

# Entscheidungsverfahren mit Anwendungen in der Softwareverifikation

## **V: Einführung SMT**

---

Carsten Sinz  
Institut für Theoretische Informatik

19.11.2019

- Abstraktes DPLL
- Von SAT zu SMT

- **Regel-basierte Formulierung von DPLL**

- sauber
- einfach
- flexibel

- **Zustände:**

- `Fail`
- $M \parallel F$ 
  - $F$ : Aussagenlogische Formel in CNF
  - $M$ : Liste von Literalen
- $M$  wird als partielle Zuweisung  $M : \text{Var}(F) \longrightarrow \{ 0, 1 \}$  interpretiert

- **Regeln:**

- $M \parallel F \implies M' \parallel F'$  oder  $M \parallel F \implies \text{Fail}$

- **UnitPropagate:**

$$M \parallel F, C \vee l \implies Ml \parallel F, C \vee l \quad \text{falls} \begin{cases} M \models \neg C \\ l \text{ undefiniert in } M \end{cases}$$

- **Decide:**

$$M \parallel F \implies Ml^d \parallel F \quad \text{falls} \begin{cases} l \text{ oder } \neg l \text{ kommt in } F \text{ vor} \\ l \text{ undefiniert in } M \end{cases}$$

- **Backtrack:**

$$Ml^d N \parallel F, C \implies M\neg l \parallel F, C \quad \text{falls} \quad \begin{cases} Ml^d N \models \neg C \\ N \text{ enthält kein Literal} \\ \text{der Form } k^d \end{cases}$$

- **Fail:**

$$M \parallel F, C \implies \text{Fail} \quad \text{falls} \quad \begin{cases} M \models \neg C \\ M \text{ enthält kein Literal der Form } l^d \end{cases}$$

$\emptyset$	$\parallel$	$\neg A \vee \neg B, B \vee C, \neg A \vee \neg C \vee D, B \vee \neg C \vee \neg D, A \vee D$	$\implies$	Decide
$A^d$	$\parallel$	$\neg A \vee \neg B, B \vee C, \neg A \vee \neg C \vee D, B \vee \neg C \vee \neg D, A \vee D$	$\implies$	UnitProp
$A^d, \neg B$	$\parallel$	$\neg A \vee \neg B, B \vee C, \neg A \vee \neg C \vee D, B \vee \neg C \vee \neg D, A \vee D$	$\implies$	$2 \times$ UnitProp
$A^d, \neg B, C, D$	$\parallel$	$\neg A \vee \neg B, B \vee C, \neg A \vee \neg C \vee D, B \vee \neg C \vee \neg D, A \vee D$	$\implies$	Backtrack
$\neg A$	$\parallel$	$\neg A \vee \neg B, B \vee C, \neg A \vee \neg C \vee D, B \vee \neg C \vee \neg D, A \vee D$	$\implies$	UnitProp
$\neg A, D$	$\parallel$	$\neg A \vee \neg B, B \vee C, \neg A \vee \neg C \vee D, B \vee \neg C \vee \neg D, A \vee D$	$\implies$	Decide
$\neg A, D, \neg C^d$	$\parallel$	$\neg A \vee \neg B, B \vee C, \neg A \vee \neg C \vee D, B \vee \neg C \vee \neg D, A \vee D$	$\implies$	UnitProp
$\neg A, D, \neg C^d, B$	$\parallel$	$\neg A \vee \neg B, B \vee C, \neg A \vee \neg C \vee D, B \vee \neg C \vee \neg D, A \vee D$	$\not\Rightarrow$	

