

Künstliche Intelligenz

Logisches Schließen

Dr. Christian Meilicke
Research Group Data and Web Science
Universität Mannheim

Teile der Vorlesung basieren auf einem
Foliensatz von Prof. Dr. Heiner Stuckenschmidt

Überblick



- Wie man ein Problem mit Logik beschreibt



- Wie man, gegeben eine Menge logischer Formeln, Inferenz betreiben kann
 - Formelebene vs. Modellebene
 - Beweisen mit Ableitungsregeln
 - Resolution
 - Wahrheitstabelle
 - Tableauverfahren
 - DPLL (Backtracking + X)
 - GSat (Lokale Suche)
 - WalkSat (Lokale Suche mit Zufallselement)

Begrifflichkeiten

- Achtung: Verschiedene Begriffe
 - Reasoner
 - Sat-Solver
 - Schlussfolgerungsmechanismus
 - Logisches Schlussverfahren
 - ...
- Die alle für dasselbe stehen, wenn wir uns auf Aussagenlogik beschränken

Logisches Schliessen

- $KB \models \alpha$
 - α folgt logisch aus KB
- $KB \vdash_i \alpha$
 - α kann aus KB mit der Methode i abgeleitet werden
- Korrektheit der Methode i:
 - Wenn $KB \vdash_i \alpha$ dann $KB \models \alpha$
- Vollständigkeit der methode i:
 - Wenn $KB \models \alpha$ dann $KB \vdash_i \alpha$
- Vollständig und korrekt
 - \vdash_i und \models fallen zusammen (d.h. „genau dann wenn“)

Arten von Inferenzverfahren

- Formelebene (nur syntaktische Umformungen)
 - Menge von Schlußregeln
 - Resolution
- Modellebene (explizite Behandlung der Semantik)
 - Wahrheitstabelle = Auflisten aller Interpretationen
 - Tableauverfahren = Zielgerichtete Modellkonstruktion
 - DPLL = Backtracking mit speziellen Regeln
 - GSAT = Lokale Suche
 - WalkSat = Randomisierte Lokale Suche

Schließen auf Formelebene

- Ableitungsregeln auf Grundlage der Formelstruktur:
 - Wenn $\alpha \rightarrow \beta$ und α dann β (modus ponens)
 - Wenn $\alpha \wedge \beta$ dann α (\wedge -elimination)
 - Wenn $\alpha \leftrightarrow \beta$ dann $\alpha \rightarrow \beta$ und $\beta \rightarrow \alpha$ (\leftrightarrow elimination)
 - Wenn $\neg\alpha \wedge \neg\beta$ dann $\neg(\alpha \vee \beta)$ (de Morgan)
 - ...
- Ein Beweis ist eine Folge von Ableitungen
- Die einzelnen Ableitungsregeln werden auf der Modellebene begründet

Wumpus-Welt

)))
	Luftzug)))))) 
Luftzug		Luftzug)))
	Luftzug		

Konkretes Beispiel

- Aussagen:
 - $P_{i,j}$: Falle (Pit) in Feld $[i,j]$
 - $B_{i,j}$: Luftzug (Breeze) in Feld $[i,j]$
 - Spielregeln:
 - $R_1 : \neg P_{1,1}$
 - $R_2 : B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$
 - $R_3 : B_{1,2} \leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
 - ...
 - Beobachtungen:
 - $R_4 : \neg B_{1,1}$
 - $R_5 : B_{2,1}$
 - ...
 - Frage:
 - $KB \models \neg P_{1,2}$
- Aussagen:
 - $P_{i,j}$: Falle (Pit) in Feld $[i,j]$
 - $B_{i,j}$: Luftzug (Breeze) in Feld $[i,j]$
 - Spielregeln:
 - $R_1 : \neg P_{1,1}$
 - $R_2 : B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$
 - $R_3 : B_{1,2} \leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
 - ...
 - Beobachtungen:
 - $R_4 : \neg B_{1,1}$
 - $R_5 : B_{2,1}$
 - ...
 - Frage:
 - $KB \models \neg P_{1,2}$

Beispielbeweis

Gegeben oder durch vorherige Schritte abgeleitet:

- R1: $((P_{1,2} \vee P_{2,1}) \leftrightarrow B_{1,1})$
- R2: $\neg B_{1,1}$

Beweis, dass $\neg P_{1,2}$ gilt:

