# Automated Theorem Proving
## – *Foundations* –

**July 2015**


Wolfgang Küchlin

**Symbolic Computation Group**
**Wilhelm-Schickard-Institute of Informatics**
**Faculty of Mathematics and Sciences**

**Universität Tübingen**

**Steinbeis Technology Transfer Centre**
**Object and Internet Technologies (STZ OIT)**

**Wolfgang.Kuechlin@uni-tuebingen.de**
**http://www-sr.informatik.uni-tuebingen.de**


SR

# *Contents*

- Propositional Algebra
- Satisfiability, Validity, Entailment
- Application Example: Automotive Configuration
- Decision Methods Overview
- Preparation for Automated Theorem Proving
  - Disjunctive Normal Form DNF
  - Conjunctive Normal Form CNF
  - Efficient CNF Conversion Methods
  - Tseitin-Plaisted-Greenbaum Transform



# *Satisfiability and Validity*

- A formula F is *satisfiable* if it has a *model*. A model is a valuation $\beta$: Var(F) $\rightarrow$ {1, 0} s.th. $\beta$(F)=1. We write $\beta \models F$ or $\models_\beta F$. We write SAT(F)=true if F is satisfiable, else SAT(F)=false if F is unsatisfiable.
  - The interpretation of the logical connectives is fixed, so $\beta$ only depends on the valuation of the propositional variables. There are no function or predicate symbols (and no quantifiers).
- A formula F is *valid* ( $\models F$), if $\beta$(F)=1 for all $\beta$.
- F is valid iff. ¬F is unsatisfiable.
  - Since $\beta$(F)=1 iff. $\beta$(¬F)=0, and $\beta$(F)=0 iff. $\beta$(¬F)=1.



# *Semantic Entailment*

- Formula F entails formula G under a valuation β ($F \models_\beta G$) (resp. G follows from F), if β(G)=1 whenever β(F)=1.

  (β must completely assign all variables in F and G)
- If $F \models_\beta G$ for *all* complete valuations β, then G follows from F, and we write $F \models G$.
- Proposition: $F \models G$ iff $F \wedge \neg G \models \bot$
  - Let $F \models G$. Then β(¬G)=0 whenever β(F)=1.
  - Let $F \wedge \neg G \models \bot$. Hence if β(F)=1 then β(¬G)=0, and so β(G)=1.
- Corollary: $F \models G$ iff SAT(F∧¬G) = false (and $\models \neg F \vee G$)
  - There is no counterexample where β(F)=1 and β(G)=0
- → Semantic entailment can be proved by SAT-Solving



# *Propositional Algebra: Equivalences*

We write $F \equiv G$ if $\models (F \leftrightarrow G)$.

- $\beta(F) = \beta(G)$ for all valuations $\beta$.
- Attn: $\equiv$ is a meta-symbol, not a propositional connective!
- $\equiv$ denotes an equivalence relation

- Distributivity:

  $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$ $\qquad$ $F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$

- Absorption:

  $F \vee (F \wedge G) \equiv F \wedge (F \vee G) \equiv F$

- DeMorgan´s Laws:

  $\neg(F \wedge G) \equiv \neg F \vee \neg G$ $\qquad$ $\neg (F \vee G) \equiv \neg F \wedge \neg G$

- Commutativity and Associativity of $\wedge$, $\vee$.

- … and many more …



# *Decision Procedures and SAT-Solving*

- Bad news: SAT is NP-complete (Cook 1972)
  - No known algorithm decides SAT (true/false) in polynomial time
- Good News: SAT(F) is decidable!
  - Truth Tables → guaranteed exponential, toy problems only
  - Disjunctive normal form (DNF) → easily exp., small problems ok.
  - Tableaus → similar to DNF, easily exp., small problems ok.
  - Boolean Polynomials → little use.
  - Binary Decision Diagrams (ROBDD) → model checking use, 100s variables ok, O(1) SAT-solving, easy model counting, canonical form
  - Propositional Resolution → too many deductions, theoretical importance
  - Davis-Putnam-Logemann-Loveland (DPLL) → toy problems
  - DPLL based CDCL SAT-Solving → practically efficient for science and industry, 100,000+ variables, method of choice, very robust, much research.



# *Method Example: Truth Table and DNF*

$F = ((x \wedge \neg(y \rightarrow z)) \oplus x)$

| x | y | z | $y \rightarrow z$ | $\neg(y \rightarrow z)$ | $x \wedge \neg(y \rightarrow z)$ | $(x \wedge \neg(y \rightarrow z)) \oplus x$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 |

F is satisfiable, but not valid. Exponential number of rows.