- **Ergebnis:**  $A = C = 0, B = D = 1$  ist ein Modell der Formel

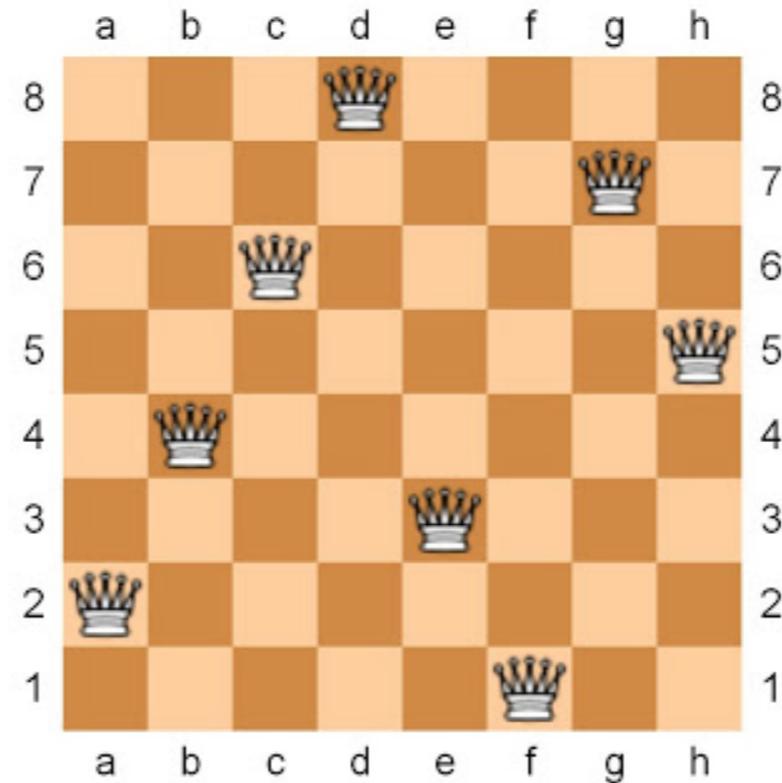
- **Satz:**

Alle Ableitungen  $\emptyset \parallel F \implies S_1 \implies \dots$  sind endlich.

- **Satz:**

Sei  $\emptyset \parallel F \implies S_1 \implies \dots \implies S$  eine maximale Ableitung. Dann gilt:

- $F$  ist unerfüllbar gdw.  $s = \text{Fail}$
- Falls  $S$  die Form  $M \parallel F$  hat, dann ist  $M$  ein Modell von  $F$



Platziere  $n$  Damen so auf einem  $n \times n$ -Schachbrett dass keine Dame eine andere Dame schlagen kann

- Führe eine 0/1-Variable pro Feld ein

$$\begin{array}{ccc} X_{1,1} & \cdots & X_{1,N} \\ \vdots & & \vdots \\ X_{N,1} & \cdots & X_{N,N} \end{array}$$

- Genau eine Dame pro Zeile/Spalte:

$$\sum_{j=1}^N X_{1,j} = 1 \wedge \dots \wedge \sum_{j=1}^N X_{N,j} = 1 \wedge \sum_{i=1}^N X_{i,1} = 1 \wedge \dots \wedge \sum_{i=1}^N X_{i,N} = 1$$

- Höchstens eine Dame pro Diagonale:

$$\begin{aligned} X_{1,1} + X_{2,2} + \dots + X_{N,N} &\leq 1 \\ X_{1,2} + X_{2,3} + \dots + X_{N-1,N} &\leq 1 \\ X_{1,3} + X_{2,4} + \dots + X_{N-2,N} &\leq 1 \end{aligned}$$

- Logik benutzt Schreibweisen aus der **Arithmetik**
- Diese müssen nach SAT **kodiert** werden
- $\sum_{j=1}^N X_{1,j} = 1$  kann kodiert werden als  
 $(X_{1,1} \vee \dots \vee X_{1,N}) \wedge \neg(X_{1,1} \wedge X_{1,2}) \wedge \neg(X_{1,1} \wedge X_{1,3}) \wedge \dots \wedge \neg(X_{1,N-1} \wedge X_{1,N})$
- $\sum_{j=1}^N X_{1,j} \leq 1$  kann kodiert werden als  
 $\neg(X_{1,1} \wedge X_{1,2}) \wedge \neg(X_{1,1} \wedge X_{1,3}) \wedge \dots \wedge \neg(X_{1,N-1} \wedge X_{1,N})$
- **Beobachtung:**  
Aussagenlogik ist wie Maschinensprache!

- **Wunsch:** Logik, die Arithmetik beherrscht
- n-Damen Problem benötigt dann (fast) keine weitere Kodierung
  - $X_{i,j} = 0 \vee X_{i,j} = 1$  für alle  $X_{i,j}$
- **Wunsch:** Unterstützung weiterer eingebauter Datentypen
- **Lösung:** Benutze SMT! (Satisfiability Modulo Theories)
- **Beobachtung:** SMT ist wie Haskell / OCaml / Scala / Swift / ...