\leftrightarrow Elimination auf R1

- R3: $(B_{1,1} \rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \rightarrow B_{1,1})$

\wedge Elimination auf R3

- R4: $(P_{1,2} \vee P_{2,1}) \rightarrow B_{1,1}$

Contraposition von R4

- R5: $\neg B_{1,1} \rightarrow \neg(P_{1,2} \vee P_{2,1})$

Modus Ponens auf R5 und R2

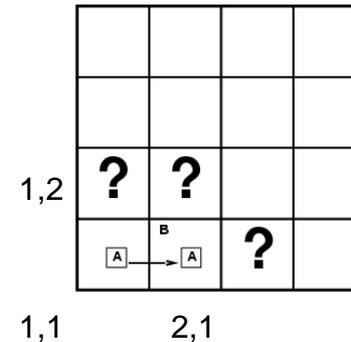
- R6: $\neg(P_{1,2} \vee P_{2,1})$

De Morgansches Gesetz auf R6

- R7: $\neg P_{1,2} \wedge \neg P_{2,1}$

\wedge Elimination auf R7

- R8: $\neg P_{1,2}$



Welche Regel wende ich wann an?

Sind die Regeln vollständig?

Wann höre ich auf (wenn ich das zu beweisende noch nicht gefunden habe)?

Normalformen

- Effizientes Schließen auf Formelebene benötigt einheitliche Strukturen auf denen Regeln arbeiten können
- Zur Erinnerung: Konjunktive Normalform (KNF):
 - Formeln sind Konjunktion von Disjunktionen
 - $(l_{1,1} \vee \dots \vee l_{1,k}) \wedge \dots \wedge (l_{n,1} \vee \dots \vee l_{n,k})$
 - Disjunktionen enthalten Aussagen oder deren Negationen (Literale)
 - Die Disjunktionen nennt man auch Klauseln
 - Jede Formel kann in konjunktive Normalform umgeformt werden

Konvertierung in KNF I

Ablesen aus der Wahrheitstabelle:

a	b	α	
0	0	0	ergibt eine klausel
0	1	1	
1	0	0	ergibt eine klausel
1	1	1	

$$(a \vee b) \wedge (\neg a \vee b)$$

Achtung: Geht natürlich nicht für die ganze KB,
wohl aber für ihre Bestandteile

Konvertierung in KNF II

- Anstatt die gesamte Wahrheitstabelle zu erstellen, verwendet man Äquivalenzumformungen für die einzelnen Formeln in der KN, z.B.

$$(1) \quad \neg(\alpha \wedge \beta) \Leftrightarrow \neg\alpha \vee \neg\beta \quad (\text{de Morgan})$$

$$(2) \quad \neg(\alpha \vee \beta) \Leftrightarrow \neg\alpha \wedge \neg\beta \quad (\text{de Morgan})$$

$$(3) \quad \alpha \rightarrow \beta \Leftrightarrow \neg\alpha \vee \beta$$

$$(4) \quad \alpha \leftrightarrow \beta \Leftrightarrow (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$

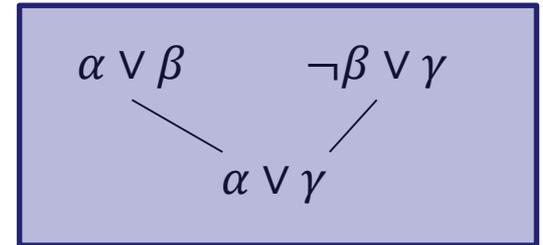
$$(5) \quad \neg\neg\alpha \Leftrightarrow \alpha$$

Resolution

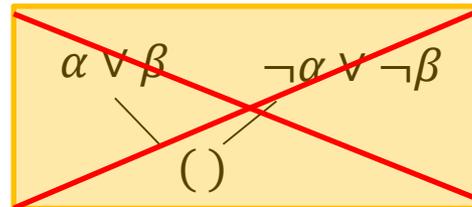
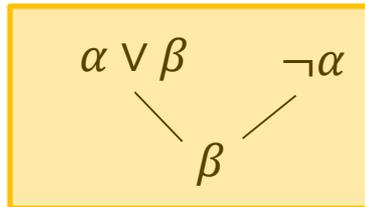
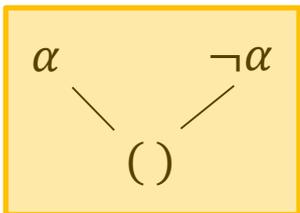
- Umformung in konjunktive Normalform (KNF)
- Sich widersprechende Literale in den Klauseln werden eliminiert
 - Terminiert immer (irgendwann) für Aussagenlogik
 - Prädikatenlogik: Terminiert i.A. nur im Fall unerfüllbarer Formeln (Semi-Entscheidbarkeit)
- Wird die leere Klausel abgeleitet, dann ist die KNF nicht erfüllbar
- Ist die KNF nicht erfüllbar, dann wird (irgendwann) die leere Klausel abgeleitet

