Aussagenlogik

**Grundbegriffe**

**Atome** A, B, C, …

Nicht weiter zerlegbare Aussage. Kann wahr (⊤) oder falsch (⊥) annehmen. Bsp.: „es Regnet“

**Wahrheitsbelegungen** κ

κ: ↦ {⊤, ⊥} ist eine Abbildung, d.h. κ ordnet Atomen konkrete Wahrheitswerte (⊤, ⊥) zu. Könnte z.B. sein: κ[esRegnet] = ⊤, κ[istAugust] = ⊥

**Prädikate** P, Q, R, …

Bsp.: „istKleiner(x, y)“. Ein Prädikat macht eine Aussage über Individuen (hier x und y) und gibt einen Wahrheitswert (⊤, ⊥) zurück, wenn es ausgewertet wird.  
0-stelliges Prädikat ≡ Atom

**Funktionen** f, g, h, …

z.B. „mutter(x)“ gibt ein Objekt / ein Individuum zurück (hier die Mutter von x).  
0-stellige Funktion ≡ Konstante

**Terme** D, E, F, …

Ein Term ist eine Instanz der Grammatik

T ∷= x | c | f (T, …, T) (x: Variable, c: Konstante, f: Funktion)

Ergebnis der Auswertung eines Terms ist ein Objekt / Individuum.

**Formeln** ϕ, ψ, …

Eine Formel ist eine Instanz der Grammatik

ϕ, ψ ∷= P(t1, …, tn) | ¬ϕ | ϕ ∧ ψ | ϕ ∨ ψ | ϕ → ψ | ∀x(ϕ) | ∃x(ϕ) (t: Term, x: Variable)

Ergebnis der Auswertung einer Formel mittels einer Wahrheitsbelegung ist ein Wahrheitswert

**Aristotelische Formen**

Alle Ps sind Qs: ∀x. P(x) → Q(x) „∀ mit →“

Einige Ps sind Qs: ∃x. P(x) ∧ Q(x) „∃ mit ∧“

**Einige Äquivalenzen**

A → B ≡ ¬A ∨ B ≡ ¬B → ¬A  
A ↔ B ≡ (A → B) ∧ (B → A) ≡ (A → B) ∧ (¬A → ¬B)

**Negationsnormalform (NNF)**

Formel in NNF, wenn “¬” nur vor Atomen vorkommt.  
Jede Formel hat eine NNF.  
  
NNF aus Formel erzeugen:

¬ ¬ ϕ ≡ ϕ (Negation aufheben)

¬ (ϕ ∧ ψ) ≡ ¬ ϕ ∨ ¬ ψ (De Morgan)  
¬ (ϕ ∨ ψ) ≡ ¬ ϕ ∧ ¬ ψ (De Morgan)

¬ ⊤ ≡ ⊥ , ¬ ⊥ ≡ ⊤

**Konjunktive Normalform (CNF)**

Beispiel: (A ∨ B ∨ ¬C) ∧ (¬A ∨ C) ∧ (¬A ∨ ¬B)  
Jede Formel ϕ hat eine CNF.  
  
CNF aus Grammatik erzeugen:

Literal: L ∷= A | ¬A (A ist Atom)  
Klausel: C ∷= ⊥ | neC (leere Klausel ≡ ⊥)

neC ∷= L | L ∨ neC

CNF: ϕ ∷= ⊤ | ψ (leere CNF ≡ ⊤)

ψ ∷= C | C ∧ ψ   
  
CNF aus NNF erzeugen:

CNF(ϕ ∧ ψ) = CNF(ϕ) ∧ CNF(ψ)  
CNF((ϕ ∧ ψ) ∨ λ) = CNF(ϕ ∨ λ) ∧ CNF(ψ ∨ λ) (Distributivgesetz)   
CNF(ϕ ∨ (ψ ∧ λ)) = CNF(ϕ ∨ ψ) ∧ CNF(ϕ ∨ λ) (Distributivgesetz)  
CNF(ϕ) = ϕ (wenn ϕ Klausel, d.h. kein „∧“ enthält)

**Disjunktive Normalform (DNF)**

DNF aus NNF erzeugen:

DNF(ϕ ∨ ψ) = DNF(ϕ) ∨ DNF(ψ)  
DNF((ϕ ∨ ψ) ∧ λ) = DNF(ϕ ∧ λ) ∨ DNF(ψ ∧ λ) (Distributivgesetz)   
DNF(ϕ ∨ (ψ ∧ λ)) = DNF(ϕ ∧ ψ) ∨ DNF(ϕ ∧ λ) (Distributivgesetz)  
DNF(ϕ) = ϕ (wenn ϕ kein ∨ enthält)

Prädikatenlogik (Logik 1. Stufe)

**Einige Äquivalenzen**

¬∀y(ϕ) ≡ ∃y(¬ϕ)  
∀y(ϕ) ≡ ¬∃y(¬ϕ)

