GOETHE UNIVERSITÄT FRANKFURT AM MAIN

www.uni-frankfurt.de

Logik in der Künstlichen Intelligenz (Logikbasierte Systeme der Wissensverarbeitung)

Aussagenlogik

Prof. Dr. M. Schmidt-Schauß

SoSe 2021

Stand der Folien: 3. Mai 2021

DPLL in Haskell

```
dpll :: [[Int]] -> Bool
dpll [] = False
dpll cnf
 | [] `elem` cnf = True
 | otherwise =
    case getUnit cnf of
     Just [x] ->
      dpll [delete (negate x) clause | clause <- cnf, not (x `elem` clause)]
     Nothing ->
      if not (null isolated) then
        dpll [clause | clause <- cnf, let (isolit:_)=isolated,
                                      not (isolit `elem` clause)]
      else (dpll ([lit]:cnf)) && (dpll ([negate lit]:cnf))
   where
    literals = (nub . sort . concat) cnf
    isolated = [lit | lit <- literals, not ((negate lit) `elem` literals)]
    ((lit:clause):_) = cnf
    getUnit []       = Nothing
    getUnit ([x]:_)  = Just [x]
    getUnit (_:xxs)  = getUnit xxs
```

Fallunterscheidung

- Verschiedene Heuristiken **welches Literal** gewählt wird.
- Gute Heuristik:

  Wähle Literal so, dass es in
  möglichst kurzen Klauseln vorkommt
- Erhöht Wahrscheinlichkeit, dass große Anteile der Klauselmenge in wenigen Schritten gelöscht werden.

Beispiel: Pfefferdieb

Wissen:

- $H \vee S \vee M$
- $H \Rightarrow \neg(S \vee M)$
- $S \Rightarrow \neg(H \vee M)$
- $M \Rightarrow \neg(H \vee S)$
- $\neg S \Rightarrow \neg H$
- $\neg H \Rightarrow \neg M$

Klauselmenge dazu:

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

Beispiel (2)

$\mathsf{DPLL}(\{\{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\})$

- Keine 1-Klauseln
- Keine isolierten Literale
- Daher Fallunterscheidung

(1) $\mathsf{DPLL}(\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\})$

$\wedge$

(2) $\mathsf{DPLL}(\{\{\neg S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\})$

Beispiel (3)

Fall (1):
DPLL$(\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\})$

- 1-Klausel $\{S\}$

Beispiel (3)

Fall (1):
DPLL($\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$)

- 1-Klausel $\{S\}$
- entferne alle Klauseln die $\{S\}$ enthalten:
  $\{\underline{\{S\}}, \underline{\{H, S, M\}}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \underline{\{S, \neg H\}}, \{H, \neg M\}\}$

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

Beispiel (3)

Fall (1):
DPLL($\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$)

- 1-Klausel $\{S\}$
- entferne alle Klauseln die $\{S\}$ enthalten:
  $\{\underline{\{S\}}, \underline{\{H, S, M\}}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \underline{\{S, \neg H\}}, \{H, \neg M\}\}$

  ergibt $\{\{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{H, \neg M\}\}$
  entferne $\neg S$ aus allen Klauseln: $\{\{\neg H, \underline{\neg S}\}, \{\neg H, \neg M\}, \{\underline{\neg S}, \neg M\}, \{H, \neg M\}\}$

  ergibt $(\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\})$
  Rekursiver Aufruf: DPLL($\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$)

Beispiel (3)

Fall (1):
DPLL($\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$)

- 1-Klausel $\{S\}$
- entferne alle Klauseln die $\{S\}$ enthalten:
  $\{\underline{\{S\}}, \underline{\{H, S, M\}}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \underline{\{S, \neg H\}}, \{H, \neg M\}\}$

  ergibt $\{\{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{H, \neg M\}\}$
  entferne $\neg S$ aus allen Klauseln: $\{\{\neg H, \underline{\neg S}\}, \{\neg H, \neg M\}, \{\underline{\neg S}, \neg M\}, \{H, \neg M\}\}$

  ergibt $(\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\})$
  Rekursiver Aufruf: DPLL($\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$)

DPLL($\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$)

- 1-Klausel $\{\neg H\}$
- Entfernen der Klauseln mit $\neg H$ und Löschen von H ergibt $\{\{\neg M\}\}$
  Rekursiver Aufruf: DPLL($\{\{\neg M\}\}$)

Beispiel (3)

Fall (1):
DPLL($\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$)

- 1-Klausel $\{S\}$
- entferne alle Klauseln die $\{S\}$ enthalten:
  $\{\underline{\{S\}}, \underline{\{H, S, M\}}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \underline{\{S, \neg H\}}, \{H, \neg M\}\}$

  ergibt $\{\{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{H, \neg M\}\}$
  entferne $\neg S$ aus allen Klauseln: $\{\{\neg H, \underline{\neg S}\}, \{\neg H, \neg M\}, \{\underline{\neg S}, \neg M\}, \{H, \neg M\}\}$

  ergibt $(\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\})$
  Rekursiver Aufruf: DPLL($\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$)

DPLL($\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$)

- 1-Klausel $\{\neg H\}$
- Entfernen der Klauseln mit $\neg H$ und Löschen von H ergibt $\{\{\neg M\}\}$
  Rekursiver Aufruf: DPLL($\{\{\neg M\}\}$)

DPLL($\{\{\neg M\}\}$) ergibt rekursiver Aufruf: DPLL($\{\}$)

Beispiel (3)

Fall (1):
DPLL($\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$)

- 1-Klausel $\{S\}$
- entferne alle Klauseln die $\{S\}$ enthalten:
  $\{\underline{\{S\}}, \underline{\{H, S, M\}}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \underline{\{S, \neg H\}}, \{H, \neg M\}\}$

  ergibt $\{\{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{H, \neg M\}\}$
  entferne $\neg S$ aus allen Klauseln: $\{\{\neg H, \underline{\neg S}\}, \{\neg H, \neg M\}, \{\underline{\neg S}, \neg M\}, \{H, \neg M\}\}$

  ergibt $(\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\})$
  Rekursiver Aufruf: DPLL($\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$)

DPLL($\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$)

- 1-Klausel $\{\neg H\}$
- Entfernen der Klauseln mit $\neg H$ und Löschen von H ergibt $\{\{\neg M\}\}$
  Rekursiver Aufruf: DPLL($\{\{\neg M\}\}$)

DPLL($\{\{\neg M\}\}$) ergibt rekursiver Aufruf: DPLL($\{\}$)
DPLL($\{\}$) ergibt `false` $\rightarrow$ erfüllbar,

Beispiel (3)

Fall (1):
DPLL($\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$)

- 1-Klausel $\{S\}$
- entferne alle Klauseln die $\{S\}$ enthalten:
  $\{\underline{\{S\}}, \underline{\{H, S, M\}}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \underline{\{S, \neg H\}}, \{H, \neg M\}\}$

  ergibt $\{\{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{H, \neg M\}\}$
  entferne $\neg S$ aus allen Klauseln: $\{\{\neg H, \underline{\neg S}\}, \{\neg H, \neg M\}, \{\underline{\neg S}, \neg M\}, \{H, \neg M\}\}$

  ergibt $(\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\})$
  Rekursiver Aufruf: DPLL($\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$)

DPLL($\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$)

- 1-Klausel $\{\neg H\}$
- Entfernen der Klauseln mit $\neg H$ und Löschen von H ergibt $\{\{\neg M\}\}$
  Rekursiver Aufruf: DPLL($\{\{\neg M\}\}$)

DPLL($\{\{\neg M\}\}$) ergibt rekursiver Aufruf: DPLL($\{\}$)
DPLL($\{\}$) ergibt `false` $\rightarrow$ erfüllbar, Modell?

