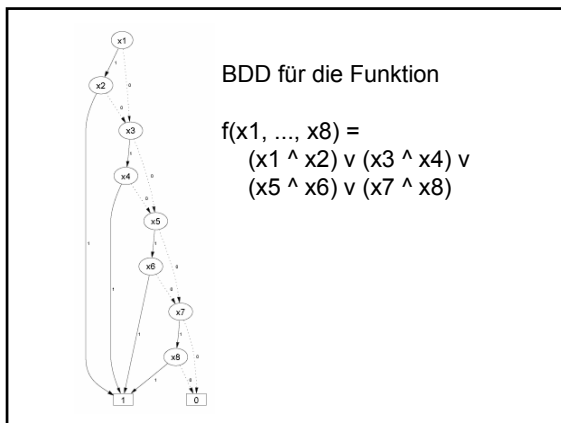


## SAT-basiertes model-checking

Michael Bast  
Seminar Systementwurf  
05. Februar 2007

## Symbolisches Model-Checking

Symbolische Model-Checker basieren entweder auf [Binary Decision Diagrams](#) (z.B. für [CTL-Formeln](#)) oder auf [SAT-Solvern](#) (z.B. für [LTL-Formeln](#)).



- Ansatz: Übersetze das Model Checking Problem in ein aussagenlogisches Erfüllbarkeitsproblem (Formel  $\phi$ ) und löse dieses.
- Viele Algorithmen setzen  $\phi$  in bestimmter Form voraus, meist CNF (Klauselmenge).

- Konjunktive Normalform (KNF)  
→ Theorie VL
- SAT steht für „satisfying“  
da es darum geht, eine erfüllende Belegung zu finden.

## Das Problem SAT

geg: aussagenlogischer Ausdruck  $\phi$

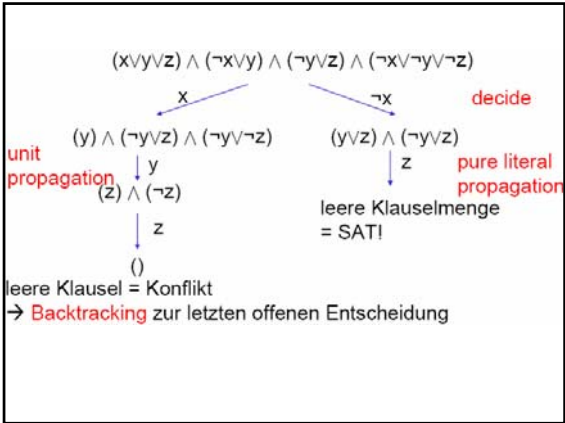
Frage: Ist  $\phi$  erfüllbar?

- Suche nach einer erfüllenden Belegung
- Ausgangspunkt: Algorithmus von Davis-Putnam aus den 60ern

### Davis-Putnam-Algorithmus

```

DP(K)      K – Klauselmenge
IF K = ∅ THEN RETURN SAT
IF () ∈ K THEN RETURN UNSAT
IF κ = ( ...∨1∨... ) ∈ K THEN RETURN DP(K \ {κ})      Tautologie
IF (0∨...∨0∨[¬]xi) ∈ K THEN RETURN DP(K/ xi ← 1[0])  unit
IF K enthält Literal l, aber nicht seine Negation THEN
    RETURN DP(K \ {κ | l ∈ κ})                          pure literal
choose i
IF DP(K/ xi ← 1) = SAT THEN RETURN SAT                decide/
RETURN DP(K/ xi ← 0)                                backtrack
  
