Aussagenlogik: Syntax

$$A ::= X \mid (A \wedge A) \mid (A \vee A) \mid (\neg\ A) \mid (A \Rightarrow A) \mid (A \Leftrightarrow A) \mid 0 \mid 1$$

Prioritätsreihenfolge : $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$.

$A \wedge B$:	Konjunktion (Verundung).
$A \vee B$:	Disjunktion (Veroderung).
$A \Rightarrow B$:	Implikation .
$A \Leftrightarrow B$:	Äquivalenz.
$\neg A$:	negierte Formel.
A	Atom, falls A eine Variable ist.
A	Literal, falls A Atom oder negiertes Atom.

Aussagenlogik: Semantik

Zunächst pro Operation $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
eine Funktion f_{op} gemäß folgender Tabelle.

A	B	$\neg A$	$\wedge$	$\vee$	$\Rightarrow$	NOR	NAND	$\Leftrightarrow$	XOR
1	1	0	1	1	1	0	0	1	0
1	0		0	1	0	0	1	0	1
0	1	1	0	1	1	0	1	0	1
0	0		0	0	1	1	1	1	0

Interpretation

Definition

Eine *Interpretation* I ist

eine Funktion $I : \{\text{aussagenlogische Variablen}\} \to \{0, 1\}$.

Eine Interpretation I liefert Wahrheitswert von Aussagen:

- $I(0) := 0, I(1) := 1$
- $I(\neg A) := f_{\neg}(I(A))$
- $I(A \ op \ B) := f_{op}(I(A), I(B))$, wobei $op \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow \ \ldots\}$

$I(F) = 1$ wird notiert als $I \models F$.

Sprechweisen für $I \models F$:

I ist ein Modell für F, F gilt in I, I macht F wahr

Definition: Tautologie usw.

- A ist eine *Tautologie (Satz, allgemeingültig)* gdw. für alle Interpretationen I gilt: $I \models A$.

- A ist *ein Widerspruch (widersprüchlich, unerfüllbar)* gdw. für alle Interpretationen I gilt: $I(A) = 0$.

- A ist *erfüllbar (konsistent)* gdw. es eine Interpretationen I gibt mit: $I \models A$.

- ein *Modell* für eine Formel A ist eine Interpretation I mit $I(A) = 1$.

Beispiele für Tautologie usw.

- $X \vee \neg X$ ist eine Tautologie.
- $(X \Rightarrow Y) \Rightarrow ((Y \Rightarrow Z) \Rightarrow (X \Rightarrow Z))$ ist eine Tautologie.
- $X \wedge \neg X$ ist ein Widerspruch.
- $X \vee Y$ ist erfüllbar.
- I mit $I(X) = 1, I(Y) = 0$ ist ein Modell für $X \wedge \neg Y$

Äquivalenzen von Aussagen

F, G sind *äquivalent* ($F \sim G$),

gdw.

$F \Leftrightarrow G$ ist eine Tautologie ist.

Tautologien/ Widersprüche: Komplexität

Satz

- Die Frage "Ist A Tautologie" ist entscheidbar
- Die Frage "Ist A erfüllbar?" ist $\mathcal{NP}$-vollständig.
- Die Frage "Ist A Tautologie (Widerspruch)?" ist co-$\mathcal{NP}$-vollständig.

Für $\mathcal{NP}$-vollständig und co-$\mathcal{NP}$-vollständig:

Es sind nur worst-case exponentielle Algorithmen bekannt

$\mathcal{NP}$-vollst.	Algorithmen sind gutartig und optimierbar
co-$\mathcal{NP}$-vollst.	Algorithmen sind nicht gutartig und wenig optimierbar

Folgerungen: Anwendung

Gegeben bzw bekannt:	Fakten, Regeln, Zusammenhänge
Frage:	Was gilt dann auch noch??
	Was weiss man noch (implizit)

Wenn das aussagenlogisch formulierbar ist,

dann entspricht das genau den **Folgerungen**

Folgerungsbegriffe

Zwei verschiedene Begriffe der Folgerungen für Logik:

- semantische Folgerung
  Basis-Begriff; definiert über Modelle

- syntaktische Folgerung (Herleitung, Ableitung)
  Algorithmus meist ein nicht-deterministischer Kalkül, auf Formeln / Formelmengen
  mögliche Ziele:
  Erkennung von Tautologien
  Erkennung von Folgerungsbeziehungen

Folgerungsbegriffe in der Aussagenlogik

Definition Sei $\mathcal{F}$ eine Menge von (aussagenlogischen) Formeln und G eine weitere Formel.