DPLL: Modell generieren

1. Isolierte Literale werden als wahr angenommen.
2. Literale in 1-Klauseln werden ebenfalls als wahr angenommen.
3. Dadurch nicht belegte Variablen können für das vollständige Modell beliebig belegt werden

Komplexität

Im Worst Case:

- Exponentielle Laufzeit
- Exponentiell in: Anzahl der verschiedenen Variablen

Besser geht es nach aktuellem Stand des Wissens nicht, da SAT $\mathcal{NP}$-vollständig.

DPLL zum Problemlösen

- Idee: Kodiere Problem als Aussagenlogische Formel
- Modell entspricht Lösung des Problems
- Erst Umformung in CNF (geht mit schneller CNF), dann DPLL
- Oft: Kodierung direkt als CNF
- Wir betrachten einige Beispielanwendungen

Logelei aus der Zeit

Abianer sagen die Wahrheit, Bebianer Lügen.

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.

Logelei aus der Zeit

Abianer sagen die Wahrheit, Bebianer Lügen.

1. Knasi: Knisi ist Abianer.
   $knasi \iff knisi$
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.

Logelei aus der Zeit

Abianer sagen die Wahrheit, Bebianer Lügen.

1. Knasi: Knisi ist Abianer.
   $knasi \iff knisi$
2. Knesi: Wenn Knösi Bebianer, dann ist auch Knusi ein Abianer.
   $knesi \iff (\neg knoesi \implies knusi)$
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.

Logelei aus der Zeit

Abianer sagen die Wahrheit, Bebianer Lügen.

1. Knasi: Knisi ist Abianer.
   $knasi \iff knisi$
2. Knesi: Wenn Knösi Bebianer, dann ist auch Knusi ein Abianer.
   $knesi \iff (\neg knoesi \implies knusi)$
3. Knisi: Wenn Knusi Abianer, dann ist Knesi Bebianer.
   $knisi \iff (knusi \implies \neg knesi)$
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.

Logelei aus der Zeit

Abianer sagen die Wahrheit, Bebianer Lügen.

1. Knasi: Knisi ist Abianer.
   $knasi \iff knisi$
2. Knesi: Wenn Knösi Bebianer, dann ist auch Knusi ein Abianer.
   $knesi \iff (\neg knoesi \implies knusi)$
3. Knisi: Wenn Knusi Abianer, dann ist Knesi Bebianer.
   $knisi \iff (knusi \implies \neg knesi)$
4. Knosi: Knesi und Knüsi sind beide Abianer.
   $knosi \iff (knesi \wedge knuesi)$
5. Knusi: Wenn Knüsi Abianer ist, dann ist auch Knisi Abianer.
6. Knösi: Entweder ist Knasi oder Knisi Abianer.
7. Knüsi: Knosi ist Abianer.

Logelei aus der Zeit

Abianer sagen die Wahrheit, Bebianer Lügen.

1. Knasi: Knisi ist Abianer.
   $knasi \iff knisi$
2. Knesi: Wenn Knösi Bebianer, dann ist auch Knusi ein Abianer.
   $knesi \iff (\neg knoesi \implies knusi)$
3. Knisi: Wenn Knusi Abianer, dann ist Knesi Bebianer.
   $knisi \iff (knusi \implies \neg knesi)$
4. Knosi: Knesi und Knüsi sind beide Abianer.
   $knosi \iff (knesi \wedge knuesi)$
5. Knusi: Wenn Knüsi Abianer ist, dann ist auch Knisi Abianer.
   $knusi \iff (knuesi \implies knisi)$
6. Knösi: Entweder ist Knasi oder Knisi Abianer.
7. Knüsi: Knosi ist Abianer.

Logelei aus der Zeit

Abianer sagen die Wahrheit, Bebianer Lügen.

1. Knasi: Knisi ist Abianer.
   $knasi \iff knisi$
2. Knesi: Wenn Knösi Bebianer, dann ist auch Knusi ein Abianer.
   $knesi \iff (\neg knoesi \implies knusi)$
3. Knisi: Wenn Knusi Abianer, dann ist Knesi Bebianer.
   $knisi \iff (knusi \implies \neg knesi)$
4. Knosi: Knesi und Knüsi sind beide Abianer.
   $knosi \iff (knesi \wedge knuesi)$
5. Knusi: Wenn Knüsi Abianer ist, dann ist auch Knisi Abianer.
   $knusi \iff (knuesi \implies knisi)$
6. Knösi: Entweder ist Knasi oder Knisi Abianer.
   $knoesi \iff (knasi \,\mathsf{XOR}\, knisi)$
7. Knüsi: Knosi ist Abianer.

Logelei aus der Zeit

Abianer sagen die Wahrheit, Bebianer Lügen.

1. Knasi: Knisi ist Abianer.
   $knasi \iff knisi$
2. Knesi: Wenn Knösi Bebianer, dann ist auch Knusi ein Abianer.
   $knesi \iff (\neg knoesi \implies knusi)$
3. Knisi: Wenn Knusi Abianer, dann ist Knesi Bebianer.
   $knisi \iff (knusi \implies \neg knesi)$
4. Knosi: Knesi und Knüsi sind beide Abianer.
   $knosi \iff (knesi \wedge knuesi)$
5. Knusi: Wenn Knüsi Abianer ist, dann ist auch Knisi Abianer.
   $knusi \iff (knuesi \implies knisi)$
6. Knösi: Entweder ist Knasi oder Knisi Abianer.
   $knoesi \iff (knasi \,\mathsf{XOR}\, knisi)$
7. Knüsi: Knosi ist Abianer.
   $knuesi \iff knosi$

DPLL-Algorithmus liefert:

```
(knasi <=> knisi)
/\
(knesi <=> (-knoesi => knusi))
/\
(knisi <=> (knusi => -knesi))
/\
(knosi <=> (knesi /\ knuesi))
/\
(knusi <=> (knuesi => knisi))
/\
(knoesi <=> (-(knasi <=> knisi)))
/\
(knuesi <=> knosi)
```