Schlussregel

- $(\alpha \vee \beta) \wedge (\neg\beta \vee \gamma) \models \alpha \vee \gamma$
 - Eventuell einfacher nachzuvollziehen:
 - ❖ $(\neg\alpha \rightarrow \beta) \wedge (\beta \rightarrow \gamma) \models \neg\alpha \rightarrow \gamma$



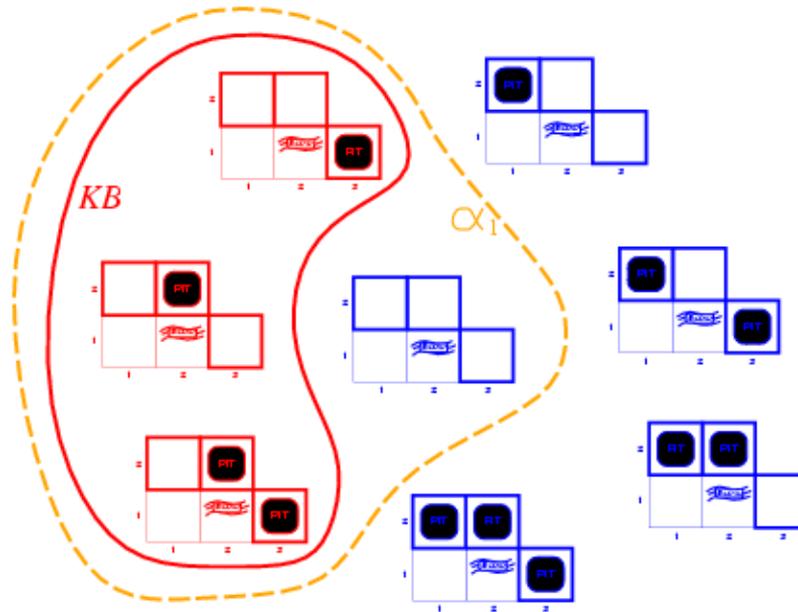
- Im allgemeinen:
 - Bilde die Vereinigung der Literale beider Formeln und streiche ein (!) Literal, das in der einen Formel positiv und in der anderen negativ vorkommt
 - Wende das Verfahren wiederholt an (auch über Ebenen hinweg)
 - Ergibt das Verfahren die leere Klausel, so ist die KNF nicht erfüllbar
- Sonderfälle:



Beweisen vs. Resolution

- Formel in beliebiger Form
 - Vollständiges Beweisverfahren nur mit vielen verschiedenen Schlußregeln möglich
- Ist eine Formel in einer bestimmten Form (KNF), dann reicht eine kleine Menge von Schlußregeln
 - Zum Beispiel Resolution: Eine Schlußregel!
 - Vorteil: Schnelles Überprüfen der Anwendbarkeit durch Indexstrukturen auf einheitlichen Formen

Beweis durch Widerspruch



$KB \models \alpha_1$ genau dann wenn $KB \wedge \neg \alpha_1$ unerfüllbar

Resolutionsbeweis

- Beweis als Suche (**an der Tafel**)
 - Zustand = Menge von Klauseln (Wissensbasis)
 - Suchschritt = Anwenden der Resolutionsregel auf zwei beliebige passende Disjunktionen
 - Zieltest = leere Klausel abgeleitet

$$(a \vee \neg b) \quad (\neg b \vee c) \quad (\neg a \vee c) \quad (\neg c \vee \neg d) \quad (\neg c \vee d)$$

Resolutionsbeweis

- Effiziente Implementierung etwas komplizierter
 - Welche Klauseln versuche ich zu resolvieren?
 - Kann ich bestimmte Kombinationen ausschließen?
 - Wann stoppe ich?
- Man kann Ordnungen über Literale definieren und diese ausnutzen
- Indexstrukturen, um zu entscheiden, welche Kombinationen zu neuen Ableitungen führen
- Behandeln wir nicht in der Vorlesung
 - Notwendig um Reasoner zu bauen, der Resolution verwendet