G *folgt semantisch* aus $\mathcal{F}$ Notation: $\mathcal{F} \models G$

gdw.

$\forall$ Interpretationen I : $\Big(\forall F \in \mathcal{F}\colon \;\; I(F) = 1 \;\Big)$ impliziert $I(G) = 1$.

Deduktionstheorem in der Aussagenlogik

Satz (Deduktionstheorem)

$$\{F_1, \ldots, F_n\} \models G$$
gdw.
$$F_1 \wedge \ldots \wedge F_n \Rightarrow G \text{ ist Tautologie.}$$

Beachte: F und G sind äquivalent

gdw.

$\forall I : I \models F$ gdw. $I \models G$

Aussagenlogische Sätze

$\wedge$ und $\vee$ sind kommutativ, assoziativ, und idempotent:

$$
\begin{array}{ccc}
\mathcal{F} \wedge \mathcal{G} & \Leftrightarrow & \mathcal{G} \wedge \mathcal{F} \\
\mathcal{F} \wedge \mathcal{F} & \Leftrightarrow & \mathcal{F} \\
\mathcal{F} \wedge (\mathcal{G} \wedge \mathcal{H}) & \Leftrightarrow & (\mathcal{F} \wedge \mathcal{G}) \wedge \mathcal{H} \\
\mathcal{F} \vee \mathcal{G} & \Leftrightarrow & \mathcal{G} \vee \mathcal{F} \\
\mathcal{F} \vee (\mathcal{G} \vee \mathcal{H}) & \Leftrightarrow & (\mathcal{F} \vee \mathcal{G}) \vee \mathcal{H} \\
\mathcal{F} \vee \mathcal{F} & \Leftrightarrow & \mathcal{F}
\end{array}
$$

Äquivalenzen:

$\neg(\neg A))$	$\Leftrightarrow$	A	
$(A \Rightarrow B)$	$\Leftrightarrow$	$(\neg A \vee B)$	
$(A \Leftrightarrow B)$	$\Leftrightarrow$	$((A \Rightarrow B) \wedge (B \Rightarrow A))$	
$\neg(A \wedge B)$	$\Leftrightarrow$	$\neg A \vee \neg B$	(DeMorgansche Gesetze)
$\neg(A \vee B)$	$\Leftrightarrow$	$\neg A \wedge \neg B$	
$A \wedge (B \vee C)$	$\Leftrightarrow$	$(A \wedge B) \vee (A \wedge C)$	Distributivität
$A \vee (B \wedge C)$	$\Leftrightarrow$	$(A \vee B) \wedge (A \vee C)$	Distributivität
$(A \Rightarrow B)$	$\Leftrightarrow$	$(\neg B \Rightarrow \neg A)$	Kontraposition
$A \vee (A \wedge B)$	$\Leftrightarrow$	A	Absorption
$A \wedge (A \vee B)$	$\Leftrightarrow$	A	Absorption

Normalformen

disjunktive Normalform (DNF).
Disjunktion von Konjunktionen von Literalen.

$$(L_{1,1} \wedge \ldots \wedge L_{1,n_1}) \vee \ldots \vee (L_{m,1} \wedge \ldots \wedge L_{m,n_m})$$

konjunktive Normalform (CNF).
Konjunktion von Disjunktionen von Literalen.

$$(L_{1,1} \vee \ldots \vee L_{1,n_1}) \wedge \ldots \wedge (L_{m,1} \vee \ldots \vee L_{m,n_m})$$

(Literal = Variable oder negierte Variable)

Transformation in Klauselnormalform

Prozedur:

1. Elimination von $\Leftrightarrow$ und $\Rightarrow$:
   $F \Leftrightarrow G \rightarrow F \Rightarrow G \wedge G \Rightarrow F$ und
   $F \Rightarrow G \rightarrow \neg F \vee G$
2. Negation ganz nach innen schieben:

$$\begin{array}{lcl} \neg\neg F & \rightarrow & F \\ \neg(F \wedge G) & \rightarrow & \neg F \vee \neg G \\ \neg(F \vee G) & \rightarrow & \neg F \wedge \neg G \end{array}$$

3. Distributivität (und Assoziativität, Kommutativität) iterativ anwenden, um $\wedge$ nach außen zu schieben ("Ausmultiplikation").
   $F \vee (G \wedge H) \rightarrow (F \vee G) \wedge (F \vee H)$

Transformation in Klauselnormalform

Resultat: Konjunktion von Disjunktionen von Literalen

D.h. eine CNF

Eigenschaften

- Die Resultat- CNF ist äquivalent zur eingegebenen Formel
- Der Algorithmus ist worst-case exponentiell (Zeit und Platz)
- Die Anzahl der Literale in der CNF kann exponentiell sein.