DPLL-Algorithmus liefert:

```
(knasi <=> knisi)
/\
(knesi <=> (-knoesi => knusi))
/\
(knisi <=> (knusi => -knesi))
/\
(knosi <=> (knesi /\ knuesi))
/\
(knusi <=> (knuesi => knisi))
/\
(knoesi <=> (-(knasi <=> knisi)))
/\
(knuesi <=> knosi)
```

```
Das Ergebnis des DP-Algorithmus ist:
Fuer die berechnete Klauselmenge existiert ein Modell:
[-knuesi,-knosi,-knoesi,-knasi,knesi,knusi,-knisi]
```

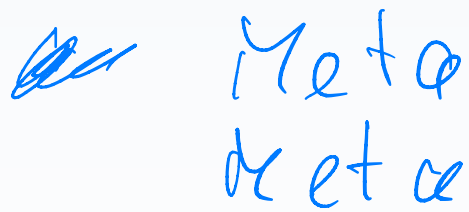
Knesi und Knusi sind Abianer,

Knüsi, Knosi, Knösi, Knisi sind Bebianer

Diebstahl von Salz

Verdächtige: Lakai mit Froschgesicht, Lakai mit Fischgesicht, Herzbube.

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



Diebstahl von Salz

Verdächtige: Lakai mit Froschgesicht, Lakai mit Fischgesicht, Herzbube.

- Frosch: der Fisch wars
  $froschSagtWahrheit \implies fischIstDieb$
- Fisch: ich wars nicht
- Herzbube: ich wars
- Genau einer ist der Dieb
- höchstens einer hat gelogen

Diebstahl von Salz

Verdächtige: Lakai mit Froschgesicht, Lakai mit Fischgesicht, Herzbube.

- Frosch: der Fisch wars
  $froschSagtWahrheit \implies fischIstDieb$
- Fisch: ich wars nicht
  $fischSagtWahrheit \implies \neg fischIstDieb$
- Herzbube: ich wars
- Genau einer ist der Dieb
- höchstens einer hat gelogen

Diebstahl von Salz

Verdächtige: Lakai mit Froschgesicht, Lakai mit Fischgesicht, Herzbube.

- Frosch: der Fisch wars
  $froschSagtWahrheit \implies fischIstDieb$
- Fisch: ich wars nicht
  $fischSagtWahrheit \implies \neg fischIstDieb$
- Herzbube: ich wars
  $bubeSagtWahrheit \implies bubeIstDieb$
- Genau einer ist der Dieb
- höchstens einer hat gelogen

Diebstahl von Salz

Verdächtige: Lakai mit Froschgesicht, Lakai mit Fischgesicht, Herzbube.

- Frosch: der Fisch wars

  $froschSagtWahrheit \implies fischIstDieb$

- Fisch: ich wars nicht

  $fischSagtWahrheit \implies \neg fischIstDieb$

- Herzbube: ich wars

  $bubeSagtWahrheit \implies bubeIstDieb$

- Genau einer ist der Dieb

  $$\begin{aligned} & (fischIstDieb \wedge \neg froschIstDieb \wedge \neg bubeIstDieb) \\ \vee\ & (\neg fischIstDieb \wedge froschIstDieb \wedge \neg bubeIstDieb) \\ \vee\ & (\neg fischIstDieb \wedge \neg froschIstDieb \wedge bubeIstDieb) \end{aligned}$$

- höchstens einer hat gelogen

Diebstahl von Salz

Verdächtige: Lakai mit Froschgesicht, Lakai mit Fischgesicht, Herzbube.

- Frosch: der Fisch wars

  $froschSagtWahrheit \implies fischIstDieb$

- Fisch: ich wars nicht

  $fischSagtWahrheit \implies \neg fischIstDieb$

- Herzbube: ich wars

  $bubeSagtWahrheit \implies bubeIstDieb$

- Genau einer ist der Dieb

  $$\begin{aligned} & (fischIstDieb \wedge \neg froschIstDieb \wedge \neg bubeIstDieb) \\ \vee\ & (\neg fischIstDieb \wedge froschIstDieb \wedge \neg bubeIstDieb) \\ \vee\ & (\neg fischIstDieb \wedge \neg froschIstDieb \wedge bubeIstDieb) \end{aligned}$$

- höchstens einer hat gelogen

  $$\begin{aligned} & (\neg froschSagtWahrheit \implies fischSagtWahrheit \wedge bubeSagtWahrheit) \\ \wedge\ & (\neg fischSagtWahrheit \implies froschSagtWahrheit \wedge bubeSagtWahrheit) \\ \wedge\ & (\neg bubeSagtWahrheit \implies froschSagtWahrheit \wedge fischSagtWahrheit) \end{aligned}$$

DPLL-Algorithmus liefert ...

```
(froschSagtWahrheit => fischIstDieb)
/\
(fischSagtWahrheit => -fischIstDieb)
/\
(bubeSagtWahrheit => bubeIstDieb)
/\
(    (fischIstDieb /\ -froschIstDieb /\ -bubeIstDieb)
  \/ (-fischIstDieb /\ froschIstDieb /\ -bubeIstDieb)
  \/ (-fischIstDieb /\ -froschIstDieb /\ bubeIstDieb) )
/\
(     (-froschSagtWahrheit => fischSagtWahrheit /\ bubeSagtWahrheit)
  /\  (-fischSagtWahrheit => froschSagtWahrheit /\ bubeSagtWahrheit)
  /\  (-bubeSagtWahrheit => froschSagtWahrheit /\ fischSagtWahrheit) )
```