DNF: enumerate all points =1

$DNF(F) = (x \wedge \neg y \wedge \neg z) \vee (x \wedge \neg y \wedge z) \vee (x \wedge y \wedge z) = (x \wedge \neg y) \vee (x \wedge y \wedge z)$



# Application Example: *Mercedes Configuration.*

Example: E-Class

- approx. 1.500 codes (options, countries, ..)
- approx. 3.000 rules in product description
- rule based BOM with approx. 35.000 parts


```
B(P09) = 297+540+543;

B(610) = (512 / 527 / 528)+608+ -978;

Z(P09) = (M271 + -M013 / M272) + 830 /
          Z04+ -(M273 + Z27)

Z(682) = 623 / 830 / 513L

B(450) = L+965+ 670+837+ -P34+
((M271/M651)+(953/955)+(100A/200A)+(334/335)+(301/336/337) /
 / M642+(2XXL/557L/571L)+(953/955)+(100A/200A)+(334/335)+ (301/337));
```

# Application Example: *Mercedes Configuration.*

Example: E-Class

- approx. 1.500 codes (options, countries, ..)
- approx. 3.000 rules in product description
- rule based BOM with approx. 35.000 parts

```
B(P09) = 297+540+543;

B(610) = (512 / 527 / 528)+608+ -978;

Z(P09) = (M271 + -M013 / M272) + 830 /
         Z04+ -(M273 + Z27)

Z(682) = 623 / 830 / 513L

B(450) = L+965+ 670+837+ -P34+
((M271/M651)+(953/955)+(100A/200A)+(334/3
 / M642+(2XXL/557L/571L)+(953/955)+(100A/2
```

**P09**: Sonnenschutzpaket
297: Rollos Fond-Türen links+rechts
540: Rollo elektr. Heckfenster
543: Sonnenbl. li+re m. Make-up-Sp.
**610**: Nachtsichtsystem
512: Comand APS mit DVD-Wechsl.
527: Comand DVD APS mit NAVI
528: Comand DVD+ ohne NAVI
608: Autom. Fernlichtschaltung
978: Spezial-Fahrgestell
**682**: Feuerlöscher
830: Zusatzteile China
623: Golf-Staaten-Ausführung
513L: Belgien
**M271**: R4-Ottomotor
M272: V6-Ottomotor
M273: V8-Ottomotor
M013: Motor leistungsreduziert
Z04: B4 Panzerung
Z27: Bodenpanzerung
**450**: TAXI-International

# Application Example: *Checking Configuration Options*

- Car order is a list L of options (option codes)
  - Codes in L are *true*, all others are *false* (for this car)
- Compile all configuration rules into a single formula **C**
  - Solutions $\beta(\mathbf{C})$=true are constructible orders
  - restriction $\mathbf{C}|_{o=true}$ sets o=true in **C**: all cars with option o
- Checking the configuration rule-set
  - Is option o available? → SAT($\mathbf{C}|_{o=true}$) = true?
  - Is option o forced? → SAT($\mathbf{C}|_{o=false}$) = false?
  - Is option o available in country c? → SAT($\mathbf{C}|_{c=true,o=true}$) = true?
  - Is option o forced in country c? → SAT($\mathbf{C}|_{c=true,o=false}$) = false?
- Configuration step
  - Is option o available after selecting options $o_1,\ldots,o_n$?
    → SAT($\mathbf{C}|_{o1=true,\ldots,on=true,\ o=true}$) = true?



# *Conjunctive Normal Form (CNF)*

- Resolution and DPLL / CDCL SAT-Solving require CNF
- CNF is a conjunction ($\wedge$) of *clauses* ($\ell_1 \vee \ldots \vee \ell_n$)
  - The $\ell_i$ are *literals* (positive or negative variables)
  - Example: $F = (x \vee \neg y) \wedge (x \vee \neg z) \wedge (z \vee \neg y) \wedge (z \vee \neg x)$
  - Simplified set notation: $F = \{\{x, \neg y\}, \{x, \neg z\}, \{z, \neg y\}, \{z, \neg x\}\}$
  - CNF lists a set of (simple) *constraints*, which must be **simultaneously** satisfied for F=1.
  - CNF enumerates conditions for $F \neq 0$.
  - CNF enumerates the negations of F´s roots.
- CNF conversion (theoretically)
  - Push negations ¬ inside → negation normal form NNF
  - Apply distributivity law: $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$