Transformation in Klauselnormalform

Gründe für die Explosion:
Verdoppelung von Unterformeln bei den Transformationsschritten:

- die Elimination von $\Leftrightarrow$:
- Ausmultiplikation mittels Distributivgesetz:

Beispiele:

$(A_1 \Leftrightarrow A_2) \Leftrightarrow (A_3 \Leftrightarrow A_4)$

$\rightsquigarrow$

$(A_1 \Rightarrow A_2 \wedge A_2 \Rightarrow A_1) \Leftrightarrow (A_3 \Rightarrow A_4 \wedge A_4 \Rightarrow A_3)$

$(A_1 \wedge \ldots \wedge A_n) \vee B_2 \vee \ldots \vee B_m \rightsquigarrow ((A_1 \vee B_2) \wedge \ldots \wedge (A_n \vee B_2)) \vee B_3 \ldots \vee B_n$

CNF-Algorithmus: Beispiel

$$
\begin{aligned}
& ((A \wedge B) \Rightarrow C) \Rightarrow C \\
\rightarrow \quad & \neg(\neg(A \wedge B) \vee C) \vee C \\
\rightarrow \quad & (A \wedge B) \wedge \neg C) \vee C \\
\rightarrow \quad & (A \vee C) \wedge (B \vee C) \wedge (\neg C \vee C)
\end{aligned}
$$

Weitere Simplifikationen sind möglich.

Resolution für Aussagenlogik

Resolutionsverfahren:
Erkennt Widersprüchlichkeit
benötigt eine Klauselmenge als Eingabe
statt „Ist F Tautologie“: „Ist $\neg F$ Widerspruch“

Lemma Eine Formel $A_1 \wedge \ldots \wedge A_n \Rightarrow F$ ist allgemeingültig gdw. $A_1 \wedge \ldots \wedge A_n \wedge \neg F$ widersprüchlich ist.

semantisch formuliert:
$\{A_1, \ldots, A_n\} \models F$ gdw. es keine Interpretation I gibt, so dass
$I \models \{A_1, \ldots, A_n, \neg F\}$

Resolutionsverfahren für Aussagenlogik

Gegeben eine Menge M von Klauseln

Iteriere folgende Operation:

Wähle (nichtdeterministisch) 2 Klauseln K, L aus M.

Falls es eine Resolvente R von K, L gibt, füge R zu M hinzu

Stoppe erfolgreich, wenn die leere Klausel erzeugt wurde.

Stoppe nicht widersprüchlich, wenn alle Resolventen schon in M sind, aber die leere Klausel nicht in M ist.

Klauseln werden als Mengen dargestellt!

Resolution für Aussagenlogik

Resolutions-Regel:

$$\frac{\begin{array}{ll} A & \vee B_1 \vee \ldots \vee B_n \\ \neg A & \vee C_1 \vee \ldots \vee C_m \end{array}}{B_1 \vee \ldots \vee B_n \vee C_1 \vee \ldots \vee C_m}$$

zwei Eingabe- Klauseln sind die Elternklauseln
die neu hergeleitete Klausel ist die Resolvente.

Resolution: Eigenschaften

Aussage

Wenn $C \rightarrow C'$ mit Resolution, dann ist C äquivalent zu C'.

Aussage

Resolution terminiert

Grund:

Es gibt nur endlich viele mögliche Klauseln, da Resolution keine neuen Variablen einführt.

Resolution: Eigenschaften

Resolution ist nicht gut geeignet um Modelle von Formeln zu berechnen:

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

Die Formelmenge ist erfüllbar und abgeschlossen bzgl Resolution.

besser Methoden zur Modellberechnung:

Davis-Putnam-Prozedur oder Tableaukalkül

Resolution: Eigenschaften

Satz In der Aussagenlogik gilt:
Für eine unerfüllbare Klauselmenge findet Resolution nach endlich vielen Schritten die leere Klausel.