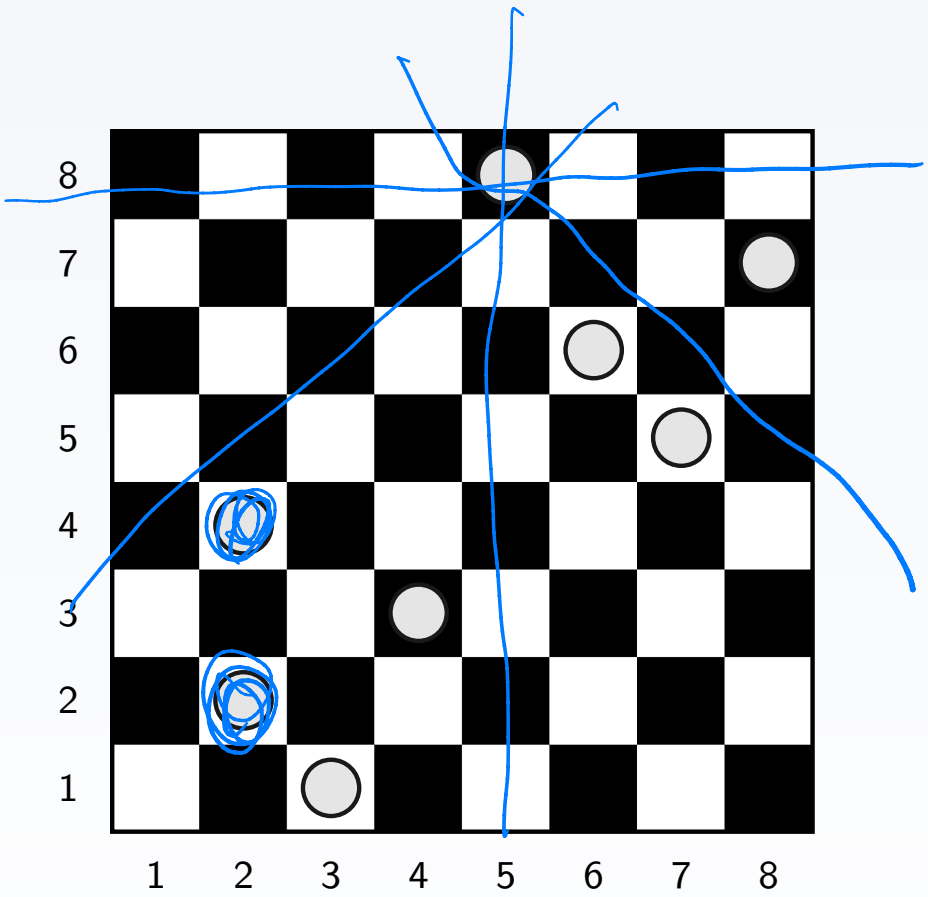
DPLL-Algorithmus liefert ...

```
(froschSagtWahrheit => fischIstDieb)
/\
(fischSagtWahrheit => -fischIstDieb)
/\
(bubeSagtWahrheit => bubeIstDieb)
/\
(    (fischIstDieb /\ -froschIstDieb /\ -bubeIstDieb)
  \/ (-fischIstDieb /\ froschIstDieb /\ -bubeIstDieb)
  \/ (-fischIstDieb /\ -froschIstDieb /\ bubeIstDieb) )
/\
(     (-froschSagtWahrheit => fischSagtWahrheit /\ bubeSagtWahrheit)
  /\  (-fischSagtWahrheit => froschSagtWahrheit /\ bubeSagtWahrheit)
  /\  (-bubeSagtWahrheit => froschSagtWahrheit /\ fischSagtWahrheit) )
```

```
Das Ergebnis des DP-Algorithmus ist:
Fuer die berechnete Klauselmenge existiert ein Modell:
[-froschIstDieb,bubeIstDieb,bubeSagtWahrheit,fischSagtWahrheit,
 -froschSagtWahrheit,-fischIstDieb]
```

$\Rightarrow$ Herzbube ist der Dieb, und Lakai mit Froschgesicht hat gelogen

N-Damen als SAT-Problem



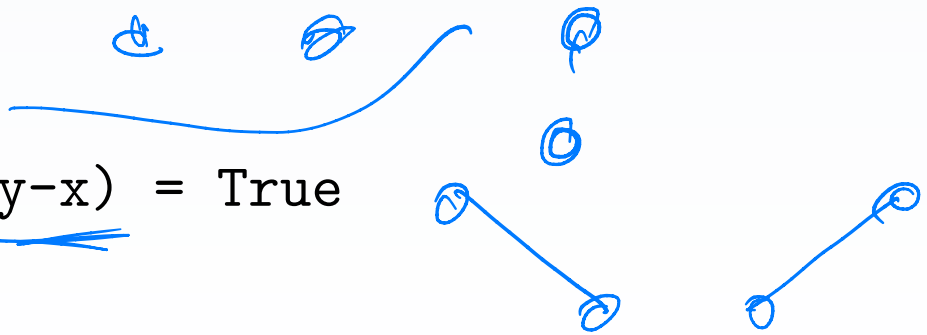
N-Damen als SAT-Problem (2)

Direkte Kodierung als CNF, Aussagenlogische Variablen sind Zahlen (negative Zahlen = negierte Literale)

```
nDamen n =
     (proZeileEineDame n)
  ++ (proSpalteEineDame n)
  ++ (bedrohendePaare n)


koordinateZuZahl (x,y) n =  (x-1)*n+y

proZeileEineDame n =
  [[koordinateZuZahl (i,j) n | j <- [1..n]] | i <- [1..n]]

proSpalteEineDame n =
  [[koordinateZuZahl (i,j) n | i <- [1..n]] | j <- [1..n]]
```

N-Damen als SAT-Problem (2)

```
bedrohendePaare n =
 [[negate (koordinateZuZahl  (x1,y1) n),
   negate (koordinateZuZahl  (x2,y2) n)]
    | x1 <- [1..n],
      y1 <- [1..n],
      x2 <- [1..n],
      y2 <- [1..n],
      (x1,y1) < (x2,y2),
      bedroht (x1,y1,x2,y2)]

bedroht (a,x,b,y)
 | a == b = True
 | x == y = True
 | abs (a-b) == abs (y-x) = True
 | otherwise = False
```



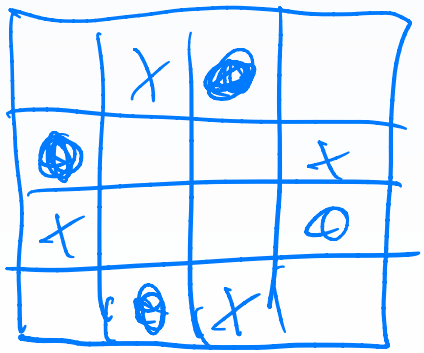
N-Damen als SAT-Problem (3)

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

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

Tatsächlich gibt es zwei Modelle:

```
*DPexamples> davisPutnamAlle (generate_nqueens 4)
[[-13,-16,15,9,-11,-14,-12,-10,8,-7,-6,-5,-4,-3,2,-1],
 [-15,-13,-9,-8,-6,-4,-2,-1,5,12,14,3,-16,-10,-7,-11]]
```



Nützliche Generatoren: Mindestens eine wahr

Sei $S = \{F_1, \ldots, F_n\}$ eine Menge von Formeln.

$$at\_least\_one(S) = (F_1 \vee \ldots \vee F_n)$$

Z.B. $at\_least\_one(\{X_1, X_2, X_3\}) = (X_1 \vee X_2 \vee X_3)$

Nützliche Generatoren: Höchstens eine wahr

Sei $S = \{F_1, \ldots, F_n\}$ eine Menge von Formeln.

$$at\_most\_one(S) = \bigwedge \{(\neg F_i \vee \neg F_j) \mid i \in \{1, \ldots, n\}, j \in \{1, \ldots, n\}, i \neq j\}$$