# *Example: CNF from function table*

$F = ((x \wedge \neg(y \rightarrow z)) \oplus x)$

| x | y | z | $y \rightarrow z$ | $\neg(y \rightarrow z)$ | $x \wedge \neg(y \rightarrow z)$ | $(x \wedge \neg(y \rightarrow z)) \oplus x$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 |

CNF: negate all roots: $\neg(\neg x) \wedge \neg(x \wedge y \wedge \neg z) \equiv x \wedge (\neg x \vee \neg y \vee z)$
$\equiv x \wedge (\neg y \vee z)$



# *Efficient CNF Conversion*

## ➢ Bottom-up CNF construction with simplification

- Definitions:
  - Clauses $X = \{x_1,\ldots,x_m\}$, $Y = \{y_1,\ldots,y_n\}$
  - $\{X, Y\}$ represents $X \wedge Y$ with clauses X and Y
  - $X \cup Y := \{x_1,\ldots,x_m,y_1,\ldots,y_n\}$
- $CNF(x) := \{\{x\}\}$
- $CNF(\wedge,\{X_1,\ldots,X_m\},\{Y_1,\ldots,Y_n\}) = \{X_1,\ldots,X_m\} \cup \{Y_1,\ldots,Y_n\}$
- $CNF(\vee,\{X_1,\ldots,X_m\},\{Y_1,\ldots,Y_n\}) = (X_1 \wedge \ldots \wedge X_m) \vee (Y_1 \wedge \ldots \wedge Y_n) =$
  $\{X_1 \cup Y_1,\ldots,X_m \cup Y_1,\ X_1 \cup Y_2,\ldots,X_m \cup Y_2,\ldots,\ X_1 \cup Y_n,\ldots,X_m \cup Y_n\}$

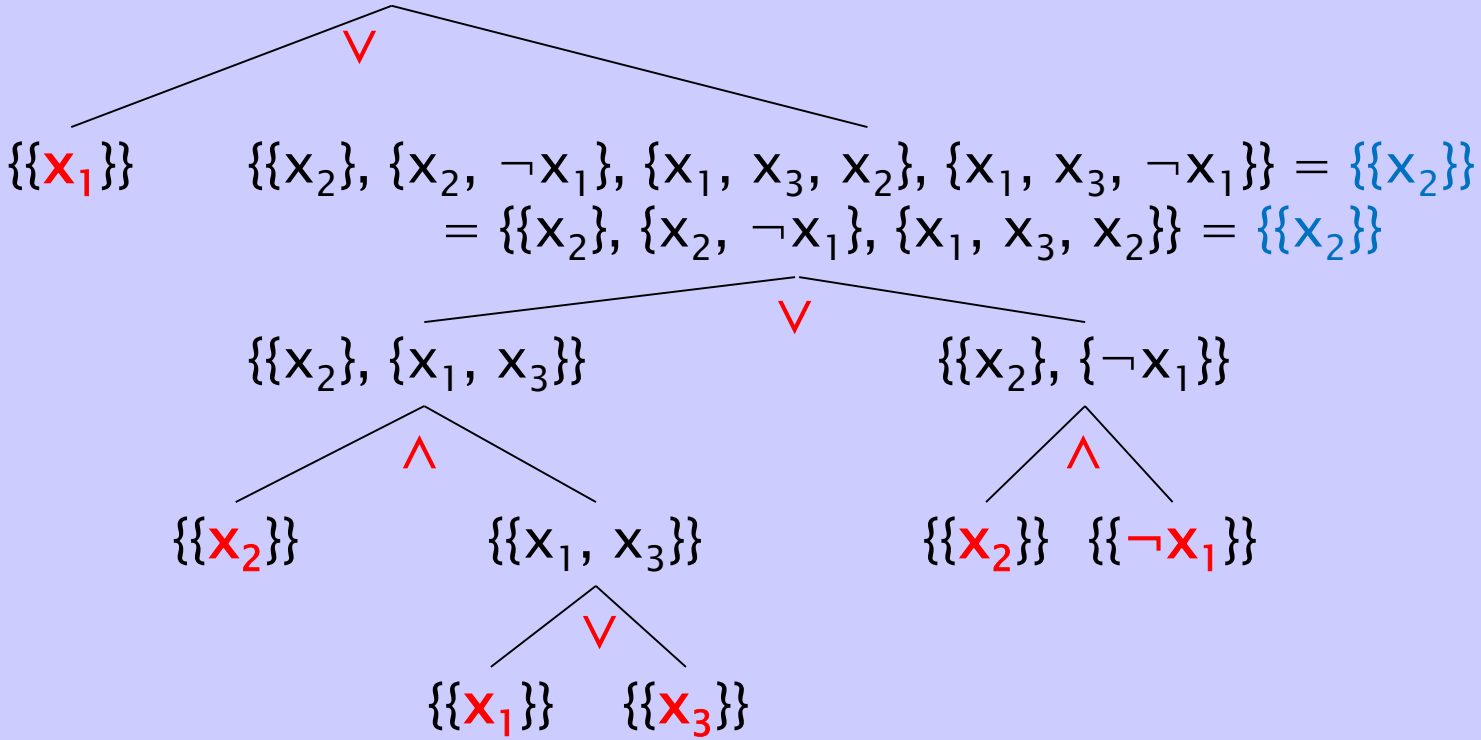
## ➢ Essential advantage: simplify early

