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

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 α ein Modell eines Weltausschnitts. Der Modellierer vereinbart mit den Verwendern des Modells α folgende Beziehung (Pragmatik):

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

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

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 α ein Modell eines Weltausschnitts. Der Modellierer vereinbart mit den Verwendern des Modells α folgende Beziehung (Pragmatik):

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

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

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

$$\alpha \quad \approx \quad \alpha \land \beta$$

Das bedeutet aus Modellierungssicht:

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

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 α ein Modell eines Weltausschnitts. Der Modellierer vereinbart mit den Verwendern des Modells α folgende Beziehung (Pragmatik):

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

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

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

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

Das bedeutet aus Modellierungssicht:

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

Zusammengefasst. Arbeiten mit Modellen α heißt Folgerungen aus α zu erzeugen oder zu überprüfen, ob eine Formel β eine Folgerung aus α ist.

Bedeutung der Folgerung

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

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 α. 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)) ...
```

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.

$\alpha \models \beta$

$\Leftrightarrow$ jede Bewertung $\mathcal{I}$ mit $\mathcal{I}(\alpha) = 1$ erzwingt $\mathcal{I}(\beta) = 1$

$\Leftrightarrow$ $\alpha \to \beta$ ist tautologisch

$\Leftrightarrow$ $\alpha \land \neg\beta$ ist widerspruchsvoll bzw. unerfüllbar

bzw.

$\alpha \not\models \beta$

$\Leftrightarrow$ nicht ($\alpha \land \neg\beta$ ist unerfüllbar)

$\Leftrightarrow$ $\alpha \land \neg\beta$ ist erfüllbar

Erfüllbarkeitsalgorithmen

Verschiedene Möglichkeiten, die Erfüllbarkeit einer Formel α zu entscheiden.

Erstellen einer Wahrheitstafel

Vorteil: beliebige Formelstruktur, beliebige Junktoren

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

Erfüllbarkeitsalgorithmen

Verschiedene Möglichkeiten, die Erfüllbarkeit einer Formel α 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

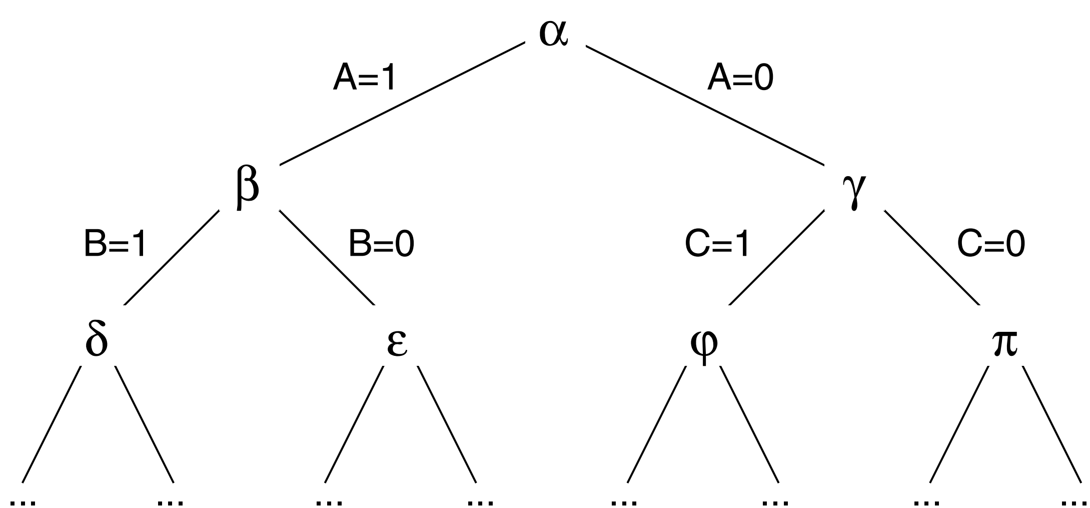
Erfüllbarkeitsalgorithmen

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



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.

Semantische Bäume

Definition 33 (1-Äquivalenzen, 0-Äquivalenzen)

1. $\neg(\beta \lor \neg\beta) \;\approx\; (\beta \land \neg\beta)$,
   $\neg(\beta \land \neg\beta) \;\approx\; (\beta \lor \neg\beta)$