Z.B.

$$at\_most\_one(\{X_1, X_2, X_3\}) = \begin{array}{l}(\neg X_1 \vee \neg X_2) \wedge (\neg X_1 \vee \neg X_3) \wedge (\neg X_2 \vee \neg X_1) \\ \wedge (\neg X_2 \vee \neg X_3) \wedge (\neg X_3 \vee \neg X_1) \wedge (\neg X_3 \vee \neg X_2)\end{array}$$

Nützliche Generatoren: Höchstens eine wahr

Sei $S = \{F_1, \ldots, F_n\}$ eine Menge von Formeln.

$$at\_most\_one(S) = \bigwedge \{(\neg F_i \vee \neg F_j) \mid i \in \{1, \ldots, n\}, j \in \{1, \ldots, n\}, i \neq j\}$$

Z.B.

$$at\_most\_one(\{X_1, X_2, X_3\}) = \begin{array}{l}(\neg X_1 \vee \neg X_2) \wedge (\neg X_1 \vee \neg X_3) \wedge (\neg X_2 \vee \neg X_1) \\ \wedge (\neg X_2 \vee \neg X_3) \wedge (\neg X_3 \vee \neg X_1) \wedge (\neg X_3 \vee \neg X_2)\end{array}$$

Beachte: Man kann noch optimieren (Symmetrien entfernen):

$$at\_most\_one(S) = \bigwedge \{(\neg F_i \vee \neg F_j) \mid i \in \{1, \ldots, n\}, j \in \{1, \ldots, n\}, i < j\}$$

Z.B.

$$at\_most\_one(\{X_1, X_2, X_3\}) = (\neg X_1 \vee \neg X_2) \wedge (\neg X_1 \vee \neg X_3) \wedge (\neg X_2 \vee \neg X_3)$$

Nützliche Generatoren: Genau eine wahr

Sei $S = \{F_1, \ldots, F_n\}$ eine Menge von Formeln.

$$exactly\_one(S) = at\_least\_one(S) \wedge at\_most\_one(S)$$

Z.B.

$$\begin{aligned}
at_least_one(\{X_1, X_2, X_3\}) &= (X_1 \vee X_2 \vee X_3) \\
at_most_one(\{X_1, X_2, X_3\}) &= (\neg X_1 \vee \neg X_2) \wedge (\neg X_1 \vee \neg X_3) \wedge (\neg X_2 \vee \neg X_3) \\
exactly_one(\{X_1, X_2, X_3\}) &= (X_1 \vee X_2 \vee X_3) \wedge \\
&\quad (\neg X_1 \vee \neg X_2) \wedge (\neg X_1 \vee \neg X_3) \wedge (\neg X_2 \vee \neg X_3)
\end{aligned}$$

Beispiel: Klausuren verteilen

- s Studenten schreiben Klausuren
- k Klausurtypen
- Paare von benachbart sitzenden Studenten gegeben
- Problem: Welcher Student bekommt welchen Klausurtyp
- So dass: Keine benachbarten Studenten bekommen gleiche Klausur

Kodiere in CNF, so dass Modell eine Zuordnung: Student $\leftrightarrow$ Klausurtyp liefert

Beispiel: Klausuren verteilen

s_i^j = Student i schreibt Klausurtyp j

$klausur\_formel(s, k, benachbart) =$

Jeder Student erhält genau eine Klausur:

$$\underbrace{\bigwedge_{i=1}^{s}}_{\text{alle Studenten}} \underbrace{exactly\_one(\{s_i^j \mid j \in \{1..k\}\})}_{\text{Student } i \text{ genau eine Klausur}}$$

Benachbarte Studenten, nicht die gleiche Klausur:

$$\wedge \bigwedge_{(a,b) \in benachbart} \bigwedge_{j=1}^{k} (\neg s_a^j \vee \neg s_b^j)$$

Verallgemeinerung: K aus N

- $at\_most(K, S)$: höchstens K viele Formeln aus S erfüllt
- $at\_least(K, S)$: mindestens K viele Formeln aus S erfüllt
- $exactly(K, S)$: genau K viele Formeln aus S erfüllt

Verallgemeinerung: K aus N

- $at\_most(K, S)$: höchstens K viele Formeln aus S erfüllt
- $at\_least(K, S)$: mindestens K viele Formeln aus S erfüllt
- $exactly(K, S)$: genau K viele Formeln aus S erfüllt

Vorarbeit: Alle Teilmengen der Mächtigkeit K

$$all\_subsets(S, K) = \{S' \subseteq S \mid |S'| = K\}$$

Z.B. Rekursive Berechnung:

$$\begin{aligned}
all_subsets(S, 0) &= \{\{\}\} \\
all_subsets(S, K) &= \{S\}, \text{wenn } |S| = K \\
all_subsets(\{s\} \dot\cup S, K) &= all_subsets(S, K) \\
&\quad \cup \{\{s\} \cup S' \mid S' \in all_subsets(S, K-1)\}
\end{aligned}$$

Höchstens K wahr

$$at\_most(K, S) = \bigwedge \left\{ \underbrace{\neg(\bigwedge S')}_{\text{nicht alle wahr}} \mid \underbrace{S'}_{K+1 \text{ Formeln}} \in all\_subsets(S, K+1) \right\}$$

$$= \bigwedge \left\{ \underbrace{\neg(F_1 \wedge \ldots \wedge F_{k+1})}_{\text{nicht alle wahr}} \mid \{F_1, \ldots, F_{K+1}\} \in all\_subsets(S, K+1) \right\}$$

$$= \bigwedge \left\{ \underbrace{(\neg F_1 \vee \ldots \vee \neg F_{k+1})}_{\text{mind. 1 falsch}} \mid \{F_1, \ldots, F_{K+1}\} \in all\_subsets(S, K+1) \right\}$$

In CNF (wenn F_i Literale sind)

Mindestens K wahr

$$at\_least(K, S) = \bigvee \left\{ \underbrace{\bigwedge S'}_{K \text{ wahr}} \mid S' \in all\_subsets(S, K) \right\}$$

Z.B.

$$\begin{aligned} at\_least(3, \{X_1, X_2, X_3, X_4\}) \quad = \quad & (X_1 \wedge X_2 \wedge X_3) \vee (X_1 \wedge X_2 \wedge X_4) \\ & \vee (X_1 \wedge X_3 \wedge X_4) \vee (X_2 \wedge X_3 \wedge X_4) \end{aligned}$$

Nachteil: Nicht in CNF (wenn S nur Literale enthält)

Mindestens K wahr (2)

$$
\begin{aligned}
&at\_least(K, S)\\
&= at\_least\_one(S)\\
&\quad \wedge \bigwedge \{f \Rightarrow \bigvee S' \mid f \in S, S' \in all\_subsets(S \setminus \{f\}, |S| - (K-1))\}\\
&= at\_least\_one(S)\\
&\quad \wedge \bigwedge \{\neg f \vee \bigvee S' \mid f \in S, S' \in all\_subsets(S \setminus \{f\}, |S| - (K-1))\}
\end{aligned}
$$

