# Kapitel L:II

## II. Aussagenlogik

- Syntax der Aussagenlogik
- Semantik der Aussagenlogik
- Eigenschaften des Folgerungsbegriffs
- Äquivalenz

- Formeltransformation
- Normalformen

- Bedeutung der Folgerung
- Erfüllbarkeitsalgorithmen
- Semantische Bäume
- Weiterentwicklung semantischer Bäume

- Syntaktische Schlussfolgerungsverfahren
- Erfüllbarkeitsprobleme


©LETTMANN/STEIN 2023

# Bedeutung der Folgerung

Q. Warum ist der Begriff der Folgerung so wichtig?

A. Folgern (Deduktion) ist ein zentrales Konzept zum Arbeiten mit Modellen.

Hintergrund: Sei $\alpha$ ein Modell eines Weltausschnitts. Der Modellierer vereinbart mit den Verwendern des Modells $\alpha$ folgende Beziehung (Pragmatik):

„$\alpha$ ist erfüllt“ $\Leftrightarrow$ „Der Weltausschnitt wird durch $\alpha$ beschrieben.“

Sinnvolle Modelle entsprechen erfüllbaren Formeln. Sie dienen zur Simulation und werden konstruiert, um Vorhersagen zu machen.


©LETTMANN/STEIN 2023

# Bedeutung der Folgerung

Q. Warum ist der Begriff der Folgerung so wichtig?

A. Folgern (Deduktion) ist ein zentrales Konzept zum Arbeiten mit Modellen.

Hintergrund: Sei $\alpha$ ein Modell eines Weltausschnitts. Der Modellierer vereinbart mit den Verwendern des Modells $\alpha$ folgende Beziehung (Pragmatik):

„$\alpha$ ist erfüllt“ $\Leftrightarrow$ „Der Weltausschnitt wird durch $\alpha$ beschrieben.“

Sinnvolle Modelle entsprechen erfüllbaren Formeln. Sie dienen zur Simulation und werden konstruiert, um Vorhersagen zu machen.

Sei $\beta$ eine Folgerung aus $\alpha$, in Zeichen: $\alpha \models \beta$, dann weiß man (aus der Definition der Folgerung):

$$\alpha \quad \approx \quad \alpha \wedge \beta$$

Das bedeutet aus Modellierungssicht:

- $\beta$ ist immer wahr, wenn $\alpha$ wahr ist.
- $\beta$ ist *verträglich* mit dem Modell $\alpha$. $\beta$ wird vom Modell $\alpha$ vorhergesagt.
- Die Folgerung hat den verträglichen Sachverhalt $\beta$ *explizit* gemacht, sie hat ihn *bewiesen*.
- Beachte: Die Folgerung hat uns über die Veträglichkeit von $\beta$ lediglich *informiert*. Auch ohne das Explizitmachen hätte $\alpha \models \beta$ gegolten.


©LETTMANN/STEIN 2023

# Bedeutung der Folgerung

Q. Warum ist der Begriff der Folgerung so wichtig?

A. Folgern (Deduktion) ist ein zentrales Konzept zum Arbeiten mit Modellen.

Hintergrund: Sei $\alpha$ ein Modell eines Weltausschnitts. Der Modellierer vereinbart mit den Verwendern des Modells $\alpha$ folgende Beziehung (Pragmatik):

„$\alpha$ ist erfüllt“ $\Leftrightarrow$ „Der Weltausschnitt wird durch $\alpha$ beschrieben.“

Sinnvolle Modelle entsprechen erfüllbaren Formeln. Sie dienen zur Simulation und werden konstruiert, um Vorhersagen zu machen.

Sei $\beta$ eine Folgerung aus $\alpha$, in Zeichen: $\alpha \models \beta$, dann weiß man (aus der Definition der Folgerung):

$$\alpha \quad \approx \quad \alpha \wedge \beta$$

Das bedeutet aus Modellierungssicht:

- $\beta$ ist immer wahr, wenn $\alpha$ wahr ist.
- $\beta$ ist *verträglich* mit dem Modell $\alpha$. $\beta$ wird vom Modell $\alpha$ vorhergesagt.
- Die Folgerung hat den verträglichen Sachverhalt $\beta$ *explizit* gemacht, sie hat ihn *bewiesen*.
- Beachte: Die Folgerung hat uns über die Veträglichkeit von $\beta$ lediglich *informiert*. Auch ohne das Explizitmachen hätte $\alpha \models \beta$ gegolten.

Zusammengefasst. Arbeiten mit Modellen $\alpha$ heißt Folgerungen aus $\alpha$ zu erzeugen oder zu überprüfen, ob eine Formel $\beta$ eine Folgerung aus $\alpha$ ist.


©LETTMANN/STEIN 2023

# Bedeutung der Folgerung

Q. Wenn Folgern lediglich Explizitmachen ist, warum hat dann das Überprüfen oder Erzeugen von Folgerungen eine so große Bedeutung?