2. $\alpha \lor (\beta \lor \neg\beta) \;\approx\; (\beta \lor \neg\beta) \lor \alpha \;\approx\; (\beta \lor \neg\beta)$,
   $\alpha \lor (\beta \land \neg\beta) \;\approx\; (\beta \land \neg\beta) \lor \alpha \;\approx\; \alpha$

3. $\alpha \land (\beta \lor \neg\beta) \;\approx\; (\beta \lor \neg\beta) \land \alpha \;\approx\; \alpha$,
   $\alpha \land (\beta \land \neg\beta) \;\approx\; (\beta \land \neg\beta) \land \alpha \;\approx\; (\beta \land \neg\beta)$

4. $\alpha \rightarrow (\beta \lor \neg\beta) \;\approx\; (\beta \lor \neg\beta)$,
   $\alpha \rightarrow (\beta \land \neg\beta) \;\approx\; \neg\alpha$,
   $(\beta \lor \neg\beta) \rightarrow \alpha \;\approx\; \alpha$,
   $(\beta \land \neg\beta) \rightarrow \alpha \;\approx\; (\beta \lor \neg\beta)$

5. $\alpha \leftrightarrow (\beta \land \neg\beta) \;\approx\; (\beta \land \neg\beta) \leftrightarrow \alpha \;\approx\; \neg\alpha$,
   $\alpha \leftrightarrow (\beta \lor \neg\beta) \;\approx\; (\beta \lor \neg\beta) \leftrightarrow \alpha \;\approx\; \alpha$

Semantische Bäume

Definition 34 (1-Reduktion, 0-Reduktion)

Sei α eine aussagenlogische Formel und $A \in$ *atoms*(α). Weiterhin sei β das Ergebnis der Ersetzung jedes Vorkommens von A in α durch $A \vee \neg A$. Dann bezeichne

$$\alpha[A/1]$$

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

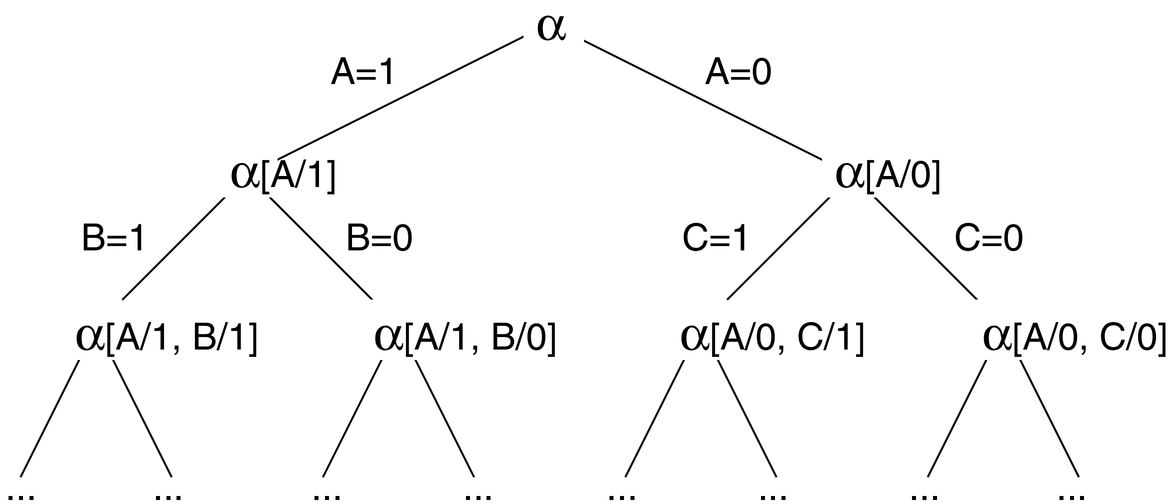
Sei α eine aussagenlogische Formel und $A \in$ *atoms*(α). Weiterhin sei β das Ergebnis der Ersetzung jedes Vorkommens von A in α durch $A \wedge \neg A$. Dann bezeichne

$$\alpha[A/0]$$

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

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

Semantische Bäume



Lemma 35 (Splitting Regel)

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

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

Semantische Bäume

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

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

SPLIT-SAT(α)
  IF $\alpha = \beta \wedge \neg\beta$ **AND** β is prime formula
  THEN RETURN('FALSE')
  IF $\alpha = \beta \vee \neg\beta$ **AND** β is prime formula
  THEN RETURN('TRUE')

  A = choose(*atoms*(α))

  IF SPLIT-SAT($\alpha[A/1]$)
  THEN RETURN('TRUE')
  ELSE RETURN SPLIT-SAT($\alpha[A/0]$)

Fragen:

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

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 α zu erfüllen.
  Stichwort: Pure-Literal-Reduktion

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

Weiterentwicklung semantischer Bäume

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

Sei L ein Literal über *atoms*(α).

- 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. α erfüllbar, falls Klauselliste leer; falls eine Literalliste leer, Backtracking.

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: α. A formula in CNF.
Output: sat. A flag indicating whether α 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])
```