Idee dabei: mind. 1 Formel aus S wahr,
und wenn eine wahr, dann noch $K-1$ weitere wahr...
Z.B.

$$
\begin{aligned}
&at\_least(3, \{X_1, X_2, X_3, X_4\}) =\\
&(X_1 \vee X_2 \vee X_3 \vee X_4)\\
&\wedge(X_1 \Rightarrow (X_2 \vee X_3)) \wedge (X_1 \Rightarrow (X_2 \vee X_4)) \wedge (X_1 \Rightarrow (X_3 \vee X_4))\\
&\wedge(X_2 \Rightarrow (X_1 \vee X_3)) \wedge (X_2 \Rightarrow (X_1 \vee X_4)) \wedge (X_2 \Rightarrow (X_3 \vee X_4))\\
&\wedge(X_3 \Rightarrow (X_1 \vee X_2)) \wedge (X_3 \Rightarrow (X_1 \vee X_4)) \wedge (X_3 \Rightarrow (X_2 \vee X_4))\\
&\wedge(X_4 \Rightarrow (X_1 \vee X_2)) \wedge (X_4 \Rightarrow (X_1 \vee X_3)) \wedge (X_4 \Rightarrow (X_2 \vee X_3))
\end{aligned}
$$

Mindestens K wahr (2)

$$
\begin{aligned}
&at\_least(K, S)\\
&= at\_least\_one(S)\\
&\quad \wedge \bigwedge \{f \Rightarrow \bigvee S' \mid f \in S, S' \in all\_subsets(S \setminus \{f\}, |S| - (K-1))\}\\
&= at\_least\_one(S)\\
&\quad \wedge \bigwedge \{\neg f \vee \bigvee S' \mid f \in S, S' \in all\_subsets(S \setminus \{f\}, |S| - (K-1))\}
\end{aligned}
$$

Idee dabei: mind. 1 Formel aus S wahr,
und wenn eine wahr, dann noch $K-1$ weitere wahr...
Z.B.

$$
\begin{aligned}
&at\_least(3, \{X_1, X_2, X_3, X_4\}) =\\
&(X_1 \vee X_2 \vee X_3 \vee X_4)\\
&\wedge(\neg X_1 \vee (X_2 \vee X_3)) \wedge (\neg X_1 \vee (X_2 \vee X_4)) \wedge (\neg X_1 \vee (X_3 \vee X_4))\\
&\wedge(\neg X_2 \vee (X_1 \vee X_3)) \wedge (\neg X_2 \vee (X_1 \vee X_4)) \wedge (\neg X_2 \vee (X_3 \vee X_4))\\
&\wedge(\neg X_3 \vee (X_1 \vee X_2)) \wedge (\neg X_3 \vee (X_1 \vee X_4)) \wedge (\neg X_3 \vee (X_2 \vee X_4))\\
&\wedge(\neg X_4 \vee (X_1 \vee X_2)) \wedge (\neg X_4 \vee (X_1 \vee X_3)) \wedge (\neg X_4 \vee (X_2 \vee X_3))
\end{aligned}
$$

Genau K wahr

$$exactly(K, S) = at\_least(K, S) \wedge at\_most(K, S)$$

Beispiel: Logelei aus der Zeit

Tom, ein Biologiestudent, sitzt verzweifelt in der Klausur, denn er hat vergessen, das Kapitel über die Wolfswürmer zu lernen. Das Einzige, was er weiß, ist, dass es bei den Multiple-Choice-Aufgaben immer **genau 3 korrekte** Antworten gibt. Und dies sind die angebotenen Antworten:

a) Wolfswürmer werden oft von Igelwürmern gefressen.
b) Wolfswürmer meiden die Gesellschaft von Eselswürmern.
c) Wolfswürmer ernähren sich von Lammwürmern.
d) Wolfswürmer leben in der banesischen Tundra.
e) Wolfswürmer gehören zur Gattung der Hundswürmer.
f) Wolfswürmer sind grau gestreift.
g) Genau eine der beiden Aussagen b) und e) ist richtig.
h) Genau eine der beiden Aussagen a) und d) ist richtig.
i) Genau eine der beiden Aussagen c) und h) ist richtig.
j) Genau eine der beiden Aussagen f) und i) ist richtig.
k) Genau eine der beiden Aussagen c) und d) ist richtig.
l) Genau eine der beiden Aussagen d) und h) ist richtig.

Können Sie Tom helfen, die richtigen Aussagen herauszufinden?

Beispiel: Logelei aus der Zeit

A,B,. . . ,L = entsprechende Aussage ist wahr

g) Genau eine der beiden Aussagen b) und e) ist richtig.

h) Genau eine der beiden Aussagen a) und d) ist richtig.

i) Genau eine der beiden Aussagen c) und h) ist richtig.

j) Genau eine der beiden Aussagen f) und i) ist richtig.

k) Genau eine der beiden Aussagen c) und d) ist richtig.

l) Genau eine der beiden Aussagen d) und h) ist richtig.

- genau 3 korrekte Antworten

Beispiel: Logelei aus der Zeit

A,B,. . . ,L = entsprechende Aussage ist wahr

g) Genau eine der beiden Aussagen b) und e) ist richtig.
$G \iff B$ XOR E

h) Genau eine der beiden Aussagen a) und d) ist richtig.

i) Genau eine der beiden Aussagen c) und h) ist richtig.

j) Genau eine der beiden Aussagen f) und i) ist richtig.

k) Genau eine der beiden Aussagen c) und d) ist richtig.

l) Genau eine der beiden Aussagen d) und h) ist richtig.

- genau 3 korrekte Antworten

Beispiel: Logelei aus der Zeit

A,B,. . . ,L = entsprechende Aussage ist wahr

g) Genau eine der beiden Aussagen b) und e) ist richtig.
$G \iff B$ XOR E

h) Genau eine der beiden Aussagen a) und d) ist richtig.
$H \iff A$ XOR D

i) Genau eine der beiden Aussagen c) und h) ist richtig.

j) Genau eine der beiden Aussagen f) und i) ist richtig.

k) Genau eine der beiden Aussagen c) und d) ist richtig.

l) Genau eine der beiden Aussagen d) und h) ist richtig.

- genau 3 korrekte Antworten

Beispiel: Logelei aus der Zeit

A,B,. . . ,L = entsprechende Aussage ist wahr

g) Genau eine der beiden Aussagen b) und e) ist richtig.
$G \iff B$ XOR E

h) Genau eine der beiden Aussagen a) und d) ist richtig.
$H \iff A$ XOR D

i) Genau eine der beiden Aussagen c) und h) ist richtig.
$I \iff C$ XOR H

j) Genau eine der beiden Aussagen f) und i) ist richtig.

k) Genau eine der beiden Aussagen c) und d) ist richtig.

l) Genau eine der beiden Aussagen d) und h) ist richtig.