Beweis geht mit Induktion nach
(Anzahl der Literale) - (Anzahl der Klauseln)

Satz In der Aussagenlogik gilt:
Resolution erkennt unerfüllbare Klauselmengen.
D.h. Resolution ist vollständig.

Resolution: Eigenschaften, Bemerkungen

Optimierung der Resolution durch

Elimination von Redundanzen:

- Löschung von Tautologien
- Löschen von redundanten Klauseln:
  solche mit isolierte Literalen (kein Resolutionsgegenpart)
  Subsumtion: Klauseln sind *redundant*,
  wenn sie Obermengen von anderen Klauseln sind

Resolution: Komplexität

Untere Abschätzung der Komplexität im schlimmsten Fall
von (A. Haken 1985):
Herleitungen können exponentiell lang dauern:

Es gibt Folge von Formeln
    (die sogenannten Taubenschlag-formeln (pigeon hole formula,
    Schubfach-formeln),
mit exponentiell langen Resolutionsherleitungen

Davis-Putnam Algorithmus $DP(\cdot)$

Ist ein Resolutionsverfahren, algorithmisiert

mit Fallunterscheidungen für Literale (A gilt oder A gilt nicht)

und Erkennung isolierter Literale

Davis-Putnam Algorithmus $DP(\cdot)$

1 a. Wenn leere Klausel in C: RETURN `true`.

1 b. Wenn C leere Klauselmenge: RETURN `false`.

2. wenn 1-Klausel $\{P\}$ (bzw. $\{\neg P\}$) ex:

   a Lösche Klauseln in denen P (bzw. $\neg P$) vorkommt.

   b Lösche Literale $\neg P$ (bzw. P) in Klauseln
   ergibt Klauselmenge C'. RETURN $DP(C')$

3. Wenn isolierte Literale existieren:
   Lösche Klauseln, in denen isolierte Literale vorkommen.
   resultierende Klauselmenge: C'. RETURN $DP(C')$

4 Sonst: wähle eine ex. Variable P aus.
   RETURN $DP(C \cup \{\{P\}\}) \wedge DP(C \cup \{\{\neg P\}\})$

Beispiel für DP

$$
\begin{array}{lll}
P, & Q & \\
\neg P, & Q & R \\
P, & \neg Q, & R \\
\neg P, & \neg Q, & R \\
P, & Q, & \neg R \\
\neg P, & Q, & \neg R \\
P, & \neg Q, & \neg R \\
\neg P, & \neg Q, & \neg R
\end{array}
$$

Fall 1: Addiere die Klausel $\{P\}$. nach einigen Schritten:

$$
\begin{array}{ll}
Q, & R \\
\neg Q, & R \\
Q, & \neg R \\
\neg Q, & \neg R
\end{array}
$$

Fall 1.1: Addiere $\{Q\}$: ergibt die leere Klausel.

Fall 1.2: Addiere $\{\neg Q\}$: ergibt die leere Klausel.

Fall 2: Addiere die Klausel $\{\neg P\}$. Nach einigen Schritten:

$$\begin{array}{cc} Q & \\ \neg Q & R \\ Q & \neg R \\ \neg Q & \neg R \end{array}$$

Weitere Schritte für Q ergeben

$$\begin{array}{c} R \\ \neg R \end{array}$$

ergibt leere Klausel.

Raymond Smullyan: Wer ist der Pfefferdieb?

Rätsel von Raymond Smullyan vor:
Es gibt drei Verdächtige: Den Hutmacher, den Schnapphasen und die (Hasel-)Maus. Folgendes ist bekannt:

- Genau einer von ihnen ist der Dieb.
- Unschuldige sagen immer die Wahrheit
- Schnapphase: der Hutmacher ist unschuldig.
- Hutmacher: die Hasel-Maus ist unschuldig

Kodierung: H, S, M

1. $H \vee S \vee M$
2. $H \Rightarrow \neg(S \vee M)$
3. $S \Rightarrow \neg(H \vee M)$
4. $M \Rightarrow \neg(H \vee S)$
5. $\neg S \Rightarrow \neg H$
6. $\neg H \Rightarrow \neg M$

Klauselmenge:

$$\{\{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$$

Kodierung: H, S, M : Resultat

```
pfefferdieb = dp
    "((H \\/  S \\/ M) /\\ (H => -(S \\/ M))
    /\\ (S => -(H \\/ M)) /\\ (M => -(H \\/ S))
    /\\(-S => -H) /\\ (-H => -M))"

Modell: S, -M, -H"
```

Schneller CNF-Algorithmus

CNF-Herstellung kann man beschleunigen zu polynomiell: fast $O(n)$.