Überblick

- ✓ • Wie man ein Problem mit Logik beschreibt
- Wie man, gegeben eine Menge logischer Formeln, Inferenz betreiben kann
 - ✓ – Formelebene vs. Modellebene
 - ✓ – Beweisen mit Ableitungsregeln
 - ✓ – Resolution
- Wahrheitstabelle
- Tableauverfahren
- DPLL (Backtracking + X)
- GSat (Lokale Suche)
- WalkSat (Lokale Suche mit Zufallselement)



Schließen auf Modellebene

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
<i>false</i>	<i>true</i>							
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>
\vdots	\vdots							
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<u><i>true</i></u>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<u><i>true</i></u>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<u><i>true</i></u>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>
\vdots	\vdots							
<i>true</i>	<i>false</i>	<i>false</i>						

Tabelle hat 2^n Zeilen!

Konstruktion von Modellen

- Tableaux Methode:
 - Top-Down Konstruktion eines Modells auf Grundlage der Formelstruktur
 - Formen in “negation normal form”: später genauer
 - Vorteil: Nicht alle Modelle müssen aufgezählt werden
- Beweisprinzip:
 - Kann kein Modell konstruiert werden ist die Formel nicht erfüllbar
 - Beweis durch Widerspruch

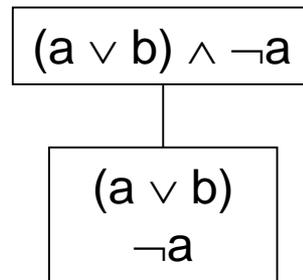
Modellkonstruktion durch Tableaux

- Beispiel:

$$(a \vee b) \wedge \neg a$$

Modellkonstruktion durch Tableaux

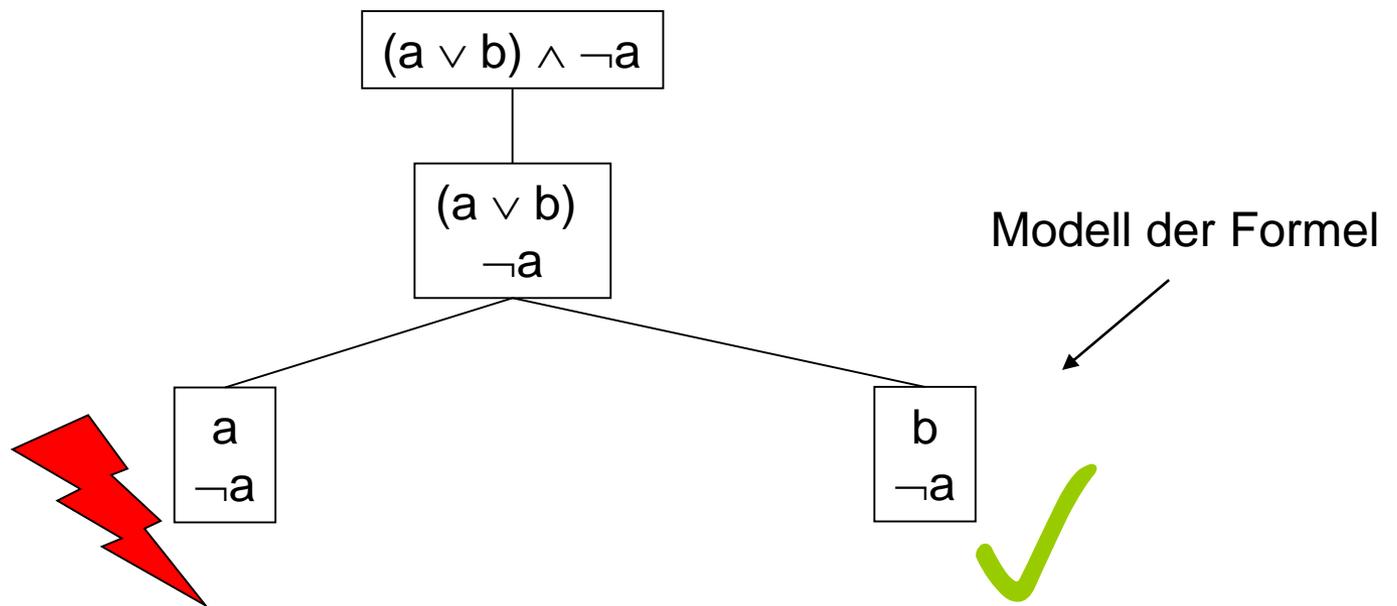
- Beispiel:



Beide Teilformeln der Konjunktion müssen erfüllt sein

Modellkonstruktion durch Tableaux

- Beispiel (erfüllbar):



Eine der Teilformeln der Disjunktion muss erfüllt sein

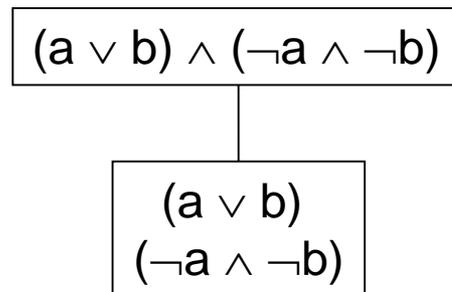
Modellkonstruktion durch Tableaux

- Beispiel:

$$(a \vee b) \wedge (\neg a \wedge \neg b)$$

Modellkonstruktion durch Tableaux

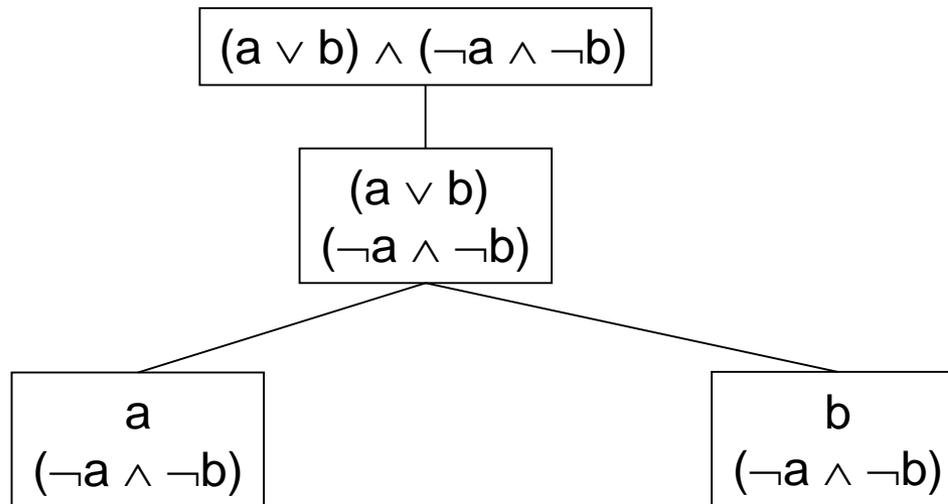
- Beispiel:



Beide Teilformeln der Konjunktion müssen erfüllt sein

Modellkonstruktion durch Tableaux

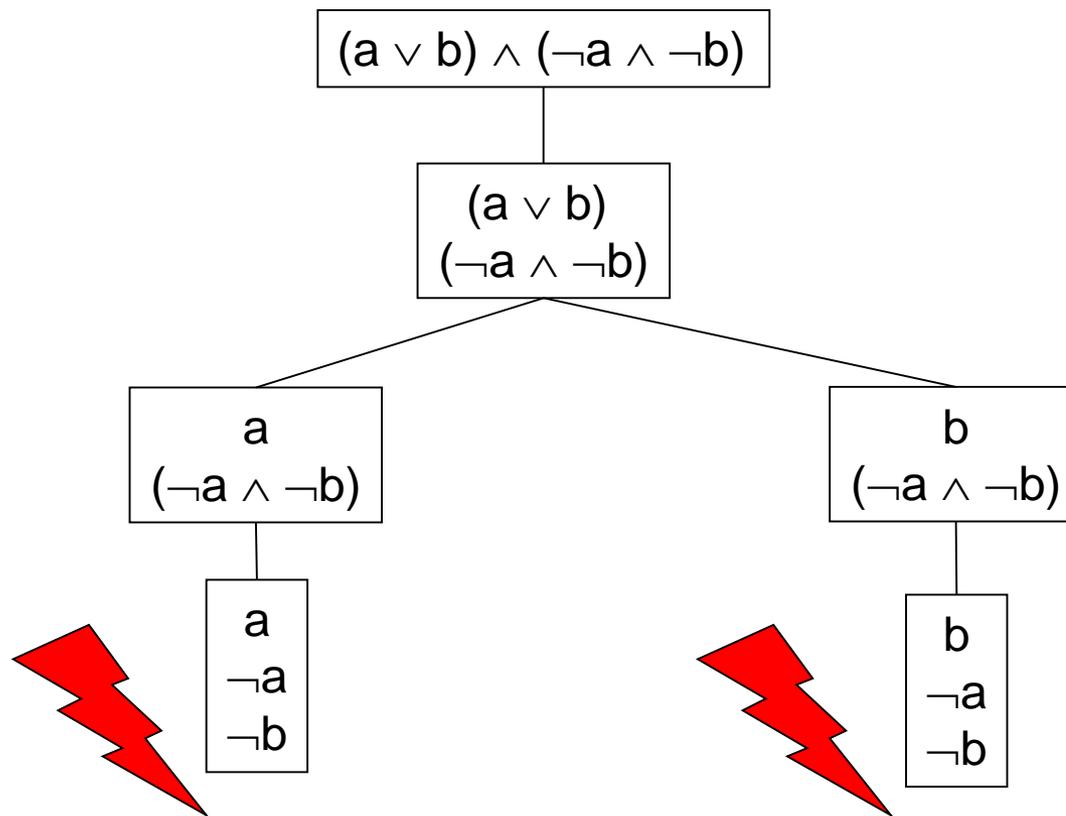
- Beispiel:



Eine der Teilformeln der Disjunktion muss erfüllt sein

Modellkonstruktion durch Tableaux

- Beispiel (unerfüllbar):



Beweisregeln

- Der Tableaux Beweis basiert auf drei einfachen Regeln:
 - Eine Konjunktion ist erfüllbar, wenn alle Teilformeln erfüllbar sind
 - Eine Disjunktion ist erfüllbar, wenn eine der Teilformeln erfüllbar ist
 - Ein Blatt in dem ein Literal und dessen Negation vorkommen ist unerfüllbar
 - Andernfalls ist das Blatt erfüllbar
- Das ist alles!
- Aber: Die Eingabe muss in Negationsnormalform vorliegen, d.h.
 - nur Disjunktion und Konjunktion dürfen verwendet werden
 - Negationen kommen nur in Literalen vor

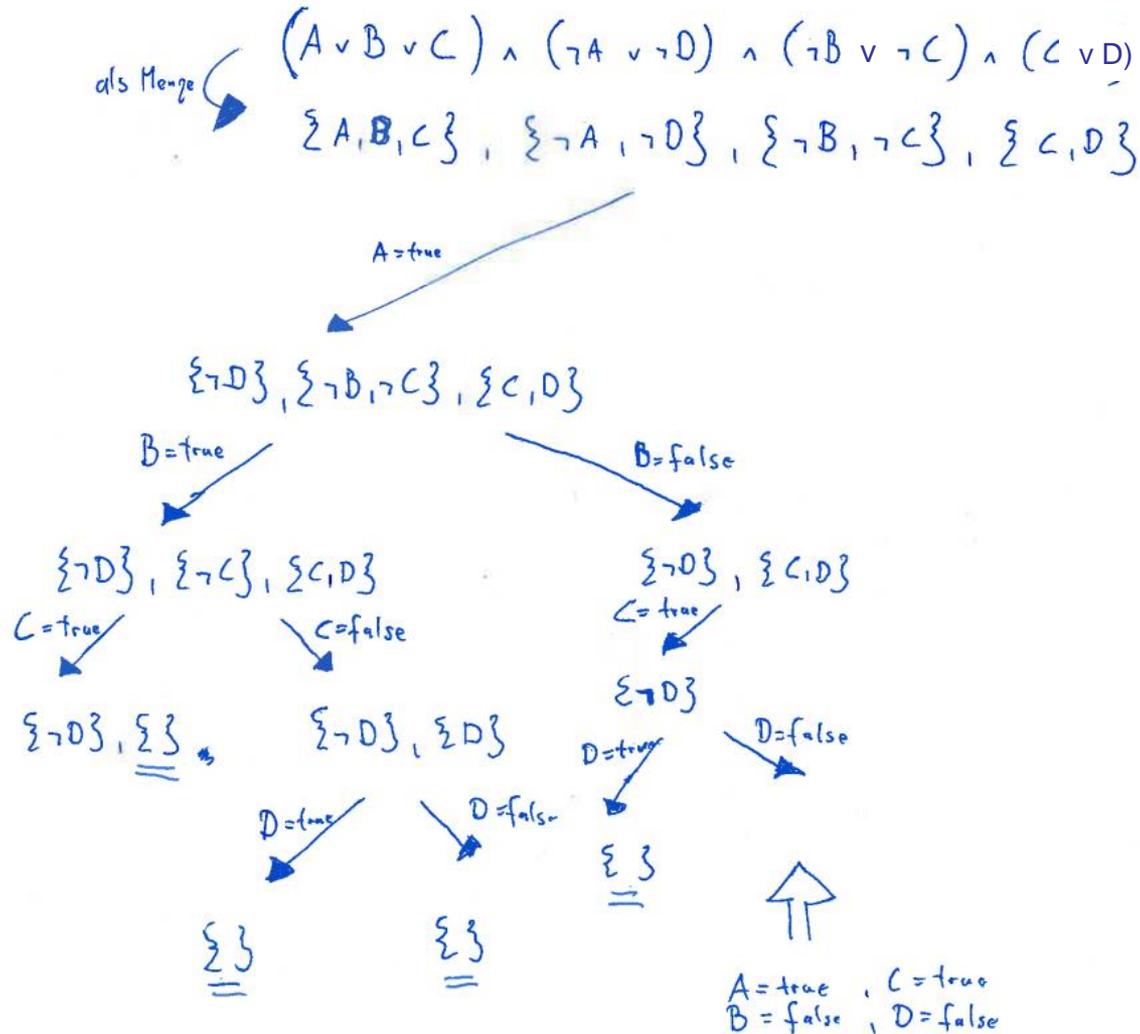
Nochmal zum Mitschreiben

- In der Regel wendet man das Tableauverfahren an, um herauszufinden, ob eine Formel unerfüllbar ist
 - Zum Beispiel im Rahmen des indirekten Verfahrens
- Sobald man einen Blattknoten ohne Widersprüche findet, kann man das Verfahren beenden
 - Man kann ein Modell aus dem Blattknoten ablesen
 - Die Formel ist erfüllbar
- Ein oder mehrere Blattknoten, die Widersprüche enthalten, erlauben keine Schlußfolgerung zur Unerfüllbarkeit
 - Man muss weiter machen, um festzustellen, ob alle Blattknoten Widersprüche enthalten