© LETTMANN/STEIN 2023

# Bedeutung der Folgerung

Q. Wenn Folgern lediglich Explizitmachen ist, warum hat dann das Überprüfen oder Erzeugen von Folgerungen eine so große Bedeutung?

A. Hier ist ein Modell $\alpha$. Gilt $\alpha \models$ „`P3=5`“ ?

```
α = (:AND P1=3 Q1=1 CYL2_IS_OK CYL1_IS_OK PIPE4_IS_OK PIPE3_IS_OK PIPE2_IS_OK PIPE1_IS_OK PUMP_IS_OK (:OR (:NOT PIPE1_IS_OK) (:AND
P1=5 P2=5 Q1=2 Q2=2) (:AND P1=5 P2=5 Q1=1 Q2=1) (:AND P1=5 P2=5 Q1=0 Q2=0) (:AND P1=4 P2=4 Q1=2 Q2=2) (:AND P1=4 P2=4 Q1=1 Q2=1)
(:AND P1=4 P2=4 Q1=0 Q2=0) (:AND P1=3 P2=3 Q1=2 Q2=2) (:AND P1=3 P2=3 Q1=1 Q2=1) (:AND P1=3 P2=3 Q1=0 Q2=0) (:AND P1=2 P2=2 Q1=2
Q2=2) (:AND P1=2 P2=2 Q1=1 Q2=1) (:AND P1=2 P2=2 Q1=0 Q2=0) (:AND P1=1 P2=1 Q1=2 Q2=2) (:AND P1=1 P2=1 Q1=1 Q2=1) (:AND P1=1 P2=1
Q1=0 Q2=0) (:AND P1=0 P2=0 Q1=2 Q2=2) (:AND P1=0 P2=0 Q1=1 Q2=1) (:AND P1=0 P2=0 Q1=0 Q2=0)) (:OR (:NOT PIPE2_IS_OK) (:AND P3=5 P4=5
Q3=2 Q4=2) (:AND P3=5 P4=5 Q3=1 Q4=1) (:AND P3=5 P4=5 Q3=0 Q4=0) (:AND P3=4 P4=4 Q3=2 Q4=2) (:AND P3=4 P4=4 Q3=1 Q4=1) (:AND P3=4
P4=4 Q3=0 Q4=0) (:AND P3=3 P4=3 Q3=2 Q4=2) (:AND P3=3 P4=3 Q3=1 Q4=1) (:AND P3=3 P4=3 Q3=0 Q4=0) (:AND P3=2 P4=2 Q3=2 Q4=2) (:AND
P3=2 P4=2 Q3=1 Q4=1) (:AND P3=2 P4=2 Q3=0 Q4=0) (:AND P3=1 P4=1 Q3=2 Q4=2) (:AND P3=1 P4=1 Q3=1 Q4=1) (:AND P3=1 P4=1 Q3=0 Q4=0)
(:AND P3=0 P4=0 Q3=2 Q4=2) (:AND P3=0 P4=0 Q3=1 Q4=1) (:AND P3=0 P4=0 Q3=0 Q4=0)) (:OR (:NOT PIPE3_IS_OK) (:AND P5=5 P6=5 Q5=2 Q6=2)
(:AND P5=5 P6=5 Q5=1 Q6=1) (:AND P5=5 P6=5 Q5=0 Q6=0) (:AND P5=4 P6=4 Q5=2 Q6=2) (:AND P5=4 P6=4 Q5=1 Q6=1) (:AND P5=4 P6=4 Q5=0
Q6=0) (:AND P5=3 P6=3 Q5=2 Q6=2) (:AND P5=3 P6=3 Q5=1 Q6=1) (:AND P5=3 P6=3 Q5=0 Q6=0) (:AND P5=2 P6=2 Q5=2 Q6=2) (:AND P5=2 P6=2
Q5=1 Q6=1) (:AND P5=2 P6=2 Q5=0 Q6=0) (:AND P5=1 P6=1 Q5=2 Q6=2) (:AND P5=1 P6=1 Q5=1 Q6=1) (:AND P5=1 P6=1 Q5=0 Q6=0) (:AND P5=0
P6=0 Q5=2 Q6=2) (:AND P5=0 P6=0 Q5=1 Q6=1) (:AND P5=0 P6=0 Q5=0 Q6=0)) (:OR (:NOT PIPE4_IS_OK) (:AND P7=5 P8=5 Q7=2 Q8=2) (:AND P7=5
P8=5 Q7=1 Q8=1) (:AND P7=5 P8=5 Q7=0 Q8=0) (:AND P7=4 P8=4 Q7=2 Q8=2) (:AND P7=4 P8=4 Q7=1 Q8=1) (:AND P7=4 P8=4 Q7=0 Q8=0) (:AND
P7=3 P8=3 Q7=2 Q8=2) (:AND P7=3 P8=3 Q7=1 Q8=1) (:AND P7=3 P8=3 Q7=0 Q8=0) (:AND P7=2 P8=2 Q7=2 Q8=2) (:AND P7=2 P8=2 Q7=1 Q8=1)
(:AND P7=2 P8=2 Q7=0 Q8=0) (:AND P7=1 P8=1 Q7=2 Q8=2) (:AND P7=1 P8=1 Q7=1 Q8=1) (:AND P7=1 P8=1 Q7=0 Q8=0) (:AND P7=0 P8=0 Q7=2
Q8=2) (:AND P7=0 P8=0 Q7=1 Q8=1) (:AND P7=0 P8=0 Q7=0 Q8=0)) (:OR (:NOT CYL1_IS_OK) (:AND P2=5 P3=4 Q2=2 Q3=2 VCYL1=2) (:AND P2=5
