Logik für Informatiker

Vorlesung 4: Zweiwertige Modelle

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

27. Oktober 2017

Ein zweiter Kalkül: logische Umformung

Definition:

Äquivalenzumformung:

- (Wiederholte) Ersetzung einer (Unter-)Formel durch äquivalente Formel

Teilformeln

Eine Formel F, die als Teil einer Formel G auftritt, heißt Teilformel von G.

- F ist eine Teilformel von F
- $\left.\begin{array}{l} F = \neg G \text{ und} \\ H \text{ Teilformel von } G \end{array}\right\} \rightarrow \quad H$ Teilformel von F
- $\left.\begin{array}{l} F = F_1 \ \rho \ F_2 \\ (\text{wo } \rho \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}) \\ H \text{ Teilformel von } F_1 \text{ oder } F_2 \end{array}\right\} \rightarrow \quad H$ Teilformel von F

Substitutionstheorem

Theorem.
Seien F und G äquivalente Formeln. Sei H eine Formel mit (mindestens) einem Vorkommen der Teilformel F.

Dann ist H äquivalent zu H', wobei H' aus H hervorgeht, indem (irgend) ein Vorkommen von F in H durch G ersetzt wird.

Beispiel:

$A \vee B \equiv B \vee A$

impliziert

$(C \wedge (A \vee B)) \equiv (C \wedge (B \vee A))$

Substitutionstheorem

Theorem.
Seien F und G äquivalente Formeln. Sei H eine Formel mit (mindestens) einem Vorkommen der Teilformel F.

Dann ist H äquivalent zu H', wobei H' aus H hervorgeht, indem (irgend) ein Vorkommen von F in H durch G ersetzt wird. $p(H)$

Beweis: Strukturelle Induktion.

Induktionsbasis: Beweisen, dass $p(H)$ für alle Formeln H in $\{\bot, \top\} \cup \Pi$ gilt.

Beweis: Falls $H \in \{\bot, \top\} \cup \Pi$ und F Teilformel von H, so muss $F = H$ sein. Dann ist die Formel H', die aus H hervorgeht, indem F (= die ganze Formel H) durch G ersetzt wird, gleich G.

Aber dann: $H = F \equiv G = H'$.

Substitutionstheorem

Beweis: (Fortsetzung)
Sei H eine Formel, $H \notin \{\bot, \top\} \cup \Pi$. Sei F eine Teilformel von H.

Fall 1: $F = H$. Dann $H' = G$ (wie vorher), so $H = F \equiv G = H'$.

Fall 2: $F \neq H$.

Induktionsvoraussetzung: Annahme: $p(H')$ gilt für alle dir. Teilformeln H' von H.

Induktionsschritt: Beweis, dass $p(H)$ gilt (durch Fallunterscheidung):

Fall 2.1: $H = \neg H_1$. Da $F \neq H$, ist F eine Teilformel von H_1.

Induktionvoraussetzung: $p(H_1)$ gilt, d.h. $H_1 \equiv H_1'$, wobei H_1' aus H_1 hervorgeht, indem (irgend) ein Vorkommen von F in H_1 durch G ersetzt wird.

Da $H = \neg H_1$, ist $H' = \neg H_1'$.

Dann für alle $\mathcal{A} : \Pi \to \{0,1\}$: $\mathcal{A}(H) = \mathcal{A}(\neg H_1) = \neg \mathcal{A}(H_1) \overset{I.V.}{=} \neg \mathcal{A}(H_1') = \mathcal{A}(\neg H_1') = \mathcal{A}(H')$

Somit ist bewiesen, dass $H \equiv H'$.

SUBSTITUTIONSTHEOREM

Beweis: (Fortsetzung)
Induktionsschritt: Beweis, dass $p(H)$ gilt (durch Fallunterscheidung):

Fall 2.2: $H = H_1$ op H_2. Da $F \neq H$, ist F Teilformel von H_1 oder von H_2.

Fall 2.2.1 F ist eine Teilformel von H_1.
Induktionvoraussetzung: $p(H_1)$ gilt, d.h. $H_1 \equiv H_1'$, wobei H_1' aus H_1 hervorgeht, indem (irgend) ein Vorkommen von F in H_1 durch G ersetzt wird.

Da $H = H_1$ op H_2, und F in H_1 vorkommt, so $H' = H_1'$ op H_2.
Dann für alle $\mathcal{A} : \Pi \to \{0, 1\}$: $\mathcal{A}(H) = \mathcal{A}(H_1 \text{ op } H_2) = \mathcal{A}(H_1)$ op $\mathcal{A}(H_2) \overset{I.V.}{=} \mathcal{A}(H_1')$ op $\mathcal{A}(H_2) = \mathcal{A}(H_1' \text{ op } H_2) = \mathcal{A}(H')$.

Somit ist bewiesen, dass $H \equiv H'$.

Fall 2.2.2 F ist eine Teilformel von H_2. Analog.


Ein zweiter Kalkül: logische Umformung

Definition:

Äquivalenzumformung:

- (Wiederholte) Ersetzung einer (Unter-)Formel durch äquivalente Formel
- Anwendung des Substitutionstheorems

Ein zweiter Kalkül: logische Umformung

Theorem

Äquivalenzumformung bildet mit den aufgelisteten wichtigen Äquivalenzen einen vollständigen Kalkul:
Wenn F und G logisch äquivalent sind, kann F in G umgeformt werden.

Anwendung

Test für Erfüllbarkeit/Unerfüllbarkeit/Allgemeingültigkeit.

Beispiel

$$(P \to Q) \wedge \neg((Q \to R) \to (P \to R)).$$

Beispiel

$$(P \to Q) \wedge \neg((Q \to R) \to (P \to R))$$

$$\begin{aligned}
&\equiv (\neg P \vee Q) \wedge \neg((\neg Q \vee R) \to (\neg P \vee R)) && \text{(Elimination Implikation)}\\
&\equiv (\neg P \vee Q) \wedge \neg(\neg(\neg Q \vee R) \vee (\neg P \vee R)) && \text{(Elimination Implikation)}\\
&\equiv (\neg P \vee Q) \wedge (\neg\neg(\neg Q \vee R) \wedge \neg(\neg P \vee R)) && \text{(De Morgan's Regel,}\vee\text{)}\\
&\equiv (\neg P \vee Q) \wedge \quad ((\neg Q \vee R) \wedge (\neg\neg P \wedge \neg R)) && \text{(Doppelte Negation, De Morgan, }\vee\text{)}\\
&\equiv (\neg P \vee Q) \wedge ((\neg Q \vee R) \wedge (P \wedge \neg R)) && \text{(Doppelte Negation)}\\
&\equiv (\neg P \vee Q) \wedge ((\neg Q \wedge P \wedge \neg R) \vee (R \wedge P \wedge \neg R)) && \text{(Distributivität)}\\
&\equiv (\neg P \vee Q) \wedge ((\neg Q \wedge P \wedge \neg R) \vee (R \wedge \neg R \wedge P)) && \text{(Kommutativität)}\\
&\equiv (\neg P \vee Q) \wedge ((\neg Q \wedge P \wedge \neg R) \vee \bot) && \text{(Äquivalenzen mit }\bot\text{)}\\
&\equiv (\neg P \vee Q) \wedge (\neg Q \wedge P \wedge \neg R) && \text{(Äquivalenzen mit }\bot\text{)}\\
&\equiv (\neg P \wedge \neg Q \wedge P \wedge \neg R) \vee (Q \wedge \neg Q \wedge P \wedge \neg R) && \text{(Distributivität)}\\
&\equiv (\neg P \wedge P \wedge \neg Q \wedge \neg R) \vee (Q \wedge \neg Q \wedge P \wedge \neg R) && \text{(Kommutativität)}\\
&\equiv ((\neg P \wedge P) \wedge \neg Q \wedge \neg R) \vee ((Q \wedge \neg Q) \wedge P \wedge \neg R) && \text{(Assoziativität)}\\
&\equiv \bot \vee \bot \equiv \bot && \text{(Äquivalenzen mit }\bot\text{)}
\end{aligned}$$

KALKÜLE

- Wahrheitstabellen
- Äquivalenzumformungen
- nicht besonders effizient...
- Ziel: Kalkül(e) zur systematischen Überprüfung von Erfüllbarkeit (für Formeln und/oder Formelmengen)

Normalformen

Definition:

- Atom: aussagenlogische Variable
- Literal: Atom, oder

  Negation eines Atoms

Beispiel. Sei $\Pi = \{P, Q, R\}$.