- genau 3 korrekte Antworten

Beispiel: Logelei aus der Zeit

A,B,. . . ,L = entsprechende Aussage ist wahr

g) Genau eine der beiden Aussagen b) und e) ist richtig.
$G \iff B$ XOR E

h) Genau eine der beiden Aussagen a) und d) ist richtig.
$H \iff A$ XOR D

i) Genau eine der beiden Aussagen c) und h) ist richtig.
$I \iff C$ XOR H

j) Genau eine der beiden Aussagen f) und i) ist richtig.
$J \iff F$ XOR I

k) Genau eine der beiden Aussagen c) und d) ist richtig.

l) Genau eine der beiden Aussagen d) und h) ist richtig.

- genau 3 korrekte Antworten

Beispiel: Logelei aus der Zeit

A,B,. . . ,L = entsprechende Aussage ist wahr

g) Genau eine der beiden Aussagen b) und e) ist richtig.
$G \iff B$ XOR E

h) Genau eine der beiden Aussagen a) und d) ist richtig.
$H \iff A$ XOR D

i) Genau eine der beiden Aussagen c) und h) ist richtig.
$I \iff C$ XOR H

j) Genau eine der beiden Aussagen f) und i) ist richtig.
$J \iff F$ XOR I

k) Genau eine der beiden Aussagen c) und d) ist richtig.
$K \iff C$ XOR D

l) Genau eine der beiden Aussagen d) und h) ist richtig.

- genau 3 korrekte Antworten

Beispiel: Logelei aus der Zeit

A,B,. . . ,L = entsprechende Aussage ist wahr

g) Genau eine der beiden Aussagen b) und e) ist richtig.
$G \iff B \text{ XOR } E$

h) Genau eine der beiden Aussagen a) und d) ist richtig.
$H \iff A \text{ XOR } D$

i) Genau eine der beiden Aussagen c) und h) ist richtig.
$I \iff C \text{ XOR } H$

j) Genau eine der beiden Aussagen f) und i) ist richtig.
$J \iff F \text{ XOR } I$

k) Genau eine der beiden Aussagen c) und d) ist richtig.
$K \iff C \text{ XOR } D$

l) Genau eine der beiden Aussagen d) und h) ist richtig.
$L \iff D \text{ XOR } H$

- genau 3 korrekte Antworten

Beispiel: Logelei aus der Zeit

A,B,. . . ,L = entsprechende Aussage ist wahr

g) Genau eine der beiden Aussagen b) und e) ist richtig.
$G \iff B$ XOR E

h) Genau eine der beiden Aussagen a) und d) ist richtig.
$H \iff A$ XOR D

i) Genau eine der beiden Aussagen c) und h) ist richtig.
$I \iff C$ XOR H

j) Genau eine der beiden Aussagen f) und i) ist richtig.
$J \iff F$ XOR I

k) Genau eine der beiden Aussagen c) und d) ist richtig.
$K \iff C$ XOR D

l) Genau eine der beiden Aussagen d) und h) ist richtig.
$L \iff D$ XOR H

- genau 3 korrekte Antworten
  $exactly(3, \{A, B, C, D, E, F, G, H, I, J, K, L\})$

Lösen mit DPLL

Direkte Kodierung mit 1,...,12 statt A,...,B ergibt:

```
*DPexamples> davisPutnam cwuermer
[-11,-10,-9,-7,-6,-5,-2,3,-12,-1,8,4]
```

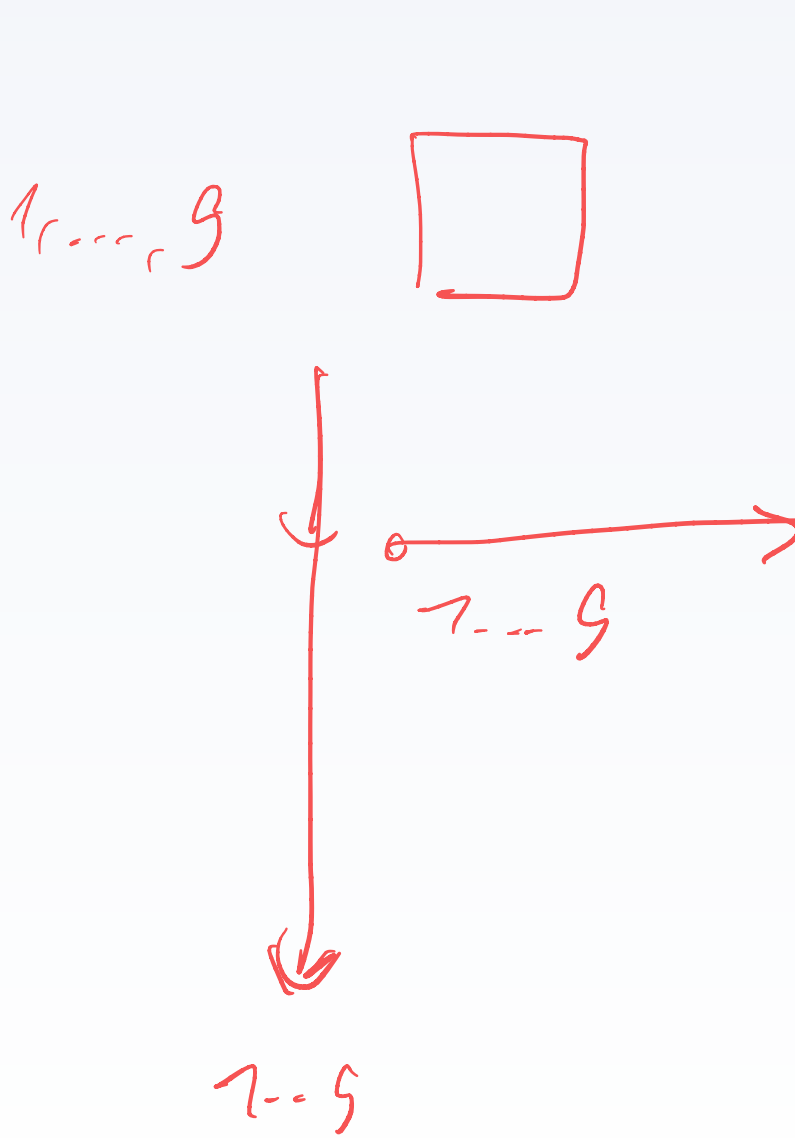
D.h. 3,4,8 sind wahr = C,D,H sind wahr

c) Wolfswürmer ernähren sich von Lammwürmern.
d) Wolfswürmer leben in der banesischen Tundra.
h) Genau eine der beiden Aussagen a) und d) ist richtig.