**Freie Variablen**

= Variablen, die durch keinen Quantor gebunden sind

Berechnung der Menge der FV:

|  |  |
| --- | --- |
| FV(x) = {x} FV(¬ϕ) = FV(ϕ) FV(E = D) = FV(E) ∪ FV(D) FV(ϕ ∧ ψ) = FV(ϕ) ∪ FV(ψ) FV(f(E1, …, En)) =  FV(P(E1, …, En)) =  FV(∀x(ϕ)) = FV(ϕ) \ {x} | Daraus folgt weiterhin:  FV(ϕ ∨ ψ) = FV(ϕ) ∪ FV(ψ) FV(∃x(ϕ)) = FV(ϕ) \ {x} |

**Substitution**

σ = [E1/x1, …, En/xn] (ersetzt jeweils alle Var. xi durch Term Ei , ließ „Ei statt xi“)

Berechnung von ϕσ: (Substitution σ angewendet auf Formel ϕ)

xσ = σ(x)  
f(E1, …, En)σ = f(E1σ, …, Enσ) f kann Funktion oder Prädikat sein.  
(E = D)σ = (Eσ = Dσ)  
(¬ϕ)σ = ¬(ϕσ)  
(ϕ ∧ ψ)σ = (ϕσ ∧ ψσ)  
(ϕ ∨ ψ)σ = (ϕσ ∨ ψσ)  
(∀x(ϕ))σ = ∀y(ϕσ‘) \*  
(∃x(ϕ))σ = ∃y(ϕσ‘) \*  
  
\* Wobei σ‘(x)=y, für alle anderen z≠x bleibt σ‘(z)=σ(z). Wähle y so, dass y ∉ FV(σ(z)) für alle  
z ∈ FV(∀x(ϕ) bzw. ∃x(ϕ))! **Kurz gesagt**: Alle freien Var. in den Ersetzungen der freien Var. der linken Seite nicht als y wählen! Wenn erlaubt, wähle als y einfach komplett neuen Var.-Namen.

**Pränexe Normalform (PNF)**

Alle Quantoren Qn ∈ {∀, ∃} stehen am Anfang der Formel: Q1x1 … Qnxn(ϕ)  
  
PNF aus NNF erzeugen:

ϕ ∧ ∃x(ψ) ≡ ∃x(ϕ ∧ ψ) ϕ ∧ ∀x(ψ) ≡ ∀x(ϕ ∧ ψ)  
ϕ ∨ ∃x(ψ) ≡ ∃x(ϕ ∨ ψ) \* ϕ ∨ ∀x(ψ) ≡ ∀x(ϕ ∨ ψ) \*

* nur falls x ∉ FV(ϕ). Dies kann durch Umbenennung erreicht werden (alle gebundenen Variablen müssen eindeutig benannt werden):  
   Wende [/x] auf den Quantifizierten Teil an. Bsp.: ϕ ∨ ∃x(ψ) ⇨ ϕ ∨ ∃(ψ[/x])
* Verschachtelte Quantoren behalten ihre Reihenfolge

Außerdem: ∀x(P(x)) ∧ ∀x(Q(x)) ∧ … ≡ ∀x (P(x) ∧ Q(x) ∧ …)

* Existenzquantoren ∃ so weit wie möglich nach vorne

**Skolemform**

= PNF ohne „∃“, nur mit „∀“

Nicht zu jeder Formel existiert eine Skolemform. Allerdings lässt sich zu jeder Formel eine *erfüllbarkeitsäquivalente* Skolemform berechnen.

ϕ und ψ erfüllbarkeitsäquivalent ↔ (ϕ erfüllbar ↔ ψ erfüllbar)  
(Also die eine Formel erfüllbar gdw. die andere erfüllbar)

Skolemisierung: ∃x(ϕ) erfüllbar ↔ ϕ[c/x] (c frisch) erfüllbar

Skolemform aus PNF erzeugen:

∀x1 ... ∀xn ∃y(φ) ⇨ ∀x1 ... ∀xn(φ[fy(x1, ..., xn)/y]) erschöpfend anwenden

Bei Existenzquantoren ∃: Für jedes ∃y(φ) eine neue Skolem-Funktion fy einführen. Diese hat als Parameter alle ∀-Quantifierten Variablen links vom zu eliminierenden „∃“ (also **nicht** die ∃-Quantifizierten). Falls keine „∀“s vor dem zu eliminierenden „∃“ stehen, ist fy von eine nullstellige Funktion: Skolem-Konstante.

**Klauselform**

Sei ψ ≔ Q1x1 … Qnxn(ϕ) ein Skolemform (d.h. alle Qn sind „∀“s). Wenn ϕ in CNF gebracht wurde, kann man die führenden „∀“s weglassen (bzw. implizit annehmen) und ϕ als Klauselmenge schreiben.