Atome: P, Q, R

Literale: P, $\neg P$, Q, $\neg Q$, R, $\neg R$

Normalformen

Definition:

- Atom: aussagenlogische Variable
- Literal: Atom, oder

  Negation eines Atoms

Definition:

Klausel: Eine Disjunktion von Literalen

- mehrstellige Disjunktionen $(P \vee \neg Q \vee R)$, $(P \vee P \vee \neg Q)$
- einstellige Disjunktionen P
- die nullstellige Disjunktion (leere Klausel) $\bot$

Normalformen

Definition:

Konjunktive Normalform (KNF): Eine Konjunktion von Disjunktionen von Literalen, d.h., eine Konjunktion von Klauseln

mehrstellig, einstellig oder nullstellig

Beispiele:

$(P \vee \neg Q) \wedge (Q \vee \neg R \vee \neg S)$

$P \vee Q$

$P \wedge (Q \vee R)$

$P \wedge Q$

$P \wedge P$

$\top$

Normalformen

Definition:

Disjunktive Normalform (DNF): Eine Disjunktion von Konjunktionen von Literalen.

mehrstellig, einstellig oder nullstellig

Beispiele:

$(P \wedge \neg Q) \vee (Q \wedge \neg R \wedge \neg S)$

$P \wedge Q$

$P \vee (Q \wedge R)$

$P \vee Q$

$P \vee P$

$\bot$

Normalformen

Eigenschaften:

- Zu jeder aussagenlogischen Formel gibt es:
  - eine äquivalente Formel in KNF
  - eine äquivalente Formel in DNF
- Diese äquivalenten Formeln in DNF bzw. KNF sind nicht eindeutig
- Solche Formeln können aus einer Wahrheitstafel abgelesen werden
- Solche Formeln können durch Umformungen hergestellt werden

Normalformen

Eigenschaften:

- Zu jeder aussagenlogischen Formel gibt es:
  - eine äquivalente Formel in KNF
  - eine äquivalente Formel in DNF
- Diese äquivalenten Formeln in DNF bzw. KNF sind nicht eindeutig
- Solche Formeln können aus einer Wahrheitstafel abgelesen werden
- Solche Formeln können durch Umformungen hergestellt werden

Beispiel

$F: \quad (P \vee Q) \wedge ((\neg P \wedge Q) \vee R)$

P	Q	R	$(P \vee Q)$	$\neg P$	$(\neg P \wedge Q)$	$((\neg P \wedge Q) \vee R)$	F
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	0	0	0	0
1	0	1	1	0	0	1	1
1	1	0	1	0	0	0	0
1	1	1	1	0	0	1	1

Beispiel: DNF

$F: \quad (P \vee Q) \wedge ((\neg P \wedge Q) \vee R)$

P	Q	R	$(P \vee Q)$	$\neg P$	$(\neg P \wedge Q)$	$((\neg P \wedge Q) \vee R)$	F
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	0	0	0	0
1	0	1	1	0	0	1	1
1	1	0	1	0	0	0	0
1	1	1	1	0	0	1	1

Beispiel: DNF

$F:\quad (P \vee Q) \wedge ((\neg P \wedge Q) \vee R)$

P	Q	R	$(P \vee Q)$	$\neg P$	$(\neg P \wedge Q)$	$((\neg P \wedge Q) \vee R)$	F
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	0	0	0	0
1	0	1	1	0	0	1	1
1	1	0	1	0	0	0	0
1	1	1	1	0	0	1	1

DNF: $(\neg P \wedge Q \wedge \neg R) \vee (\neg P \wedge Q \wedge R) \vee (P \wedge \neg Q \wedge R) \vee (P \wedge Q \wedge R)$

Beispiel: KNF

$F: \quad (P \vee Q) \wedge ((\neg P \wedge Q) \vee R)$

P	Q	R	$(P \vee Q)$	$\neg P$	$(\neg P \wedge Q)$	$((\neg P \wedge Q) \vee R)$	F	$\neg F$
0	0	0	0	1	0	0	0	1
0	0	1	0	1	0	1	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	1	1	1	1	0
1	0	0	1	0	0	0	0	1
1	0	1	1	0	0	1	1	0
1	1	0	1	0	0	0	0	1
1	1	1	1	0	0	1	1	0

DNF für F: $(\neg P \wedge Q \wedge \neg R) \vee (\neg P \wedge Q \wedge R) \vee (P \wedge \neg Q \wedge R) \vee (P \wedge Q \wedge R)$

DNF für $\neg F$: $(\neg P \wedge \neg Q \wedge \neg R) \vee (\neg P \wedge \neg Q \wedge R) \vee (P \wedge \neg Q \wedge \neg R) \vee (P \wedge Q \wedge \neg R)$

KNF für F: $(P \vee Q \vee R) \wedge (P \vee Q \vee \neg R) \wedge (\neg P \vee Q \vee R) \wedge (\neg P \vee \neg Q \vee R)$

Normalformen

DNF für F:

$$\bigvee_{\substack{\mathcal{A}:\{P_1,\ldots,P_n\}\to\{0,1\}\\ \mathcal{A}(F)=1}} (P_1^{\mathcal{A}(P_1)} \wedge \cdots \wedge P_n^{\mathcal{A}(P_n)})$$

wobei:

$P^0 = \neg P$

$P^1 = P$

Theorem

Für alle Interpretationen $\mathcal{A}' : \{P_1, \ldots, P_n\} \to \{0, 1\}$:

$$\mathcal{A}'(F) = 1 \quad \text{gdw.} \quad \mathcal{A}'\Big(\bigvee_{\substack{\mathcal{A}:\{P_1,\ldots,P_n\}\to\{0,1\}\\ \mathcal{A}(F)=1}} (P_1^{\mathcal{A}(P_1)} \wedge \cdots \wedge P_n^{\mathcal{A}(P_n)})\Big) = 1.$$

Normalformen

DNF für F:

$$\bigvee_{\substack{\mathcal{A}:\{P_1,\ldots,P_n\}\to\{0,1\}\\ \mathcal{A}(F)=1}} (P_1^{\mathcal{A}(P_1)} \wedge \cdots \wedge P_n^{\mathcal{A}(P_n)})$$

wobei:

$P^0 = \neg P$

$P^1 = P$

KNF für F: $\neg F'$,

wobei F' die DNF von $\neg F$ ist.

KNF für F:

$$\bigwedge_{\substack{\mathcal{A}:\{P_1,\ldots,P_n\}\to\{0,1\}\\ \mathcal{A}(F)=0}} (P_1^{1-\mathcal{A}(P_1)} \vee \cdots \vee P_n^{1-\mathcal{A}(P_n)})$$

Umformung in KNF

Vier Schritte:

1. Elimination von $\leftrightarrow$

   Verwende $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

2. Elimination von $\rightarrow$

   Verwende $A \rightarrow B \equiv (\neg A \vee B)$

3. "Nach innen schieben" von $\neg$

   Verwende de Morgans Regeln und $\neg\neg A \equiv A$

$\mapsto$ Negationsnormalform (NNF)

Eine logische Formel ist in Negationsnormalform (NNF), falls die Negationsoperatoren in ihr nur direkt über atomaren Aussagen vorkommen.