- Start with small trivial formulae, simplify as CNF grows
- $\{\{x\},\{x,y\}\} = \{\{x\}\}$, and $\{\{x, \neg x\}\} = T$

# *Example: Bottom-up CNF Conversion*

Beispiel: $F = x_1 \vee (x_2 \wedge (x_1 \vee x_3) \vee (x_2 \wedge \neg x_1))$

$\{\{x_1, x_2\}, \{x_1, x_2, \neg x_1\}, \{x_1, x_3, x_2\}\} =$
$\{\{x_1, x_2\}, \{x_1, x_3, x_2\}\} = \{\{x_1, x_2\}\}$



# *Complexity of CNF-Transformation*

- Application of Distributivity Law may double the formula size
  - $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$
- Worst case: CNF(F) exponentially larger than F
  - Very serious impediment for applications
- Example: $(x_{11} \wedge x_{12}) \vee \ldots \vee (x_{n1} \wedge x_{n2})$.
  - CNF contains $2^n$ clauses with n variables each (→ Induction)
- Solution in 2 steps
  1. Tseitin transformation
  2. Plaisted-Greenbaum Transformation
- Transformations maintain satisfiability, not equivalence!



# CNF: Tseitin-Transformation

- Key idea:
  - introduce auxiliary variables to represent (sub-)formulae
  - introduce additional constraints to link variables to formulae
- General concept, we use it only for CNF
  - In $F \vee (G \wedge H)$, represent $(G \wedge H)$ by new x
  - Add constraint $x \leftrightarrow (G \wedge H)$.
  - Hence $F \vee (G \wedge H)$ is replaced by $(F \vee x) \wedge (x \leftrightarrow (G \wedge H))$
  - Now convert $x \leftrightarrow (G \wedge H)$ to $x \rightarrow (G \wedge H)$ and $(G \wedge H) \rightarrow x$, giving clauses $(\neg x \vee G)$, $(\neg x \vee H)$, $(\neg G \vee \neg H \vee x)$ in addition to $(F \vee x)$.
  - This is the „full“ Tseitin-Transformation
  - Only satisfiability is preserved: $F \vee (G \wedge H) \cong (F \vee x) \wedge (x \leftrightarrow (G \wedge H))$
  - Model count is preserved
- Bad news: Now G and H are doubled instead of F!



# *CNF: Tseitin-Plaisted-Greenbaum Transformation*

- Problem: full Tseitin-Transform doubles G and H in $F \vee (G \wedge H) \cong (F \vee x) \wedge (x \rightarrow (G \wedge H)) \wedge ((G \wedge H) \rightarrow x)$
- Plaisted & Greenbaum Transform: $(F \vee x) \wedge (x \rightarrow (G \wedge H))$
  - David A. Plaisted, Steven Greenbaum: A Structure Preserving Clause Form Transform. *J. Smbolic Computation* Vol 2(3), 1986.
  - Often also misnamed „Tseitin-Transform“
  - Drops constraint $((G \wedge H) \rightarrow x)$
  - G and H no longer doubled, no exponential blow-up!
  - Model count may (but need not) double with every aux. Variable
- Theorem [Plaisted & Greenbaum]

  $F \vee (G \wedge H) \cong (F \vee x) \wedge (x \rightarrow (G \wedge H))$
- Absolutely crucial in scientific/industrial applications