P3=4 Q2=1 Q3=1 VCYL1=1) (:AND P2=4 P3=3 Q2=2 Q3=2 VCYL1=2) (:AND P2=4 P3=3 Q2=1 Q3=1 VCYL1=1) (:AND P2=3 P3=2 Q2=2 Q3=2 VCYL1=2)
(:AND P2=3 P3=2 Q2=1 Q3=1 VCYL1=1) (:AND P2=2 P3=1 Q2=2 Q3=2 VCYL1=2) (:AND P2=2 P3=1 Q2=1 Q3=1 VCYL1=1) (:AND P2=1 P3=0 Q2=2 Q3=2
VCYL1=2) (:AND P2=1 P3=0 Q2=1 Q3=1 VCYL1=1) (:AND P2=5 P3=5 Q2=0 Q3=0 VCYL1=0) (:AND P2=4 P3=4 Q2=0 Q3=0 VCYL1=0) (:AND P2=3 P3=3
Q2=0 Q3=0 VCYL1=0) (:AND P2=2 P3=2 Q2=0 Q3=0 VCYL1=0) (:AND P2=1 P3=1 Q2=0 Q3=0 VCYL1=0) (:AND P2=0 P3=0 Q2=0 Q3=0 VCYL1=0)) (:OR
(:NOT CYL2_IS_OK) (:AND P4=5 P5=4 Q4=2 Q5=2 VCYL2=2) (:AND P4=5 P5=4 Q4=1 Q5=1 VCYL2=1) (:AND P4=4 P5=3 Q4=2 Q5=2 VCYL2=2) (:AND P4=4
P5=3 Q4=1 Q5=1 VCYL2=1) (:AND P4=3 P5=2 Q4=2 Q5=2 VCYL2=2) (:AND P4=3 P5=2 Q4=1 Q5=1 VCYL2=1) (:AND P4=2 P5=1 Q4=2 Q5=2 VCYL2=2)
(:AND P4=2 P5=1 Q4=1 Q5=1 VCYL2=1) (:AND P4=1 P5=0 Q4=2 Q5=2 VCYL2=2) (:AND P4=1 P5=0 Q4=1 Q5=1 VCYL2=1) (:AND P4=5 P5=5 Q4=0 Q5=0
VCYL2=0) (:AND P4=4 P5=4 Q4=0 Q5=0 VCYL2=0) (:AND P4=3 P5=3 Q4=0 Q5=0 VCYL2=0) (:AND P4=2 P5=2 Q4=0 Q5=0 VCYL2=0) (:AND P4=1 P5=1
Q4=0 Q5=0 VCYL2=0) (:AND P4=0 P5=0 Q4=0 Q5=0 VCYL2=0)) (:OR (:NOT VT_IS_OK) (:AND P6=5 P7=5 Q6=0 Q7=0) (:AND P6=5 P7=4 Q6=1 Q7=1)
(:AND P6=5 P7=3 Q6=2 Q7=2) (:AND P6=4 P7=4 Q6=0 Q7=0) (:AND P6=4 P7=3 Q6=1 Q7=1) (:AND P6=4 P7=2 Q6=2 Q7=2) (:AND P6=3 P7=3 Q6=0
Q7=0) (:AND P6=3 P7=2 Q6=1 Q7=1) (:AND P6=3 P7=1 Q6=2 Q7=2) (:AND P6=2 P7=2 Q6=0 Q7=0) (:AND P6=2 P7=1 Q6=1 Q7=1) (:AND P6=2 P7=0
Q6=2 Q7=2) (:AND P6=1 P7=1 Q6=0 Q7=0) (:AND P6=1 P7=0 Q6=1 Q7=1) (:AND P6=0 P7=0 Q6=0 Q7=0)) (:OR (:NOT Q1=1) (:NOT Q1=2)) (:OR (:NOT
Q1=0) (:NOT Q1=2)) (:OR (:NOT Q1=0) (:NOT Q1=1)) (:OR (:NOT P1=4) (:NOT P1=5)) (:OR (:NOT P1=3) (:NOT P1=5)) (:OR (:NOT P1=3) (:NOT
P1=4)) (:OR (:NOT P1=2) (:NOT P1=5)) (:OR (:NOT P1=2) (:NOT P1=4)) (:OR (:NOT P1=2) (:NOT P1=3)) (:OR (:NOT P1=1) (:NOT P1=5)) (:OR
(:NOT P1=1) (:NOT P1=4)) (:OR (:NOT P1=1) (:NOT P1=3)) (:OR (:NOT P1=1) (:NOT P1=2)) (:OR (:NOT P1=0) (:NOT P1=5)) (:OR (:NOT P1=0)
(:NOT P1=4)) (:OR (:NOT P1=0) (:NOT P1=3)) (:OR (:NOT P1=0) (:NOT P1=2)) (:OR (:NOT P1=0) (:NOT P1=1)) (:OR (:NOT Q8=1) (:NOT Q8=2))
(:OR (:NOT Q8=0) (:NOT Q8=2)) (:OR (:NOT Q8=0) (:NOT Q8=1)) (:OR (:NOT P8=4) (:NOT P8=5)) (:OR (:NOT P8=3) (:NOT P8=5)) (:OR (:NOT
P8=3) (:NOT P8=4)) (:OR (:NOT P8=2) (:NOT P8=5)) (:OR (:NOT P8=2) (:NOT P8=4)) (:OR (:NOT P8=2) (:NOT P8=3)) (:OR (:NOT P8=1) (:NOT
P8=5)) (:OR (:NOT P8=1) (:NOT P8=4)) (:OR (:NOT P8=1) (:NOT P8=3)) ...
```