Umformung in KNF

Vier Schritte:

1. Elimination von $\leftrightarrow$

   Verwende $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

2. Elimination von $\rightarrow$

   Verwende $A \rightarrow B \equiv (\neg A \vee B)$

3. "Nach innen schieben" von $\neg$

   Verwende de Morgans Regeln und $\neg\neg A \equiv A$ (NNF)

4. "Nach innen schieben" von $\vee$

   Verwende Distributivität von $\vee$ über $\wedge$

Umformung in KNF: Beispiel

Gegeben:

$$P \leftrightarrow (Q \vee R)$$

1. Elimination von $\leftrightarrow$

$$(P \rightarrow (Q \vee R)) \wedge ((Q \vee R) \rightarrow P)$$

2. Elimination von $\rightarrow$

$$(\neg P \vee Q \vee R) \wedge (\neg(Q \vee R) \vee P)$$

3. "Nach innen schieben" von $\neg$ (NNF)

$$(\neg P \vee Q \vee R) \wedge ((\neg Q \wedge \neg R) \vee P)$$

4. "Nach innen schieben" von $\vee$

$$(\neg P \vee Q \vee R) \wedge (\neg Q \vee P) \wedge (\neg R \vee P)$$

Umformung in DNF

Vier Schritte:

1. Elimination von $\leftrightarrow$

   Verwende $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

2. Elimination von $\rightarrow$

   Verwende $A \rightarrow B \equiv (\neg A \vee B)$

3. "Nach innen schieben" von $\neg$

   Verwende de Morgans Regeln und $\neg\neg A \equiv A$ (NNF)

4. "Nach innen schieben" von $\wedge$

   Verwende Distributivität von $\wedge$ über $\vee$

KNF: Mengenschreibweise

Notation:

Klausel als Menge von Literalen

Formel in KNF als Menge von Klauseln

Beispiel:

$$(P \vee Q \vee R) \wedge (P \vee Q \vee \neg R) \wedge (\neg P \vee Q \vee R) \wedge (\neg P \vee \neg Q \vee R)$$

$$\{ \ \{P, Q, R\}, \ \{P, Q, \neg R\}, \ \{\neg P, Q, R\}, \ \{\neg P, \neg Q, R\} \ \}$$

KNF: Mengenschreibweise

Bedeutung der leeren Menge

- Leere Klausel
  - = leere Menge von Literalen
  - = leere Disjunktion
  - $= \bot$
- Leere Menge von Klausels
  - = leere Konjunktion
  - $= \top$

Vereinfachung der KNF: Subsumption

Theorem (Subsumption Regel)
Enthält eine KNF-Formel (= Klauselmenge) Klauseln K, K' mit

$$K \subset K'$$

dann entsteht eine äquivalente Formel, wenn K' weggelassen wird.

Beweis:

$K = \{L_1, \ldots, L_p\} \subseteq \{L_1, \ldots, L_p, L_{p+1}, \ldots, L_m\} = K'$

F enthält $K \wedge K'$

$$\begin{aligned} K \wedge K' = & \quad (L_1 \vee \cdots \vee L_p) \wedge ((L_1 \vee \cdots \vee L_p) \vee L_{p+1} \vee \ldots L_m) \\ \equiv & \quad (L_1 \vee \cdots \vee L_p) = K \qquad \text{(Absorption)} \end{aligned}$$

Erfüllbarkeit und Allgemeingültigkeit

- Syntax
- Semantik
- Formeln
- Schön, aber wem hilft's?

$\Longrightarrow$ Automatische Bestimmung der Erfüllbarkeit und Allgemeingültigkeit!

Erfüllbarkeit und Allgemeingültigkeit

- Eine Formel ist erfüllbar: Es existiert eine Variablenbelegung, so dass die Formel wahr ist.
- Algorithmus?

Erfüllbarkeit und Allgemeingültigkeit

- Eine Formel ist erfüllbar: Es existiert eine Variablenbelegung, so dass die Formel wahr ist.
- Algorithmus?
- Brute Force: Probiere alle Möglichkeiten und wähle diejenigen, die die Formel wahr machen.

Erfüllbarkeit und Allgemeingültigkeit

- Die Formel ist allgemeingültig (Tautologie) falls alle Belegungen die Formel wahr machen!
- Algorithmus?

Erfüllbarkeit und Allgemeingültigkeit

- Die Formel ist allgemeingültig (Tautologie) falls alle Belegungen die Formel wahr machen!
- Algorithmus?
- Brute Force: Probiere alle Möglichkeiten und überprüfe, ob sie die Formel wahr machen.
- Gibt es bessere Algorithmen? JA!

SATISFIABILITY PROBLEMS - ERFÜLLBARKEIT

- Viele Probleme können als eine Liste von Bedingungen/Einschränkungen (engl. constraints) beschrieben werden → wir suchen Belegungen, die diese Einschränkungen erfüllen.
- Erfolgreich - JA! ⇒ Problem gelöst, Antwort gefunden.
- Beispiele: Stundenplan an der Uni
- Schichtenarbeit im Krankenhaus
- Ein jeder hat seine eigene Bedingungen: Vorlesungen dürfen nicht in Laborräume verlegt werden, Termine dürfen sich nicht überlappen, es sollten nicht mehr als 10 Stunden/Tag eingeplant werden, etc.
- Viele solcher Situationen können als eine Liste von Einschränkungen beschrieben werden.

Existiert eine Zurodnung von Stunden/Räume, bzw. Personal, die alle Einschränkungen erfüllt?

SATISFIABILITY PROBLEMS - ERFÜLLBARKEIT

- Programme automatisch (!) testen
- Spezifiziere die Logik Einschränkungen logical constraints). Z.B. Kontrollsystem für Bahnfahrt
- Beschreibe die Situation *Zwei Züge fahren aufeinander auf derselben Gleisstrecke* als logische Aussage
- Teste ob die logischen Einschränkungen + obige Situation sind gleichzeitig erfüllbar
- JA? $\Rightarrow$ Systemfehler!

KNF

- Erfüllbarkeitsprobleme werden als KNF Formeln geschrieben:

$$(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$$

- $(A \vee B \vee \neg C)$ ist eine Klausel:

KNF

- Erfüllbarkeitsprobleme werden als KNF Formeln geschrieben:

$$(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$$

- $(A \vee B \vee \neg C)$ ist eine Klausel: Disjunktion von Literale
- $A, B, \neg C$ sind Literale:

KNF

- Erfüllbarkeitsprobleme werden als KNF Formeln geschrieben:

$$(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$$

- $(A \vee B \vee \neg C)$ ist eine Klausel: Disjunktion von Literale
- $A, B, \neg C$ sind Literale: eine Konstante oder die Negation einer Konstanten
- Jede Klausel ist eine Anforderung die erfüllbar sein muss und es gibt mehrere Möglichkeiten diese Anforderung zu erfüllen.
- Jede Aussage der propositionalen Logik läßt sich in KNF bringen!

KNF Umformung

Vier Schritte:

1. Elimination von $\leftrightarrow$

   Verwende $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

2. Elimination von $\rightarrow$

   Verwende $A \rightarrow B \equiv (\neg A \vee B)$

3. "Nach innen schieben" von $\neg$

   Verwende de Morgans Regeln und $\neg\neg A \equiv A$ (NNF)

4. "Nach innen schieben" von $\vee$

   Verwende Distributivität von $\vee$ über $\wedge$

KNF Umformung

- Jede Aussage läßt sich in KNF umformen aber die Länge der Formel kann exponentiell wachsen!