Trick: komplexe Subformeln iterativ durch neue Variablen abkürzen.

ABER: Formel F ist nicht äquivalent zur berechneten CNF(F).

Es gilt:

F erfüllbar gdw. CNF(F) erfüllbar.

Das reicht aus für Beweisverfahren und semantische Herleitungen

Schneller CNF-Algorithmus

Wesentlicher Schritt:

Gegeben $H_1 \wedge \ldots \wedge H_n$ wobei H_j eine Tiefe ≥ 4 hat:

Ersetze in H_j alle Subformeln $G_1, \ldots, G_m$ von H_j in Tiefe 3 durch neue Variablen A_i:

D.h. $\boxed{H_j[G_1, \ldots, G_m] \rightsquigarrow (G_1 \Leftrightarrow A_1) \wedge \ldots \wedge (G_m \Leftrightarrow A_m) \wedge H_j[A_1, \ldots, A_m]}$

Iteriere diesen Schritt, bis er nicht mehr durchführbar ist.

Schnelle CNF-Herstellung

$$H_j[G_1, \ldots, G_m] \rightsquigarrow (G_1 \Leftrightarrow A_1) \wedge \ldots \wedge (G_m \Leftrightarrow A_m) \wedge H_j[A_1, \ldots, A_m]$$

Man sieht, dass Erfüllbarkeit erhalten bleibt.

Gegen-Beispiel zur Äquivalenz:

$$(A \vee \neg A) \rightsquigarrow (X \Leftrightarrow (A \vee \neg A)) \wedge X$$

Die rechte Formel ist keine Tautologie:

$X = 0$ ist eine Interpretation, die die rechte Formel falsch macht.

Eine Logelei aus „die Zeit“

Abianer sagen die Wahrheit, Bebianer lügen. Aussagen:

1. Knasi: Knisi ist Abianer.
2. Knesi: Wenn Knösi Bebianer, dann ist auch Knusi ein Abianer.
3. Knisi: Wenn Knusi Abianer, dann ist Knesi Bebianer.
4. Knosi: Knesi und Knüsi sind beide Abianer.
5. Knusi: Wenn Knüsi Abianer ist, dann ist auch Knisi Abianer.
6. Knösi: Entweder ist Knasi oder Knisi Abianer.
7. Knüsi: Knosi ist Abianer.

```
A <=> I
E <=>  (-OE => U)
I <=> (U => -E)
O <=> (E /\ UE)
U <=> (UE => I)
OE <=> (A XOR I)
UE <=> O
```

Logelei: Lösung

Die Eingabe in den Davis-Putnam-Algorithmus ergibt:

```
abianer1Expr = "((A <=> I) /\\ (E <=> (-OE => U)) /\\ (I <=> (U => -E))
               /\\ (O <=> (E /\\ UE)) /\\ (U <=> (UE => I))
                /\\ (OE <=> -(A <=> I)) /\\ (UE <=> O))"
```

```
 Resultat:
"Modell: -OE, -O, -UE, E, U, -I, -A"
```

Damit sind Knesi und Knusi Abianer, die anderen sind Bebianer.

Ein weiteres Rätsel von Raymond Smullyan:

Hier geht es um den Diebstahl von Salz.
Die Verdächtigen sind:
Lakai mit dem Froschgesicht, Lakai mit dem Fischgesicht, Herzbube.

Die Aussagen und die bekannten Tatsachen sind:

- Frosch: der Fisch wars
- Fisch: ich wars nicht
- Herzbube: ich wars
- Genau einer ist der Dieb
- höchstens einer hat gelogen

Kodierung des Problems

Wir verwenden Variablen mit folgenden Namen und Bedeutung:

FRW	Frosch sagt die Wahrheit
FIW	Fisch sagt die Wahrheit
HBW	Herzbube sagt die Wahrheit
FID	der Fisch ist der Dieb
FRD	der Frosch ist der Dieb
HBD	der Herzbube ist der Dieb