Sudoku-Lösen mit DPLL

Sudoku:

		6	4		1	5		
		3	6	5				
5	8			2		6		
4	6		8					3
	3	5					2	
2					3	9	8	
9		1			5			
				4	6			5
			1				7	



Sudoku-Lösen mit DPLL

Sudoku:

7	9	6	4	8	1	5	3	2
1	2	3	6	5	9	8	4	7
5	8	4	3	2	7	6	9	1
4	6	9	8	1	2	7	4	3
8	3	5	7	9	4	1	2	6
2	1	7	5	6	3	9	8	4
9	4	1	2	7	5	3	6	8
3	7	8	9	4	6	2	1	5
6	5	2	1	3	8	4	7	9

Sudoku-Lösen mit DPLL

Sudoku:

7	9	6	4	8	1	5	3	2
1	2	3	6	5	9	8	4	7
5	8	4	3	2	7	6	9	1
4	6	9	8	1	2	7	4	3
8	3	5	7	9	4	1	2	6
2	1	7	5	6	3	9	8	4
9	4	1	2	7	5	3	6	8
3	7	8	9	4	6	2	1	5
6	5	2	1	3	8	4	7	9

Sudoku-Lösen mit DPLL

Sudoku:

7	9	6	4	8	1	5	3	2
1	2	3	6	5	9	8	4	7
5	8	4	3	2	7	6	9	1
4	6	9	8	1	2	7	4	3
8	3	5	7	9	4	1	2	6
2	1	7	5	6	3	9	8	4
9	4	1	2	7	5	3	6	8
3	7	8	9	4	6	2	1	5
6	5	2	1	3	8	4	7	9

Sudoku-Lösen mit DPLL

Sudoku:

7	9	6	4	8	1	5	3	2
1	2	3	6	5	9	8	4	7
5	8	4	3	2	7	6	9	1
4	6	9	8	1	2	7	4	3
8	3	5	7	9	4	1	2	6
2	1	7	5	6	3	9	8	4
9	4	1	2	7	5	3	6	8
3	7	8	9	4	6	2	1	5
6	5	2	1	3	8	4	7	9

Kodierung von Sudokus als Klauselmenge

- Direkt Ganzzahlen als Variablennamen
- Negative Zahlen $=$ negierte Variable
- Dreistellige Zahlen $XYV \in \{111, 112, \ldots, 999\}$
- $X =$ Zeile
- $Y =$ Spalte
- $V =$ Zahl die in der Zelle stehen kann in $\{1, \ldots, 9\}$
- D.h. pro Zelle 9 Variablen, von den genau eine Wahr sein muss
- Insgesamt 729 Variablen

Kodierung von Sudokus als Klauselmenge

Eingabe: Sudoku (teilweise gefüllt)

Erzeuge Klauselmenge:

$$\begin{aligned} \text{Klauselmenge} = \quad & \text{Startbelegung} \\ & \cup \text{ FelderEindeutigBelegt} \\ & \cup \text{ ZeilenBedingung} \\ & \cup \text{ SpaltenBedingung} \\ & \cup \text{ QuadratBedingung} \end{aligned}$$

Startbelegung:

- Gegebene Zahlen werden auf wahr gesetzt:
  Startbelegung: Wenn in Feld (x, y) die Zahl z steht:
  Füge 1-Klausel $\{xyz\}$ hinzu

Kodierung von Sudokus als Klauselmenge

$$\text{FelderEindeutigBelegt} = \bigwedge_{X=1}^{9} \bigwedge_{Y=1}^{9} exactly\_one(\{XY1, \ldots, XY9\})$$

$$\text{ZeilenBedingung} = \bigwedge_{X=1}^{9} \bigwedge_{V=1}^{9} at\_most\_one(\{X1V, \ldots, X9V\})$$

$$\text{SpaltenBedingung} = \bigwedge_{Y=1}^{9} \bigwedge_{V=1}^{9} at\_most\_one(\{1YV, \ldots, 9YV\})$$

Kodierung von Sudokus als Klauselmenge (2)

QuadratBedingung $=$

$$\bigwedge_{V=1}^{9} \left(\begin{array}{l} at\_most\_one(\{11V, 12V, 13V, 21V, 22V, 23V, 31V, 32V, 33V\}) \wedge \\ at\_most\_one(\{14V, 15V, 16V, 24V, 25V, 26V, 34V, 35V, 36V\}) \wedge \\ at\_most\_one(\{17V, 18V, 19V, 27V, 28V, 29V, 37V, 38V, 39V\}) \wedge \\ at\_most\_one(\{41V, 42V, 43V, 51V, 52V, 53V, 61V, 62V, 63V\}) \wedge \\ at\_most\_one(\{44V, 45V, 46V, 54V, 55V, 56V, 64V, 65V, 66V\}) \wedge \\ at\_most\_one(\{47V, 48V, 49V, 57V, 58V, 59V, 67V, 68V, 69V\}) \wedge \\ at\_most\_one(\{71V, 72V, 73V, 81V, 82V, 83V, 91V, 92V, 93V\}) \wedge \\ at\_most\_one(\{74V, 75V, 76V, 84V, 85V, 86V, 94V, 95V, 96V\}) \wedge \\ at\_most\_one(\{77V, 78V, 79V, 87V, 88V, 89V, 97V, 98V, 99V\}) \end{array} \right)$$

Mögliche Anwendung: Produktkonfigurator / Auto-Konfigurator

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

"Die Produktkonfiguration beschreibt das Zusammensetzen eines Produktes aus vorgegebenen Produktkomponenten (sogenannte **Selektion und Kombination**) und die Selektion inhaltlicher Ausprägungen der Komponenteneigenschaften (sogenannte **Parametrisierung**) unter Einhaltung der **Konfigurationsregeln**. Die **Konfigurationsmöglichkeiten** ergeben sich aus den Selektions-, Kombinations- und Parametrisierungsmöglichkeiten eines Produktes eingeschränkt durch die **Konfigurationsregeln**"

Mögliche Anwendung: Produktkonfigurator / Auto-Konfigurator

Aussagenlogik und DPLL-Beweiser mit Modellausgabe kann hilfreich sein:
Es gibt nur **endlich** viele Möglichkeiten:

- Farben
- Ja/nein Entscheidungen: Klimaanlage j/n: Ja /nein
- Liste von Varianten (zB Motoren)
- Reeller Wert (zB PS-Zahl): kann diskretisiert werden.

Schwieriger:

- Preis Obergrenze: Durch Addieren der Komponentenpreise

Mögliche Funktionen:

- Überpfüfung der Konfigurationsregeln
- Ausgabe der möglichen Varinanten.
- Testen des Systems, z.B bei Erweiterung / Anpassungen