KNF Vereinfachung

- Ziel ist Formeln in KNF zu vereinfachen.
- Manchmal landet man auf die sogenannte leere Klausel und leere Formel
- Was ist die Bedeutung dieser leeren Klauseln und leeren Formeln?

KNF Vereinfachung

- Eine leere Klausel ist falsch: Es gibt keine Bedingungen zu erfüllen. Im Allgemeinen, eine Disjunktion ohne Disjunktionsglieder ist falsch.
- Weshalb?

KNF Vereinfachung

- Eine leere Klausel ist falsch: Es gibt keine Bedingungen zu erfüllen. Im Allgemeinen, eine Disjunktion ohne Disjunktionsglieder ist falsch.
- Weshalb? Um eine Disjunktion wahr zu machen, muss man mindestens ein Glied wahr machen. Es gibt aber keine Glieder. Daher kann die leere Disjunktion nie wahr sein!
- Eine leere Aussage (ohne Klauseln) ist wahr: Keine Anforderungen...
- Weshalb?

KNF Vereinfachung

- Eine leere Klausel ist falsch: Es gibt keine Bedingungen zu erfüllen. Im Allgemeinen, eine Disjunktion ohne Disjunktionsglieder ist falsch.
- Weshalb? Um eine Disjunktion wahr zu machen, muss man mindestens ein Glied wahr machen. Es gibt aber keine Glieder. Daher kann die leere Disjunktion nie wahr sein!
- Eine leere Aussage (ohne Klauseln) ist wahr: Keine Anforderungen...
- Weshalb? Eine Konjunktion ohne Konjunktionsglieder ist wahr. Um diesen Ausdruck wahr zu machen, müssen wir alle Glieder wahr machen. Es gibt aber keine Glieder, d.h. der Ausdruck ist trivialerweise wahr.

KNF Vereinfachung

- Eine Aussage die die leere Klausel enthält ist falsch!
- Es gibt eine unmögliche Anforderung...
- Weshalb?

KNF Vereinfachung

- Eine Aussage die die leere Klausel enthält ist falsch!
- Es gibt eine unmögliche Anforderung...
- Weshalb? Die leere Klausel ist falsch, die Aussage ist eine Konjunktion von Klauseln, d.h. falsch

Erfüllbarkeitsalgorithmen

- Gegeben eine Aussage in KNF, wie können wir deren Erfüllbarkeit testen?

ERFÜLLBARKEITSALGORITHMEN

- Gegeben eine Aussage in KNF, wie können wir deren Erfüllbarkeit testen?
- Wir testen, ob es eine Belegung existiert, so dass die Aussage wahr ist.
- D.h. wir suchen eine solche Belegung/Modell.
- Erste Methode: Brute Force - Teste alle Möglichkeiten durch. Funktioniert, aber die Anzahl möglicher Belegungen wächst exponentiell zur Variablenanzahl.
- SCHNECKENLANGSAM!!!

ERFÜLLBARKEITSALGORITHMEN

- Wir basteln einen sogenannten Suchbaum
- Startpunkt ist eine Variable P und alle möglichen Belegungen von P: 0 oder 1, falsch oder wahr.
- Auf dem einen Ast nehmen wir an P sei falsch, auf dem anderen wahr.
- Die Belegung falsch vereinfacht unser Problem. Weshalb?

Erfüllbarkeitsalgorithmen

- Wir basteln einen sogenannten Suchbaum
- Startpunkt ist eine Variable P und alle möglichen Belegungen von P: 0 oder 1, falsch oder wahr.
- Auf dem einen Ast nehmen wir an P sei falsch, auf dem anderen wahr.
- Die Belegung falsch vereinfacht unser Problem. Weshalb?
- Wir suchen eine Belegung, so dass alle Variablen erfüllen eine Menge von Anforderungen. Falls P falsch ist, dann haben wir eine kürzere Menge von Anforderungen für unser Problem.

Erfüllbarkeitsalgorithmen

- Gegeben eine Belegung für eine Variable, können wir die Aussage vereinfachen und dann können wir den Vorgang mit einer anderen Variable wiederholen.

Beispiel

$$(P \vee Q) \wedge (P \vee \neg Q \vee R) \wedge (T \vee \neg R) \wedge (\neg P \vee \neg T)$$
$$\wedge (P \vee S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

Beispiel

$$(P \vee Q) \wedge (P \vee \neg Q \vee R) \wedge (T \vee \neg R) \wedge (\neg P \vee \neg T)$$
$$\wedge (P \vee S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

- Lasst uns jetzt P falsch machen. Was geht ab?

Beispiel

$$(P \vee Q) \wedge (P \vee \neg Q \vee R) \wedge (T \vee \neg R) \wedge (\neg P \vee \neg T)$$
$$\wedge (P \vee S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

- Lasst uns jetzt P falsch machen. Was geht ab? Die Menge der Einschränkungen vereinfacht sich. Wie?

Beispiel

$$(P \vee Q) \wedge (P \vee \neg Q \vee R) \wedge (T \vee \neg R) \wedge (\neg P \vee \neg T)$$
$$\wedge (P \vee S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

- Lasst uns jetzt P falsch machen. Was geht ab? Die Menge der Einschränkungen vereinfacht sich. Wie?
- $P \vee Q$ ist dann erfüllungsäquivalent zu Q. Weshalb?

Beispiel

$$(P \vee Q) \wedge (P \vee \neg Q \vee R) \wedge (T \vee \neg R) \wedge (\neg P \vee \neg T)$$
$$\wedge (P \vee S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

- Lasst uns jetzt P falsch machen. Was geht ab? Die Menge der Einschränkungen vereinfacht sich. Wie?
- $P \vee Q$ ist dann erfüllungsäquivalent zu Q. Weshalb?
- $P \vee \neg Q \vee R$ wird zu $\neg Q \vee R$
- $\neg P \vee \neg T$ ist erfüllbar und wird weggelassen!
- $P \vee S$ wird S
- Wir erhalten

$$(Q) \wedge (\neg Q \vee R) \wedge (T \vee \neg R) \wedge (S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

- Wir erhalten eine Aussage die P frei ist und ist einfacher als die ursprüngliche Aussage zu der sie erfüllungsäquivalent ist!

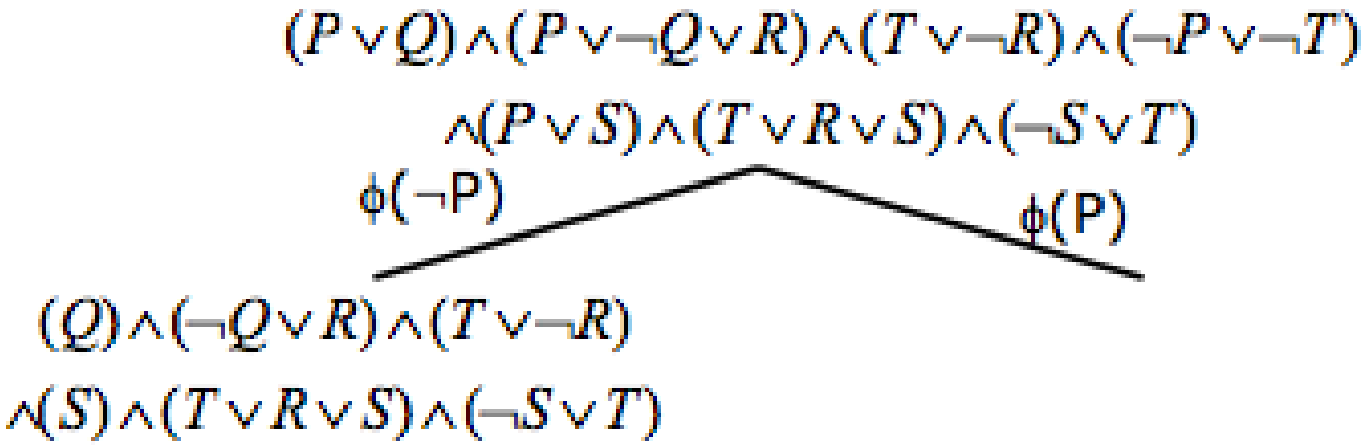
Belege und vereinfache Prozedur

- Sei ϕ eine Aussage und U ein Literal (d.h. eine Variable oder eine negierte Variable)
- Entferne alle Klauseln die U enthalten (diese Klauseln sind erfüllt)
- Entferne $\neg U$ aus allen anderen Klauseln
- Bezeichne die verinfachte Klausel mit $\phi(U)$
- Die Prozedur funktioniert sowohl mit positive als auch mit negative Literale