Kodierung des Frosch-Fisch-Problems

höchstens einer hat gelogen:

$$\begin{aligned}
\neg FRW &\Rightarrow FIW \wedge HBW \\
\neg FIW &\Rightarrow FRW \wedge HBW \\
\neg HBW &\Rightarrow FRW \wedge HIW
\end{aligned}$$

genau einer ist der Dieb:

$$FID \vee FRD \vee HBD$$

$$\begin{aligned}
FID &\Rightarrow \neg FRD \wedge \neg HBD \\
FRD &\Rightarrow \neg FID \wedge \neg HBD \\
HBD &\Rightarrow \neg FID \wedge \neg FRD
\end{aligned}$$

Die Aussagen:

$$\begin{aligned}
FRW &\Rightarrow FID \\
FIW &\Rightarrow \neg FID \\
HBW &\Rightarrow HBD
\end{aligned}$$

Frosch-Fisch: DP-Algorithmus:

```
dp "((-FRW => FIW /\\ HBW) /\\ (-FIW =>  FRW /\\  HBW)
    /\\ (-HBW =>  FRW /\\ HIW)
    /\\  (FID => -FRD /\\ -HBD) /\\ (FRD => -FID /\\ -HBD)
    /\\ (HBD => -FID /\\ -FRD) /\\  (FRW =>  FID)
    /\\ (FIW =>  -FID) /\\ (HBW =>  HBD))"
```

Die berechnete Lösung ist:
$HBD, -FID, HBW, FIW, -FRW, -FRD$
D.h FRW ist falsch, d.h. der Lakai mit dem Froschgesicht hat gelogen und der Herzbube war der Dieb.

Das n-Damen Problem

Platziere auf einem $n \times n$ Schachbrett
n Damen so, dass keine Dame die andere sofort schlagen kann

Zug- und Schlagmöglichkeiten der Schach-Dame:
oben, unten, rechts, links
diagonal noch rechts oben, links oben, rechts unten, links unten
Jeweils beliebig viele Felder

Das n-Damen Problem aussagenlogisch

Funktion zum Erzeugen der aussagenlogischen Bedingungen:

Ausgabe im Fall $n = 4$:

```
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12],
 [13, 14, 15, 16], [1, 5, 9, 13],
[2, 6, 10, 14], [3, 7, 11, 15], [4, 8, 12, 16],
[-1, -5], [-1, -9], [-1, -13],
[-1, -2], [-1, -6], [-1, -3], [-1, -11], [-1, -4], [-1, -16],
[-5, -9], [-5, -13],
[-5, -2], [-5, -6], [-5, -10], [-5, -7], [-5, -15], [-5, -8],
[-9, -13],
[-9, -6], [-9, -10], [-9, -14], [-9, -3], [-9, -11], [-9, -12],
[-13, -10], [-13, -14],
[-13, -7], [-13, -15], [-13, -4], [-13, -16], [-2, -6], [-2, -10],
```

```
[-2, -14],[-2, -3], [-2, -7], [-2, -4], [-2, -12], [-6, -10],
[-6, -14], [-6, -3],
[-6, -7], [-6, -11], [-6, -8], [-6, -16], [-10, -14], [-10, -7],
[-10, -11], [-10, -15],[-10, -4], [-10, -12], [-14, -11],
[-14, -15], [-14, -8], [-14, -16], [-3, -7],
[-3, -11], [-3, -15], [-3, -4], [-3, -8], [-7, -11], [-7, -15],
[-7, -4], [-7,-8],
[-7, -12], [-11, -15], [-11, -8], [-11, -12], [-11, -16], [-15, -12],
[-15, -16], [-4, -8], [-4, -12], [-4, -16], [-8, -12],
[-8, -16], [-12, -16]]
```

Das n-Damen Problem aussagenlogisch

Das Ergebnis der DP-Prozedur sind zwei Interpretationen:

```
[[-4, -8, -15, 5, -13, 14, -6, -2, 12, -9, -1, 3, -16, -10, -7, -11],
 [-4, 2, 8, -6, -1, 9, -12, -14, -13, -5, 15, -3, -16, -10, -7, -11]]
```

Zwei mögliche Platzierungen im Fall $n = 4$:

```
*DPexamples> dpqueensAlle 4

1.                          2.
- - D -                    - D - -
D - - -                    - - - D
- - - D                    D - - -
- D - -                    - - D -
```

Das n-Damen Problem aussagenlogisch

Der Aufruf `dpqueens 8` ergibt nach kurzer Zeit:

```
- - D - - - - -
- - - - - - D -
- D - - - - - -
- - - - - - - D
- - - - D - - -
D - - - - - - -
- - - D - - - -
- - - - - D - -
```