**Unifikation**

Gleichungssystem S = {E1 E2, E3 E4, …}, wobei Ei Terme sind. S ist gelöst, falls

1. jede Gleichung die Form hat, wobei v: Variable und E: Term
2. jedes solche v max. 1 mal auf einer linken Seite steht (nochmal in rechter Seite ist okay)

S lösen bzw. Unifizierbarkeit prüfen:

(delete):

S ∪ {x x} ⇨ S

(decomp):

S ∪ { f(E1, …, En) f(D1, …, Dn) } ⇨ S ∪ { E1 D1, …, En Dn }

(conﬂict):

S ∪ { f(E1, . . . , En) g(D1, …, Dk )} ⇨ ⊥ (wenn f ≠ g)

(orient):

S ∪ {E x} ⇨ S ∪ {x E} (E *keine* Variable)

(occurs) / (elim):

S ∪ {x E} ⇨

Falls ⊥ erreicht wird → “nicht unifizierbar“  
Falls gelöste Form erreicht wird → mgu=[E1/x1, E2/x2, …] (aus ablesen)

**Resolution in FOL**

1. NNF (Negations-Normalform)
2. PNF (Pränexe Normalform)
3. Skolemform
4. CNF (Konjunktive Normalform)
5. Unifikation

**Fitch / natürliche Deduktion**

A, B: Terme  
ϕ, ψ: Formeln  
c: ein konkretes Subjekt

Die (∃ I) kann man eigentlich nur in Kombination mit der (∃ E) verwenden

A

…

⊥

¬A

(¬ I)

A ∨ B

(∨ I1)

A

A ∨ B

(∨ I2)

B

A ∧ B

(∧ I)

A

B

A

(¬ E)

¬¬A

A

…

C

C

(∨ E)

B

…

C

A ∨ B

A ∧ B

(∧ E1)

A

A ∧ B

(∧ E2)

B

A

…

B

A → B

(→ I)

ϕ[c/x]

(∃ I)

∃x(ϕ)

c

…

ϕ[c/x]

∀x(ϕ)

(∀ I)

A→B

(→ E)

B

A

c

…

ψ

ψ

(∃ E)

ϕ[c/x]

∃x(ϕ)

∀x(ϕ)

(∀ E)

ϕ[c/x]

A

(⊥ I)

⊥

¬A

beliebig

(⊥ E)

⊥

**Not**

**Oder**

**Und**

**Implikation**

**Existenz**

**Für Alle**

**Bottom**

A = A

(= I)

(= E)

ϕ[A/x]

A = B

ϕ[B/x]

**Gleich**

**Modelle / Semantik**

Ein **Σ-Modell 𝔐** besteht aus:

* einer nichtleeren **Trägermenge M**.
* Einer **Interpretation** für jedes n-stellige Funktionssymbol f/n ∈ Σ, durch eine Funktion
* Einer **Interpretation** für jedes n-stellige Prädikatsymbol P/n ∈ Σ, durch eine Teilmenge

Eine **Umgebung η** ist eine Abbildung η: V ↦ M,  
ordnet jeder Variablen v ∈ V einen Wert m ∈ M der Trägermenge zu.

Ein spezielles Σ-Modell ist das **Herbrand-Modell**, in diesem gibt es keine Variablen in den Termen. Deshalb ist η dann leer und kann weggelassen werden.  
  
Interpretation eines Terms E berechnen:

Erfülltheit einer Formel prüfen: Erfülltheit von ϕ durch 𝔐 und η (kurz 𝔐, η ⊨ ϕ):

(E = D) ⟷

P(E1, …, En) ⟷ ) ∈

∀x(ϕ) ⟷ für alle m ∈ M gilt: 𝔐, η[x ↦ m] ⊨ ϕ

wobei η[x ↦ m](y) =

**Induktion**

Wenn von einer „wenn dann“ Beziehung die Rede ist, ist eine Implikation zu beweisen. Bei einer „genau dann wenn“ Beziehung hingegen eine Äquivalenz.

**Implikation:** „Wenn ein Term E Eigenschaft x erfüllt, dann erfüllt er auch Eigenschaft y“

I. A. : Prüfe ob E die Eigenschaft x erfüllt

Falls ja: Zeige, dass E die Eigenschaft y erfüllt

Falls nein: Fertig (da Implikation trivial erfüllt)

I. V. : seien Terme für die gelte: Wenn sie Eig. x erfüllen, dann erfüllen sie auch Eig. y

I. S. :

Angenommen habe die Eigenschaft x: Zeige, dass auch Eigenschaft y hat

**Äquivalenz:** „Genau dann wenn ein Term E Eig. x erfüllt, dann erfüllt er auch Eig. y“

I. A. : (Beide Fälle müssen gezeigt werden (Term E muss entsprechend gewählt werden))

1. E hat Eigenschaft x: Zeige, dass E Eigenschaft y hat
2. E hat Eigenschaft x nicht: Zeige, dass E Eigenschaft y nicht hat

I. V. : seien Terme für die gelte: Genau dann wenn sie Eig. x erfüllen, erfüllen sie auch Eig. y

I. S. : (Es muss nur entweder a) oder b) gezeigt werden)

Angenommen hat Eig. x: Zeige, dass auch Eig. y hat

* 1. Angenommen hat Eig. y: Zeige, dass auch Eig. x hat
  2. Angenommen hat Eig. x nicht: Zeige, dass auch Eig. y nicht hat