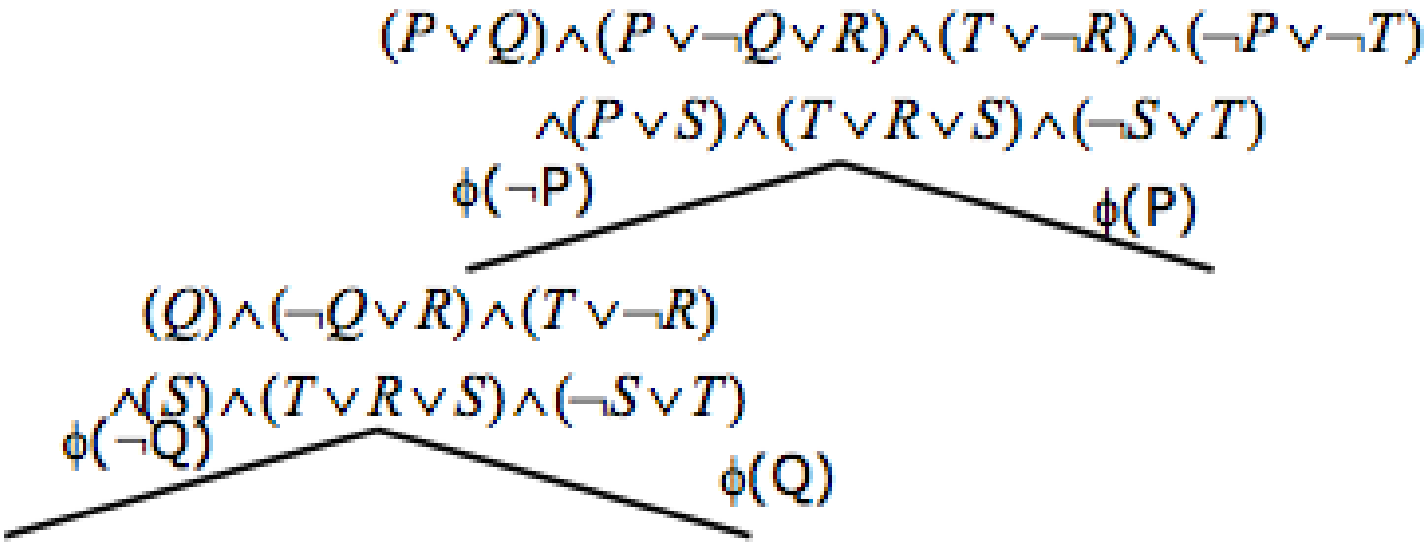
Beispiel: Suchbaum

$$(P \vee Q) \wedge (P \vee \neg Q \vee R) \wedge (T \vee \neg R) \wedge (\neg P \vee \neg T) \wedge (P \vee S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