Weiterentwicklung semantischer Bäume

Beispiel für `DPLL-SAT`:

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

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

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

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

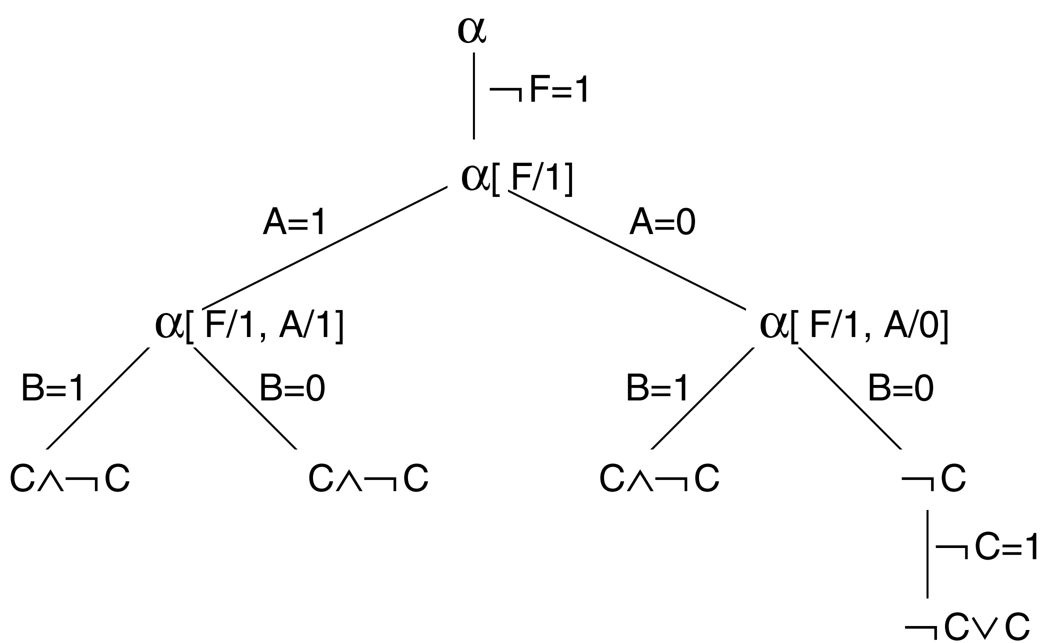
$$\alpha[\neg F/1, A/1, B/1] = C \wedge \neg C$$

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

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

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

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



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

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).

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

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