© LETTMANN/STEIN 2023

# Kapitel L:II

## II. Aussagenlogik

- Syntax der Aussagenlogik
- Semantik der Aussagenlogik
- Eigenschaften des Folgerungsbegriffs
- Äquivalenz

- Formeltransformation
- Normalformen

- Bedeutung der Folgerung
- Erfüllbarkeitsalgorithmen
- Semantische Bäume
- Weiterentwicklung semantischer Bäume

- Syntaktische Schlussfolgerungsverfahren
- Erfüllbarkeitsprobleme


©LETTMANN/STEIN 2023

# Erfüllbarkeitsalgorithmen

Q. Was hat die Beantwortung der Folgerungsfrage „Gilt $\alpha \models \beta$ ?“ mit dem Erfüllbarkeitsproblem zu tun?

A. Die Frage „Gilt $\alpha \models \beta$ ?“ lässt sich mit Hilfe eines Erfüllbarkeitsalgorithmus beantworten.

$$\begin{aligned}
& \alpha \models \beta \\
\Leftrightarrow\ & \text{jede Bewertung } \mathcal{I} \text{ mit } \mathcal{I}(\alpha) = 1 \text{ erzwingt } \mathcal{I}(\beta) = 1 \\
\Leftrightarrow\ & \alpha \rightarrow \beta \text{ ist tautologisch} \\
\Leftrightarrow\ & \alpha \wedge \neg\beta \text{ ist widerspruchsvoll bzw. unerfüllbar}
\end{aligned}$$

bzw.

$$\begin{aligned}
& \alpha \not\models \beta \\
\Leftrightarrow\ & \text{nicht } (\alpha \wedge \neg\beta \text{ ist unerfüllbar}) \\
\Leftrightarrow\ & \alpha \wedge \neg\beta \text{ ist erfüllbar}
\end{aligned}$$


©LETTMANN/STEIN 2023

# Erfüllbarkeitsalgorithmen

Verschiedene Möglichkeiten, die Erfüllbarkeit einer Formel $\alpha$ zu entscheiden.

## Erstellen einer Wahrheitstafel

Vorteil: beliebige Formelstruktur, beliebige Junktoren

Nachteil: auch in einfachen Fällen exponentiell,
keine Berücksichtigung der Formelstruktur


©LETTMANN/STEIN 2023

# Erfüllbarkeitsalgorithmen

Verschiedene Möglichkeiten, die Erfüllbarkeit einer Formel $\alpha$ zu entscheiden.

## Erstellen einer Wahrheitstafel

Vorteil: beliebige Formelstruktur, beliebige Junktoren

Nachteil: auch in einfachen Fällen exponentiell,
keine Berücksichtigung der Formelstruktur

## Analyse der Formelstruktur

(a) Top-down-Auswertung des Formelbaumes mit Fallunterscheidung bei Alternativen.