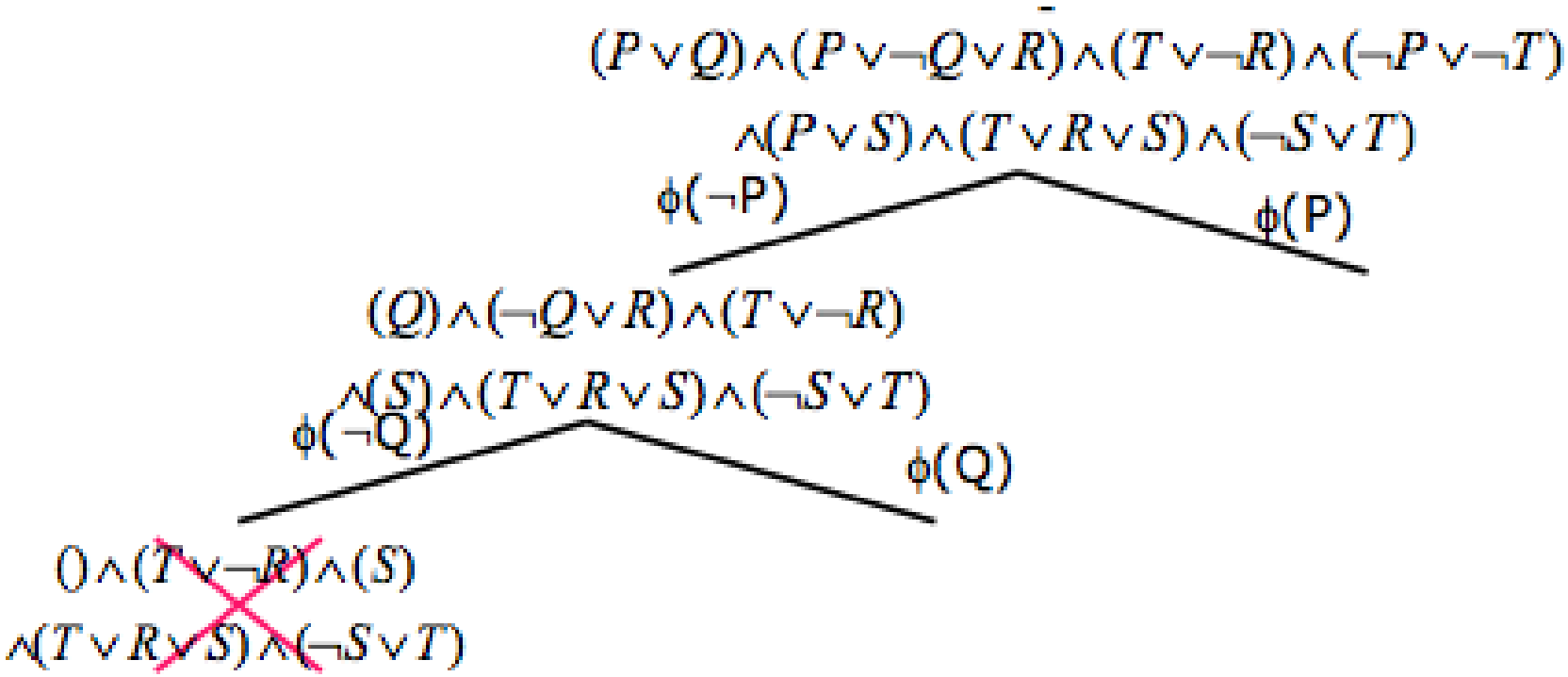
Beispiel: Suchbaum



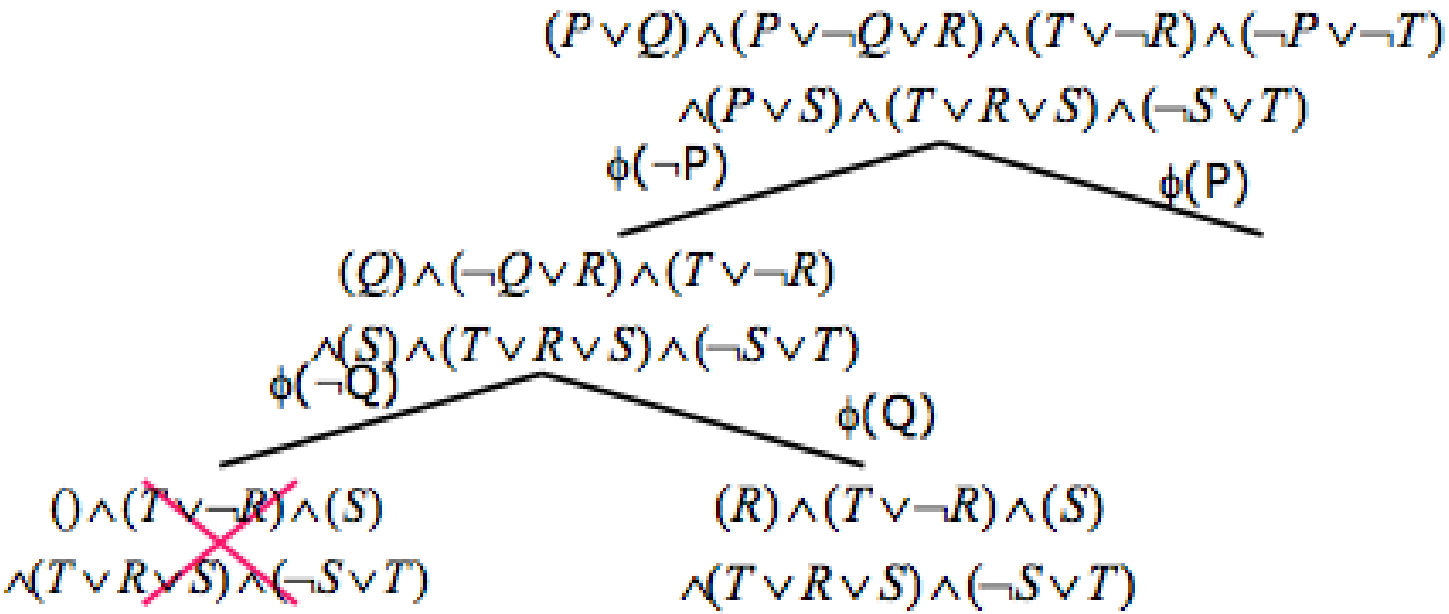
BEISPIEL: SUCHBAUM



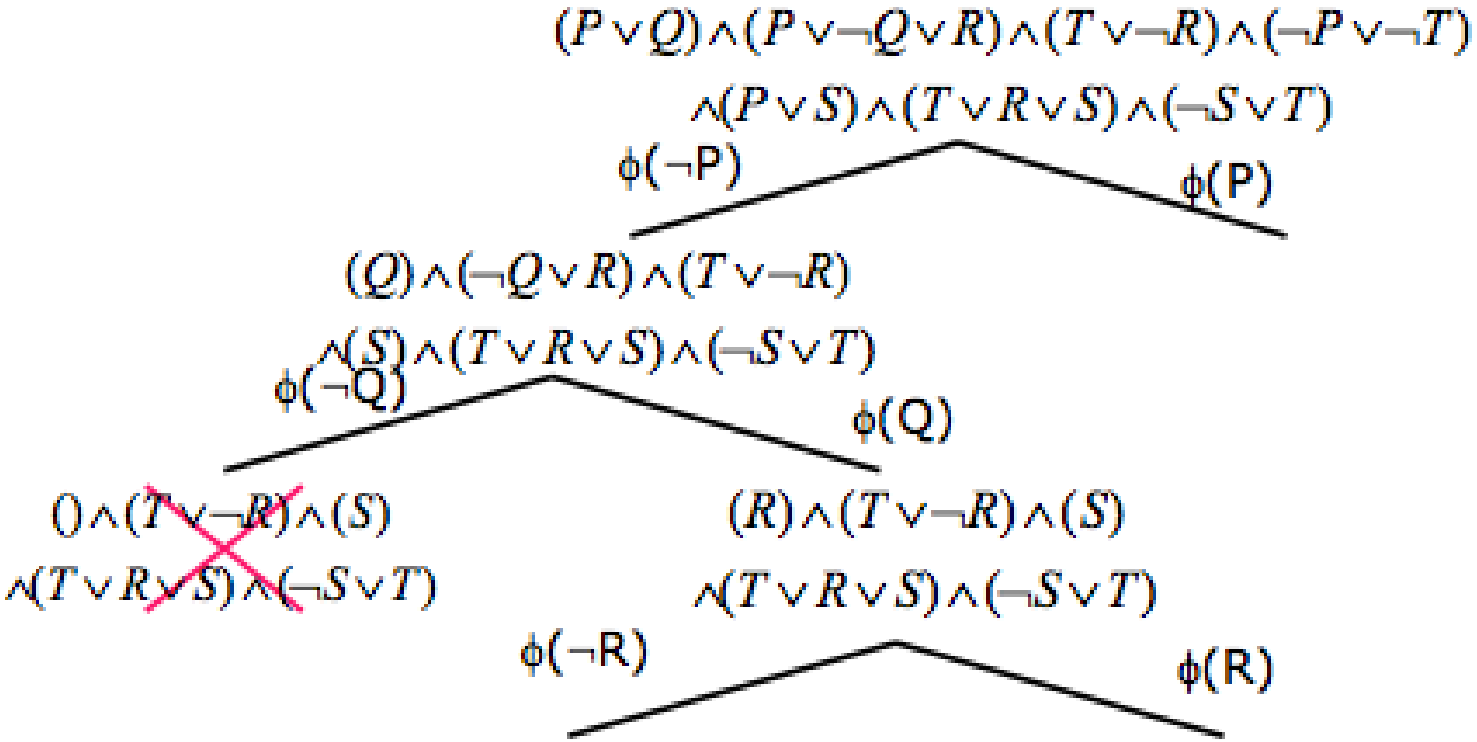
Beispiel: Suchbaum



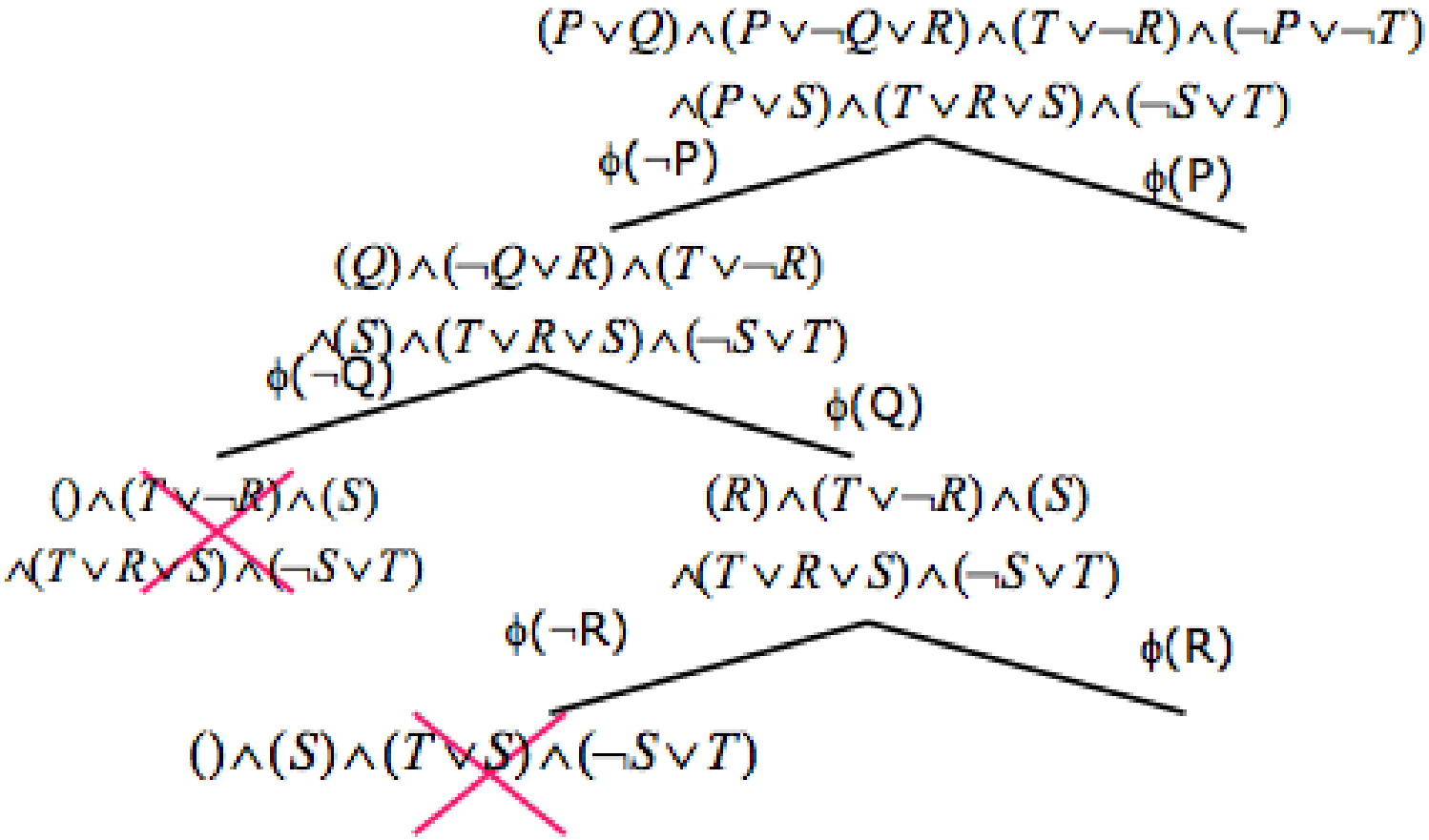
BEISPIEL: SUCHBAUM



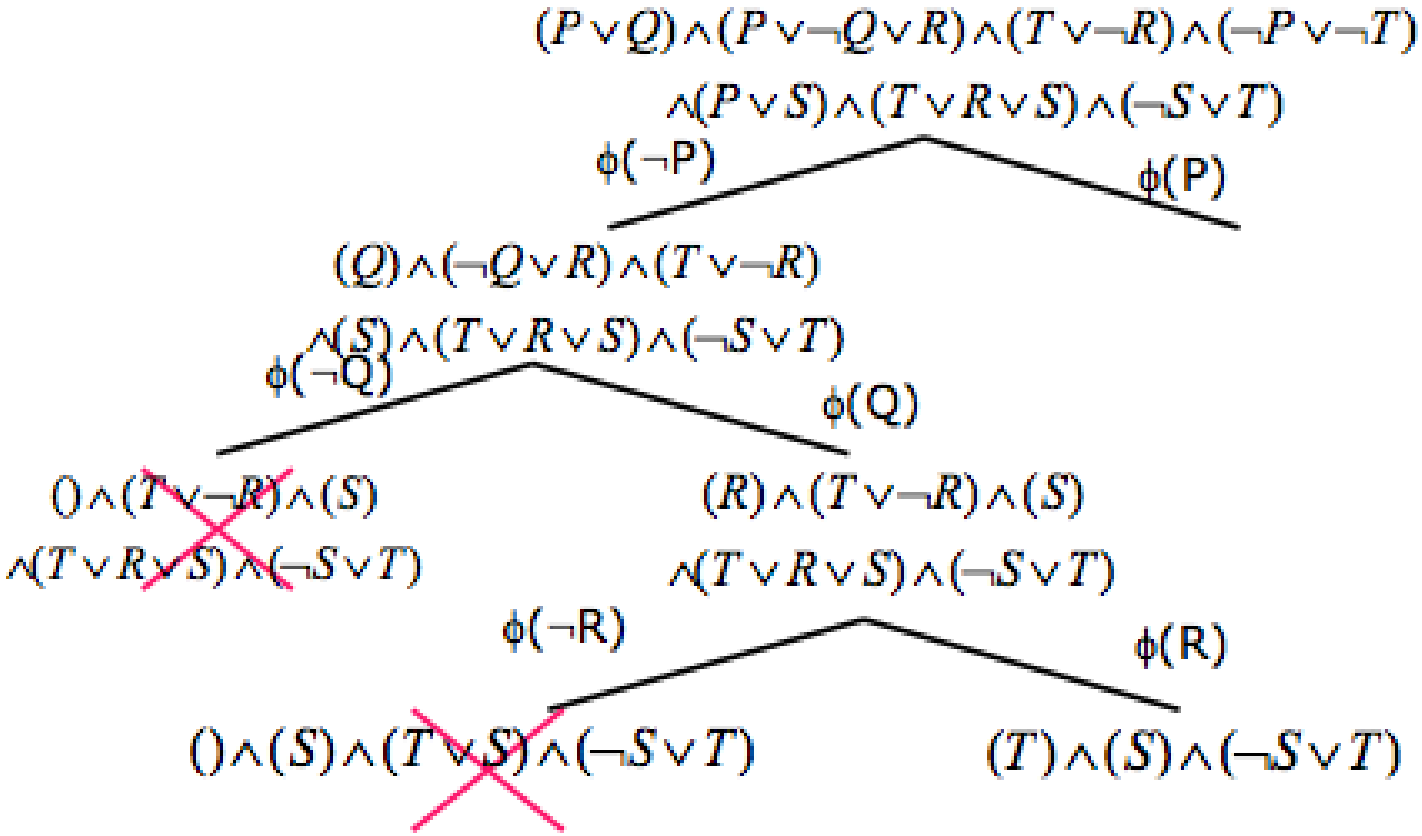
Beispiel: Suchbaum



Beispiel: Suchbaum

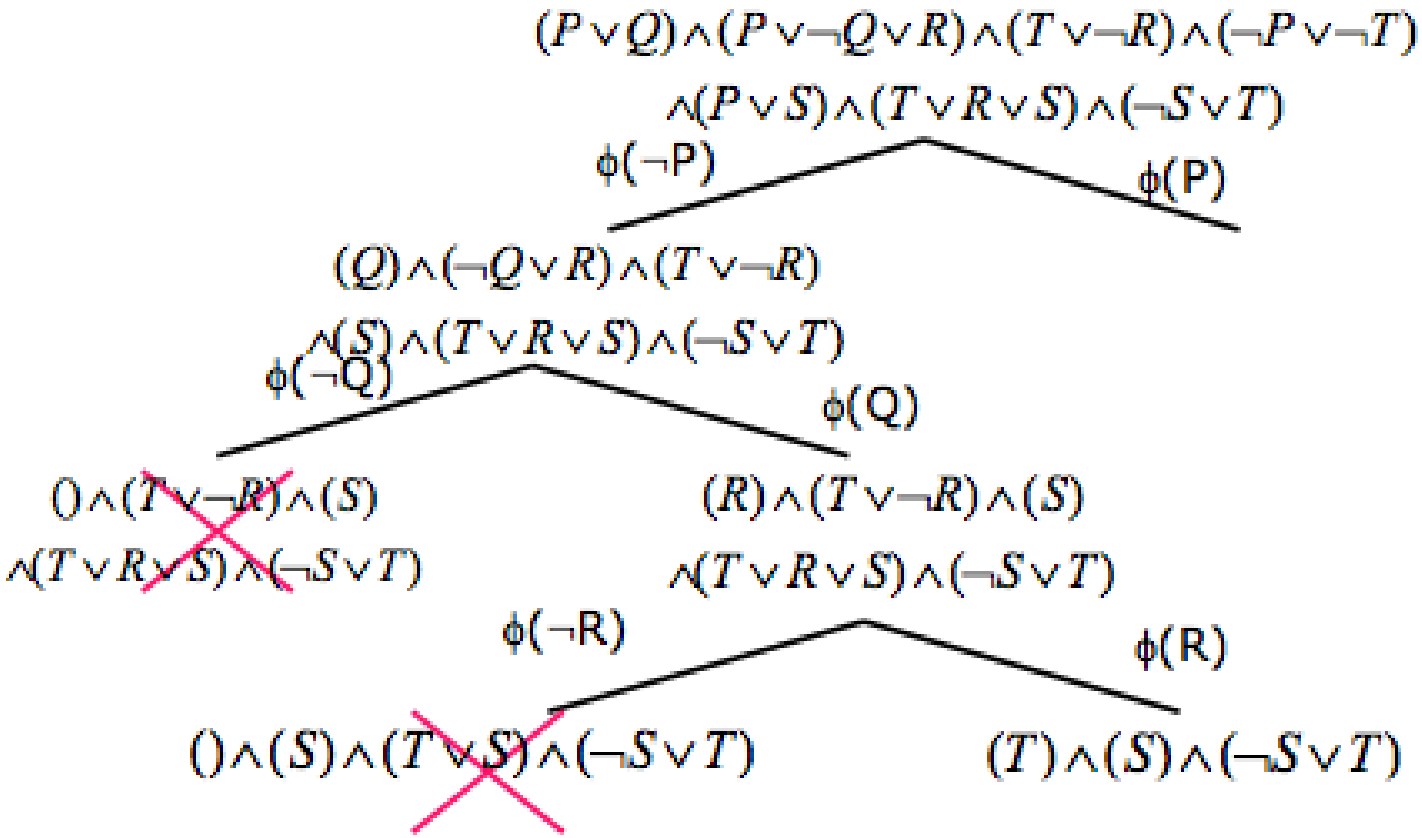


Beispiel: Suchbaum



Beispiel: Suchbaum

Gibt es eine schlauere Möglichkeit?



Beispiel: Suchbaum

- JA!
- Falls ein Literal alleine in einer Klausel erscheint (s. Q, R, T, S) dann wählen wir folgende Belegung: wahr für positive Literale und falsch für negative Literale
- Die letzte Formel der Baums ergibt mit S wahr die Aussage $(T) \wedge (T)$.
- Die einzige Möglichkeit ist T wahr, d.h. die Formel ist wahr.
- Wir erhalten somit die Belegung P falsch, Q wahr, R wahr, S wahr, T wahr.
- Mit dieser Belegung ist die ursprüngliche Formel wahr.

Ein weiteres Beispiel

$$(T \vee X) \wedge (\neg S \vee T) \wedge (S \vee X)$$

- T kommt nur als positives Literal vor.
- Wir müssen nicht unbedingt T als wahr betrachten, schaden tut es auf jeden Fall nicht!
- Daraus lernen wir folgendes: Falls eine Variable als positives Literal vorkommt, wird sie als wahr betrachtet, andernfalls als falsch.
- Im Beispiel: T wahr $\Rightarrow$ alle Klauseln die T enthalten werden entfernt. Es bleibt nur $(S \vee X)$ übrig
- Also wahr.

DPLL Algorithmus

- Aus all dem bisher gelerntem basteln wir nun einen Algorithmus.
- DPLL Algorithmus: Davis, Putnam, Logeman und Loveland.
- Es gibt auch andere Algorithmen die die Erfüllbarkeit von Aussagen testen (satisfiability algorithms): GSAT, WalkSAT
