und ∞! 

2 Annahmen zusätzlich zur Prädikatenlogik

Eindeutige Namensgebung, *Unique Names Assumption*, UNA

Unterschiedliche Namen bezeichnen unterschiedliche Individuen

Abgeschlossene Welt, *Closed World Assumption*, CWA: …

Die Information (über positive Fakten) ist abgeschlossen:
Ist eine Grundformel nicht als wahr bekannt, nimm an, dass sie falsch ist

CWA und Prolog

Im PG-Beispiel:

Eine PG, die nicht im Vorlesungsverzeichnis steht, existiert auch nicht.

Erinnerung an PROLOG:

`not(p)` ist wahr, wenn `p` nicht bewiesen werden kann.

Kein Junktor!
In PROLOG: Ein Prädikat
Allgemein: Ein außer-logisches Konstrukt

Mehr dazu Vorlesung „Wissensbasierte Systeme

Ausblick auf eine höhere Stufe

Da es Prädikatenlogik „1. Stufe“ gibt, gibt es natürlich auch Prädikatenlogik höherer Stufen.

Beispiel: $\forall R.[\ \exists y.R(0,y) \wedge \forall x.(\exists y_1.R(x,y_1) \Rightarrow \exists y_2.R(x+1, y_2)) \Rightarrow \forall x.\exists y.R(x,y)\ ]$

Peano‘sches Induktionsaxiom

- Logik der Stufe (n+1) erlaubt Äußerungen über Logiken der Stufe n (**Meta-Sprache**)
- **Für Informatiker/innen**: Meta-Sprachen braucht man z.B., um Signaturen formaler Systeme zu spezifizieren
- **Für Mathematiker/innen**: Meta-Sprachen braucht man z.B., um Theoreme über Strukturen zu formulieren