(b) systematische Anwendung mit Darstellung als Baum: analytisches Tableau

Vorteil: beliebige Formelstruktur, beliebige Junktoren

Nachteil: auch in einfachen Fällen exponentiell,
umfangreiche Implementation


©LETTMANN/STEIN 2023

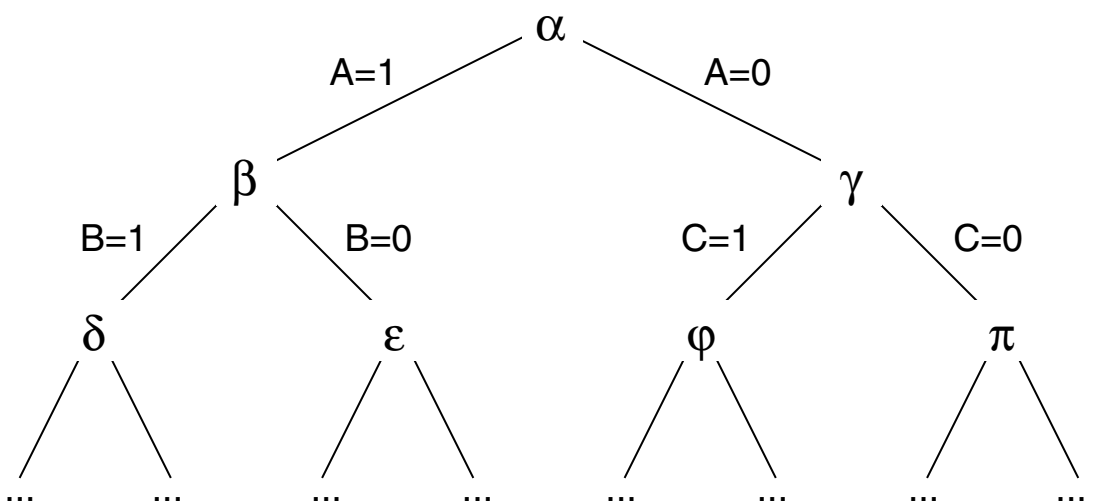
# Erfüllbarkeitsalgorithmen

## Semantische Bäume

Kombination aus der Überprüfung von Bewertungen und Analyse der Formelstruktur.

Vorteil: sukzessive Auswertung orientiert sich an Formelstruktur

Nachteil: reduzierte (=teil-ausgewertete) Formel relativ kompliziert zu berechnen





©LETTMANN/STEIN 2023

Bemerkungen:

- Die Bewertung eines Atoms $A$ mit 0 bzw. 1 kann auf Formelebene dadurch nachgebildet werden, dass $A$ durch die widerspruchsvolle Formel $T \wedge \neg T$ bzw. die tautologische Formel $T \vee \neg T$ substituiert wird. ($T$ beliebig, aber fest, für alle Ersetzungen gleich wählbar.)
- Die reduzierte Formel ist die Formel, die sich aufgrund der möglichen Vereinfachungen aus einer entsprechend substituierten Formel ergibt. Die Vereinfachungsregeln folgen auf der nächsten Folie.


© LETTMANN/STEIN 2023

# Kapitel L:II

## II. Aussagenlogik

- Syntax der Aussagenlogik
- Semantik der Aussagenlogik
- Eigenschaften des Folgerungsbegriffs
- Äquivalenz

- Formeltransformation
- Normalformen

- Bedeutung der Folgerung
- Erfüllbarkeitsalgorithmen
- Semantische Bäume
- Weiterentwicklung semantischer Bäume

- Syntaktische Schlussfolgerungsverfahren
- Erfüllbarkeitsprobleme


©LETTMANN/STEIN 2023

# Semantische Bäume

## Definition 34 (1-Äquivalenzen, 0-Äquivalenzen)

1. $\neg(\beta \vee \neg\beta) \approx (\beta \wedge \neg\beta)$,
   $\neg(\beta \wedge \neg\beta) \approx (\beta \vee \neg\beta)$

2. $\alpha \vee (\beta \vee \neg\beta) \approx (\beta \vee \neg\beta) \vee \alpha \approx (\beta \vee \neg\beta)$,
   $\alpha \vee (\beta \wedge \neg\beta) \approx (\beta \wedge \neg\beta) \vee \alpha \approx \alpha$

3. $\alpha \wedge (\beta \vee \neg\beta) \approx (\beta \vee \neg\beta) \wedge \alpha \approx \alpha$,
   $\alpha \wedge (\beta \wedge \neg\beta) \approx (\beta \wedge \neg\beta) \wedge \alpha \approx (\beta \wedge \neg\beta)$

4. $\alpha \rightarrow (\beta \vee \neg\beta) \approx (\beta \vee \neg\beta)$,
   $\alpha \rightarrow (\beta \wedge \neg\beta) \approx \neg\alpha$,
   $(\beta \vee \neg\beta) \rightarrow \alpha \approx \alpha$,
   $(\beta \wedge \neg\beta) \rightarrow \alpha \approx (\beta \vee \neg\beta)$