- Der Algorithmus nimmt als Eingabe eine aussagenlogische Formeln und ergibt die Antwort wahr falls die Aussage erfüllbar ist und falsch andernfalls.
- Es basiert auf Rekursion.

DPLL Algorithmus

- If ϕ is empty, return true (embrace truth)

DPLL Algorithmus

- If ϕ is empty, return true
  (embrace truth)
- If there is an empty clause in ϕ, return false
  (reject falsity)
- `Bemerkung: Eine leere Klausel ist immer falsch, d.h. falls wir eine leere Klausel in der Aussage haben, muss die ganze Aussage falsch sein.`

DPLL Algorithmus

- If ϕ is empty, return true
  (embrace truth)
- If there is an empty clause in ϕ, return false
  (reject falsity)
- `Bemerkung: Eine leere Klausel ist immer falsch, d.h. falls wir eine leere Klausel in der Aussage haben, muss die ganze Aussage falsch sein.`
- If there is a unit clause U in ϕ, return DPLL($\phi(U)$)
  (accept the inevitable)
- `Bemerkung: Einheitsklauseln besitzen nur ein Literal!` Dieses Literal wird automatisch belegt mit wahr oder falsch, die Aussage wird vereinfacht und DPLL wird wieder aufgerufen für die vereinfachte Aussage.

DPLL Algorithmus

- If there is a pure literal U in ϕ, return DPLL($\phi(U)$) (go with the flow)
- `Bemerkung: Reine Literale kommen ausschließlich positiv oder negativ in` ϕ `vor`. Ein reines Literal wird mit wahr oder falsch belegt, die Formel wird vereinfacht und DPLL wird erneut aufgerufen (Rekursiver Aufruf!)

DPLL Algorithmus

- If there is a pure literal U in ϕ, return DPLL($\phi(U)$)
  (go with the flow)
- `Bemerkung: Reine Literale kommen ausschließlich positiv oder negativ in` ϕ `vor`. Ein reines Literal wird mit wahr oder falsch belegt, die Formel wird vereinfacht und DPLL wird erneut aufgerufen (Rekursiver Aufruf!)
- For some variable V
  (take a guess)