 - Man kann nur schließen, dass die Formel keine Tautologie ist
- Alle Blattknoten enthalten Widersprüche => Formel unerfüllbar

Davis-Putnam Algorithmus

- Backtracking + Heuristiken zum Forward Checking:
- Backtracking:
 - Setze ein Atom α auf true (bzw. false) und vereinfache die resultierende Formel
 - Entferne alle Klauseln in denen α (bzw. $\neg\alpha$) vorkommt, da diese durch die Setzung erfüllt sind
 - Entferne aus allen Klauseln $\neg\alpha$ (bzw. α), da die anderen Literale dafür sorgen müssen, dass die jeweilige Klausel wahr wird
- Leere Klausel erzeugt, dann gehe zurück, ansonsten mach mit der vereinfachten Formel weiter

Reines Backtracking



Davis-Putnam Algorithmus

- Wende die folgenden Heuristiken an, wann immer möglich:
 - Tautology Checking
 - Beispiel ... $(A \vee \neg A)$...
 - Entfernen der Tautologie
 - Unit Propagation
 - Beispiel: $A \wedge (B \vee \neg C) \wedge (\neg B \vee A) \wedge (C \vee \neg A)$
 - Setze $A = \text{true}$ und vereinfache die Formel zu $(B \vee \neg C) \wedge C$
 - Pure Literal Deletion
 - Beispiel: $(A \vee \neg B) \wedge (\neg B \vee D) \wedge (C \vee A)$
 - Setze $A = \text{true}$ (bzw. $B = \text{false}$) und vereinfache die Formel