5. $\alpha \leftrightarrow (\beta \wedge \neg\beta) \approx (\beta \wedge \neg\beta) \leftrightarrow \alpha \approx \neg\alpha$,
   $\alpha \leftrightarrow (\beta \vee \neg\beta) \approx (\beta \vee \neg\beta) \leftrightarrow \alpha \approx \alpha$


©LETTMANN/STEIN 2023

# Semantische Bäume

**Definition 35 (1-Reduktion, 0-Reduktion)**

Sei $\alpha$ eine aussagenlogische Formel und $A \in \mathit{atoms}(\alpha)$. Weiterhin sei $\beta$ das Ergebnis der Ersetzung jedes Vorkommens von $A$ in $\alpha$ durch $A \vee \neg A$. Dann bezeichne

$$\alpha[A/1]$$

eine kürzeste Formel, die aus $\beta$ unter Anwendung der 1- bzw. 0-Äquivalenzen entstehen kann. (1-Reduktion)

Sei $\alpha$ eine aussagenlogische Formel und $A \in \mathit{atoms}(\alpha)$. Weiterhin sei $\beta$ das Ergebnis der Ersetzung jedes Vorkommens von $A$ in $\alpha$ durch $A \wedge \neg A$. Dann bezeichne

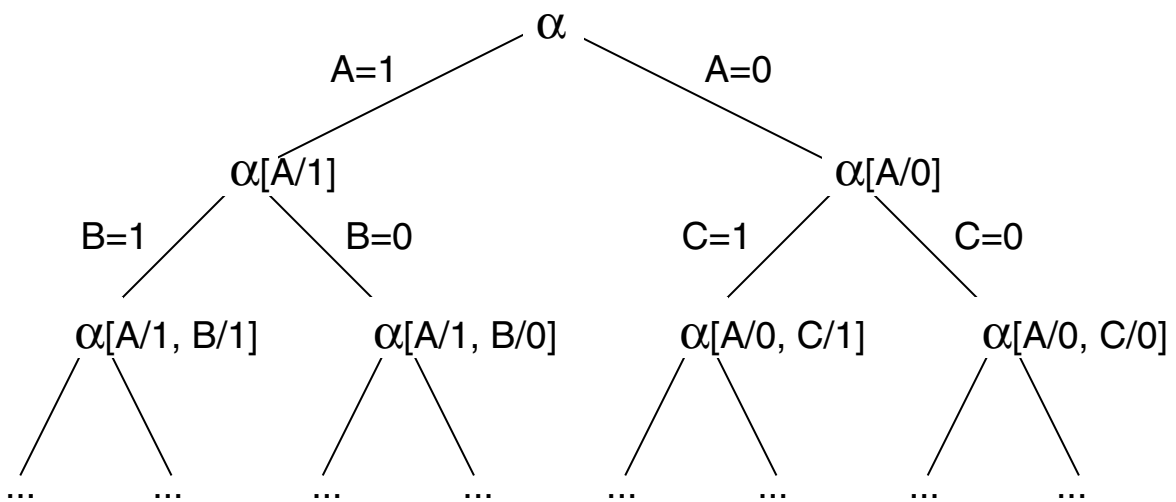
$$\alpha[A/0]$$

eine kürzeste Formel, die aus $\beta$ unter Anwendung der 1- bzw. 0-Äquivalenzen entstehen kann. (0-Reduktion)

Mehrfache Reduktionen hintereinander werden in einer Klammer zusammengefasst. Beispiel: $\alpha[A/1, B/1, C/0] := ((\alpha[A/1])[B/1])[C/0]$


©LETTMANN/STEIN 2023

# Semantische Bäume




**Lemma 36 (Splitting Regel)**

Für eine aussagenlogische Formel $\alpha$ und ein Atom $A \in$ *atoms*$(\alpha)$ gilt

$$\alpha \text{ erfüllbar} \quad \Leftrightarrow \quad \begin{array}{l} \alpha[A/1] \text{ erfüllbar} \\ \text{oder} \\ \alpha[A/0] \text{ erfüllbar} \end{array}$$


©LETTMANN/STEIN 2023

# Semantische Bäume

Beschränkung der Formelstruktur: Sei $\alpha \in$ KNF, $A \in$ *atoms*$(\alpha)$

Algorithmus: SPLIT-SAT
Input: $\alpha$. A formula in CNF.
Output: sat. A flag indicating whether $\alpha$ is satisfiable.

```
SPLIT-SAT(α)
  IF α = β ∧ ¬β AND β is prime formula
  THEN RETURN('FALSE')
  IF α = β ∨ ¬β AND β is prime formula
  THEN RETURN('TRUE')
  A = choose(atoms(α))
  IF SPLIT-SAT(α[A/1])
  THEN RETURN('TRUE')
  ELSE RETURN SPLIT-SAT(α[A/0])
```

Fragen:

- Welche Suchstrategie verfolgt `SPLIT-SAT`?
- Geht `SPLIT-SAT` systematisch vor?