- `Bemerkung: Falls keine der vorherigen Bedingungen erfüllt wird, wählen wir zufällig irgendeine Variable.`

DPLL Algorithmus

- If DPLL($\phi(V)$) then return true
- `Bemerkung: Belege` V `mit wahr`. Vereinfache und wende erneut DPLL an. Falls das Ergebnis wahr ist, dann ist auch die ursprüngliche Aussage wahr.

DPLL Algorithmus

- If DPLL($\phi(V)$) then return true
- `Bemerkung: Belege` V `mit wahr`. Vereinfache und wende erneut DPLL an. Falls das Ergebnis wahr ist, dann ist auch die ursprüngliche Aussage wahr.
- Else return DPLL($\phi(\neg V)$).
- `Bemerkung: Falls nicht, dann belege` V `mit falsch`. Vereinfache und wende erneut DPLL rekursiv an.

Wie findet man schnell die GUTEN Variablen?

- MOMS Heuristik
- Maximum number of Occurrences, Minimum Sized clauses
- Wähle die Variable die am häufigsten erscheint in Klauseln minimaler Länge.
- Wähle stark eingeschränkte Variablen. Falls es schief gehen sollte, dann bitte schön so früh wie nur möglich! $\Rightarrow$ Keine Riesenbäume, wenn möglich
- Intuitiv: Belegen der stark eingeschränkten Variablen kann uns schnell die Lösung liefern.

Korrektheit und Vollständigkeit

- Falls ein Algorithmus korrekt ist, dann ist jede Antwort eine korrekte Antwort
- Vollständigkeit: Wir finden immer eine Antwort
- DPLL ist korrekt und vollständig
- DPLL ist ein systematischer Algorithmus. Er überspringt nur diejenigen Belegungen die garantiert zur Unerfüllbarkeit führen.
- Deshalb kann es manchmal langsam sein...
- Deshalb ist es zu empfehlen auf korrekte aber nicht vollständige Algorithmen zu greifen: Falls diese Algorithmen eine Antwort liefern ist sie korrekt. ABER eine richtige Antwort kann nicht immer gefunden werden.
- Diese Algorithmen sind schneller als vollständige Algorithmen