DPLL Beispiel

$\{A, B, C\}$, $\{ \neg A, \neg D \}$, $\{ \neg B, \neg C \}$, $\{ C, D \}$

$A = \text{true}$

$\{ \neg D \}$, $\{ \neg B, \neg C \}$, $\{ C, D \}$

Unit Propagation $D = \text{false}$

$\{ \neg B, \neg C \}$, $\{ C \}$

Unit Propagation $C = \text{true}$

$\{ \neg B \}$

Unit Propagation $B = \text{false}$

✓

$A = \text{true}$
 $B = \text{false}$
 $C = \text{true}$
 $D = \text{false}$

Schwierige Probleme

- Probleme mit einem bestimmten Verhältnis zwischen Klauseln und Symbolen sind schwer und ab einer gewissen Größe für DPLL nicht mehr lösbar
- Idee: Lokale Suche
 - Nach einem Vergleich von 1993 kommt die vollständige Suche bis max 400 Literale, lokale Suche bis ca. 2000 Literale
 - Aktuelle Solver sind in der Lage Probleme mit bis zu 1 Million Variablen zu lösen
- Wie funktionierte noch mal lokale Suche und wie könnte man das auf Aussagenlogik übertragen ?

GSAT

```
for i = 1 to max-tries
  T = random assignment      (T = eine Interpretation)
  loop
    If satisfied then return T
    best_flip = null
    best_flip_score = satisfied_clauses(T)
    for each variable v
      T' = T with v flipped
      if (satisfied_clauses(T') > best_flip_score)
        best_flip = v
        best_flip_score = satisfied_clauses(T')
    if best_flip != null
      T = T' with best_flip flipped
    else
      return from inner loop
  end loop
end for
return unsatisfiable
```

GSAT Beispiel

Es sei T die Belegung bei der alle Variablen auf true gesetzt sind:

$$\neg a \wedge \neg b \wedge c \wedge (a \vee \neg e) \wedge (a \vee \neg f) \wedge (\neg d \vee b \vee g \vee h)$$

Welche Variable wird geflippt?

Umfrage: a = A, b = B, c = C, d = D

... ich verrate damit schonmal dass es keine der anderen Variablen ist

GSAT Beispiel

Es sei T die Belegung bei der alle Variablen auf true gesetzt sind:

$$\neg a \wedge \neg b \wedge c \wedge (a \vee \neg e) \wedge (a \vee \neg f) \wedge (\neg d \vee b \vee g \vee h)$$

Welche Variable wird geflippt?

-	a	b	c	d	e	f	g	h
4	3	5	3	4	4	4	4	4

Eigenschaften / Probleme

- Ist die Methode korrekt und vollständig, wenn angewendet im Kontext eines indirekten Beweis?
 - Korrektheit:
 - Wenn $KB \vdash_i \alpha$ dann $KB \models \alpha$
 - Vollständigkeit:
 - Wenn $KB \models \alpha$ dann $KB \vdash_i \alpha$
- **Nicht korrekt, aber vollständig!**
- Problem von GSAT: lokale Minima
- Welche Lösungen gibt es dafür
 - Erinnerung: lokale Suche?

WalkSAT

- Randomisierte Variante von GSAT (verwendet ‚random walks‘)
 - With probability p , pick a variable occurring in some unsatisfied clause and change the truth assignment
 - With probability $1-p$ follow the GSAT strategy

Überblick

- 
- Wie man ein Problem mit Logik beschreibt
 - Wie man, gegeben eine Menge logischer Formeln, Inferenz betreiben kann
 - Formelebene vs. Modellebene
 - Beweisen mit Ableitungsregeln
 - Resolution
-
- Wahrheitstabelle
 - Tableauverfahren
 - DPLL (Backtracking + X)
 - GSat (Lokale Suche)
 - WalkSat (Lokale Suche mit Zufallselement)

Zusammenfassung

- Alle Modelle aufzuzählen ist nicht effizient, bereits bei kleinen Problemen nicht anwendbar
- Formulierung als KNF für viele Algorithmen Standardinput
- Einfache Verfahren sind Resolution und Wahrheitstafel

- Weiter kommt man mit Backtracking Algorithmus wie DPLL
- Lokale Suchverfahren bei größeren Problemen
 - Findet nicht unbedingt ein Modell, wenn es ein Modell gibt

- Algorithmen werden ständig weiterentwickelt
 - <http://www.satcompetition.org/>

Ausblick

- Nächste Woche letzte Vorlesung zum Thema Planen
- Dies wird wohl auch das Thema des Seminars
 - Themen kann ich hoffentlich bis Ende der Woche festlegen
 - Eventuell einzelne weiterführende Kapitel eines Lehrbuchs
- Fragestunde gibt es keine, aber:
 - Bitte Fragen in ILIAS stellen
 - Antwort in ILIAS als Text oder bei Bedarf Mini-Video