©LETTMANN/STEIN 2023

# Kapitel L:II

## II. Aussagenlogik

- Syntax der Aussagenlogik
- Semantik der Aussagenlogik
- Eigenschaften des Folgerungsbegriffs
- Äquivalenz

- Formeltransformation
- Normalformen

- Bedeutung der Folgerung
- Erfüllbarkeitsalgorithmen
- Semantische Bäume
- Weiterentwicklung semantischer Bäume

- Syntaktische Schlussfolgerungsverfahren
- Erfüllbarkeitsprobleme


© LETTMANN/STEIN 2023

# Weiterentwicklung semantischer Bäume

Verbesserung von `SPLIT-SAT`.

- Idee 1:
  Tritt in einer Formel $\alpha \in$ KNF eine Unitklausel $L$ auf, so muss $L$ mit 1 bewertet werden.
  Stichwort: Unit-Reduktion

- Idee 2:
  Tritt in einer Formel $\alpha \in$ KNF ein Atom $A$ nur in positiven oder nur in negativen Literalen auf, so kann das Literal mit 1 bewertet werden, um $\alpha$ zu erfüllen.
  Stichwort: Pure-Literal-Reduktion

Bemerkung: Beachte den Unterschied zwischen „muss“ und „kann“.


©LETTMANN/STEIN 2023

# Weiterentwicklung semantischer Bäume

Verallgemeinerung von $\alpha[A/1]$.

Sei $L$ ein Literal über *atoms*$(\alpha)$.

- Dann gilt für $L = A$: $\alpha[L/1] := \alpha[A/1]$
- Dann gilt für $L = \neg A$: $\alpha[L/1] = \alpha[\neg A/1] := \alpha[A/0]$

Mit $\neg L$ meinen wir das komplementäre Literal, d.h. für $L = \neg A$ sei $\neg L = A$.

Beispiel:
Sei $\alpha = \neg A$ und $L = \neg A$.
$\alpha[L/1] = \alpha[\neg A/1] = \alpha[A/0] \approx \neg(T \wedge \neg T) \approx T \vee \neg T$
Man ersetzt quasi Literal und Komplement durch die Formeln für die Wahrheitswerte.

## Algorithmische Hinweise zur Realisierung von `SPLIT-SAT`

- Datenstruktur: Listen von Listen von Literalen
- Berechnung von $\alpha[L/1]$:
  1. Streiche Klauseln mit Literal $L$.
  2. Streiche $\neg L$ in den verbliebenen Klauseln.
  3. $\alpha$ erfüllbar, falls Klauselliste leer; falls eine Literalliste leer, Backtracking.


©LETTMANN/STEIN 2023

# Weiterentwicklung semantischer Bäume

Davis-Putnam-Algorithmus [Davis/Putnam 1960, Davis/Loveland/Logemann 1962]

1. Unit-Reduktion
2. Pure-Literal-Reduktion
3. Splitting

Algorithmus: DPLL-SAT
Input: $\alpha$. A formula in CNF.
Output: sat. A flag indicating whether $\alpha$ is satisfiable.

```
DPLL-SAT(α)
  IF α = β ∧ ¬β AND β is prime formula
  THEN RETURN('FALSE')
  IF α = β ∨ ¬β AND β is prime formula
  THEN RETURN('TRUE')

  IF units(α) ≠ ∅
  THEN L = choose(units(α)), RETURN(DPLL-SAT(α[L/1]))
  IF pures(α) ≠ ∅
  THEN L = choose(pures(α)), RETURN(DPLL-SAT(α[L/1]))

  A = choose(atoms(α))
  IF DPLL-SAT(α[A/1])
  THEN RETURN('TRUE')
  ELSE RETURN DPLL-SAT(α[A/0])
```


©LETTMANN/STEIN 2023

# Weiterentwicklung semantischer Bäume

Beispiel für `DPLL-SAT`:

$$\alpha = (A \vee \neg B \vee \neg C) \wedge (\neg A \vee B \vee C) \wedge$$
$$(\neg B \vee C) \wedge (B \vee \neg C) \wedge (\neg B \vee \neg C) \wedge (\neg F \vee C)$$

$$\alpha[\neg F/1] = (A \vee \neg B \vee \neg C) \wedge (\neg A \vee B \vee C) \wedge (\neg B \vee C) \wedge (B \vee \neg C) \wedge (\neg B \vee \neg C)$$

$$\alpha[\neg F/1, A/1] = (B \vee C) \wedge (\neg B \vee C) \wedge (B \vee \neg C) \wedge (\neg B \vee \neg C)$$

$$\alpha[\neg F/1, A/0] = (\neg B \vee \neg C) \wedge (\neg B \vee C) \wedge (B \vee \neg C) \wedge (\neg B \vee \neg C)$$