```



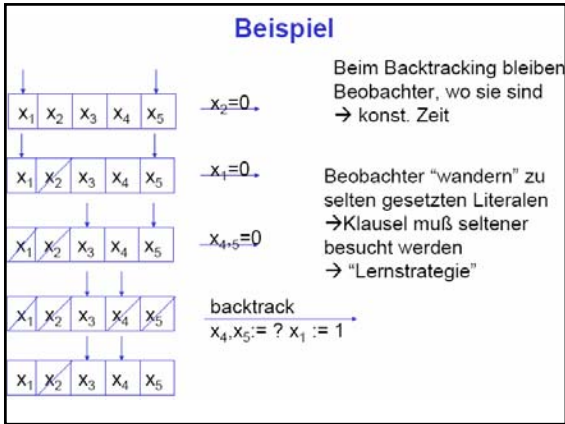
- Problem: SAT ist NP-vollständig!!
  - Es existieren einige Möglichkeiten, den Ablauf des Programms zu beschleunigen:
- Im folgenden vier „Tricks“ um die Laufzeit zu senken.

### 1. Trick: Schnelles Finden von Unit Klauseln

Unit-Klausel = Klausel mit n-1 Literalen auf 0, 1 Literal auf ?

Lösung (z.B. in *chaff*): Pro Klausel 2 Beobachter, die auf ?-Literale zeigen

Solange beide Beobachter auf ?-Literale zeigen, wird Klausel nicht angerührt



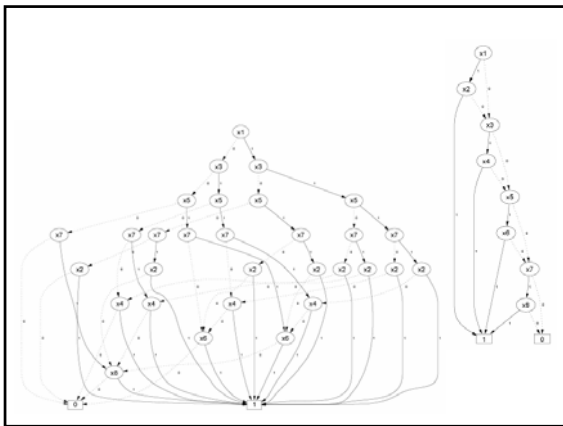
## 2. Trick: Konfliktanalyse

Konflikt = leere Klausel = Literal, für das sowohl 0 als auch 1 propagiert wird

Idee: "Grund" für den Widerspruch wird explizit als Klausel zur Klauselbasis hinzugefügt "Lernen"

- "denselben Fehler nicht noch mal machen"
- Suchraum einschränken

"Grund" wird durch Analyse des Implikationsgraphen generiert, der Ursache/Wirkung von Wertzuweisungen dokumentiert



## 3. Trick: Nichtchronologisches Backtracking, Zufällige Restarts

Auf Grund der Konfliktanalyse nicht die letzte, sondern eine frühere Entscheidung rückgängig machen

Von Zeit zu Zeit einfach von vorn anfangen, und (randomisiert) an anderen Variablen verzweigen

→ Wissen über bisherige Suche in Form der gelernten Klauseln verfügbar

→ Möglichkeit, komplizierten Teilen des Suchraums zu entfliehen

## 4. Trick: Heuristiken für Entscheidungen

*Grasp*: Setze die Variable, die in den meisten offenen Klauseln vorkommt

*Chaff*: Setze Variable, die häufig in gelernten Klauseln vorkommt

- weitere Algorithmen:

**Stålmarck-Algorithmus**: patentierter Algorithmus, um Tautologie/Erfüllbarkeit *beliebiger* Formeln zu entscheiden

- Google spuckt viele SAT-Solver (open source) aus, viel auf Universitätsseiten verbreitet

## Zusammenfassung SAT-Checker

arbeiten hervorragend auf Problemen "mit Struktur"

einige 10.000 – 100.000 Variablen

neben den vollständigen Checkern ex. noch viele unvollständige Verfahren

- Quellen:

Model Checking,  
Clarke, Grumberg & Peled  
Wikipedia.org  
[www2.informatik.hu-berlin.de/top/  
mitarbeiter/schmidt/lehre/cgv/  
cgv030530b.pdf](http://www2.informatik.hu-berlin.de/top/mitarbeiter/schmidt/lehre/cgv/cgv030530b.pdf)

- Fragen..?