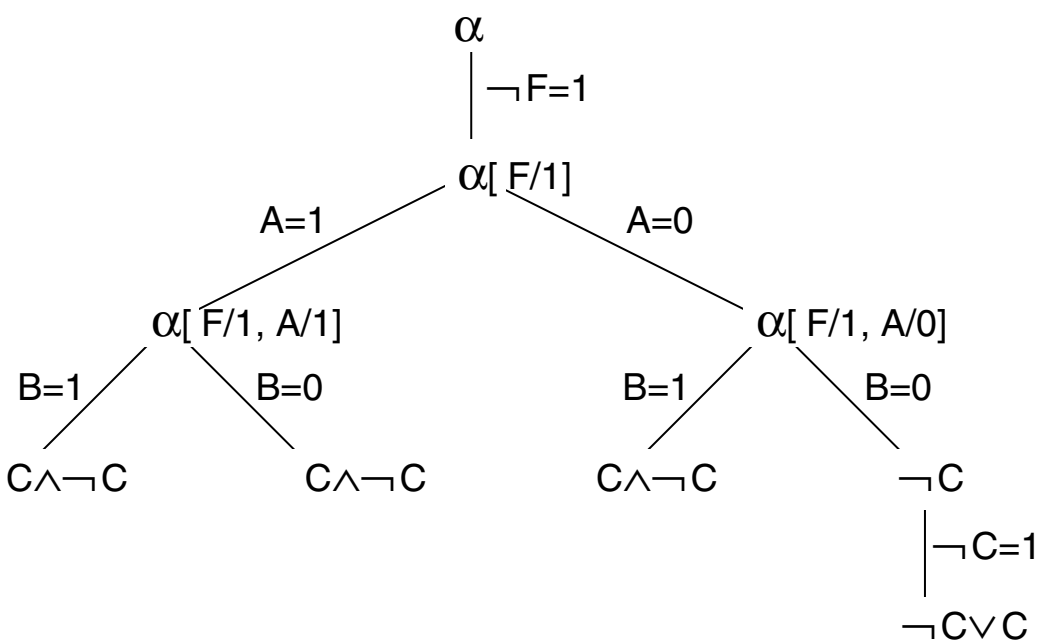
$$\alpha[\neg F/1, A/1, B/1] = C \wedge \neg C$$

$$\alpha[\neg F/1, A/1, B/0] = C \wedge \neg C$$

$$\alpha[\neg F/1, A/0, B/1] = \neg C \wedge C$$

$$\alpha[\neg F/1, A/0, B/0] = \neg C$$

$$\alpha[\neg F/1, A/0, B/0, C/1] = \neg C \vee C$$





© LETTMANN/STEIN 2023

# Weiterentwicklung semantischer Bäume

Auswahlkriterien für Splittingregel:

- Erstes Vorkommen des Atoms bzw. Literals (systematische Suche)
- häufigstes Atom bzw. Literal
  (gute Heuristik, falls Wahrscheinlichkeit für Erfüllbarkeit der Formel hoch)
- Atom mit größter Differenz aus Anzahl positiver und negativer Vorkommen


© LETTMANN/STEIN 2023

# Weiterentwicklung semantischer Bäume

Auswahlkriterien für Splittingregel:

- Erstes Vorkommen des Atoms bzw. Literals (systematische Suche)
- häufigstes Atom bzw. Literal
  (gute Heuristik, falls Wahrscheinlichkeit für Erfüllbarkeit der Formel hoch)
- Atom mit größter Differenz aus Anzahl positiver und negativer Vorkommen

- van Geldern:
  a) Mehrere Atome mit mehr als 3 Vorkommen vorhanden:
     Wähle Atom mit maximalem Produkt aus Anzahl positiver und negativer Vorkommen.
  b) Wähle Atom mit maximalem Vorkommen (2 oder 3).


© LETTMANN/STEIN 2023

# Weiterentwicklung semantischer Bäume

Auswahlkriterien für Splittingregel:

- Erstes Vorkommen des Atoms bzw. Literals (systematische Suche)
- häufigstes Atom bzw. Literal
  (gute Heuristik, falls Wahrscheinlichkeit für Erfüllbarkeit der Formel hoch)
- Atom mit größter Differenz aus Anzahl positiver und negativer Vorkommen

- van Geldern:
  a) Mehrere Atome mit mehr als 3 Vorkommen vorhanden:
     Wähle Atom mit maximalem Produkt aus Anzahl positiver und negativer Vorkommen.
  b) Wähle Atom mit maximalem Vorkommen (2 oder 3).

- SAT-Winner (Paderborn, 1991):
  $h_i(L)$ = Anzahl Klauseln der Länge $i$ mit Literal $L$.
  $H_i(A) = \max(h_i(A), h_i(\neg A)) + 2 \cdot \min(h_i(A), h_i(\neg A))$
  Wähle Atom $A$ mit lexikographisch größtem Vektor

$$(H_2(A), H_3(A), \ldots, H_n(A))$$


©LETTMANN/STEIN 2023