Solving Boolean Equations with BDDs and Clause Forms

Gert Smolka

Abstract

- Methods for solving Boolean equations
  - BDDs [Bryant 1986]
  - Clause forms [Quine 1959]
- Efficient data structure and algorithms for large finite sets (e.g. 2^{1000})

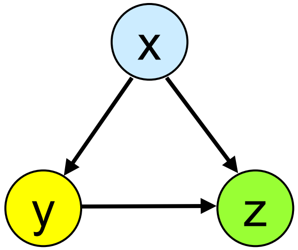
Applications

- Verification (e.g. model checking)
- CAD of HW (e.g. circuit minimization)
- Knowledge representation (e.g. truth maintainance)

Why do I talk about it?

- Beautiful and important
- Interesting trip from logic to algorithms
- Equation solving not covered in textbook accounts of propositional logic
- Had to work it out for our introductory course on Computational Logic

Modelling with Boolean Equations: Graph Coloring



Colorings of the graph are the solutions of the equations

Is graph bipartite?

$x \neq y, \;\; x \neq z, \;\; y \neq z$

$\neg(x \leftrightarrow y) = 1, \ldots$

Is graph 4-partite?

$(x_1, x_2) \neq (y_1, y_2), \;\; \ldots$

$\neg(x_1 \leftrightarrow x_2) \vee \neg(y_1 \leftrightarrow y_2) = 1, \ldots$

Modelling with Boolean Equations: Secrets of a Long Live

1) If I don't drink beer, I always eat fish
2) If I have both beer and fish, I don't have ice cream
3) If I have ice cream or do not drink beer, I don't have fish

$$\neg B \rightarrow F = 1$$
$$B \wedge F \rightarrow \neg I = 1$$
$$I \vee \neg B \rightarrow \neg F = 1$$

$$B = 1$$
$$\neg F \vee \neg I = 1$$

solved form

Formalities

$Bool = \{0,1\}$

$x,y,z \in Var$

$s \in State = Var \to Bool$

$f,g \in BF = State \to Bool$

$BF \cong P(State)$ $\{s \in State \mid fs=1\}$

$a,b,c \in Exp$

$Den \in Exp \to BF$

Boolean Operations

$Bool^n \to Bool$

$x \wedge y = \min \{x,y\}$

$x \vee y = \max \{x,y\}$

$\neg x = 1-x$

$x \to y =$ if $x \leq y$ then 1 else 0

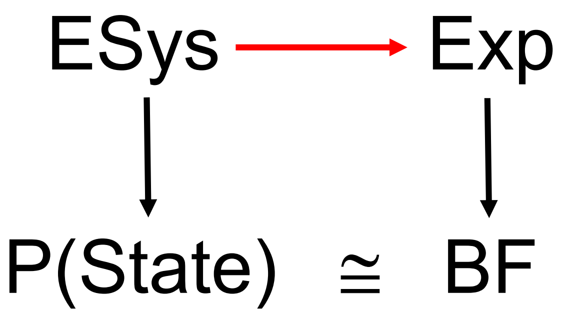
$x \leftrightarrow y =$ if $x=y$ then 1 else 0

Solving Equation Systems

$$
\begin{array}{ccc}
\text{ESys} & & \text{Exp} \\
\downarrow & & \downarrow \\
\text{P(State)} & \cong & \text{BF}
\end{array}
$$

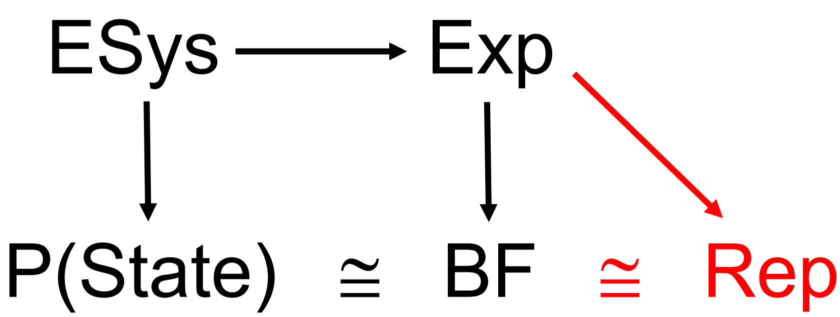
Solutions of equation system can be described by Boolean function

Solving Equation Systems (2)



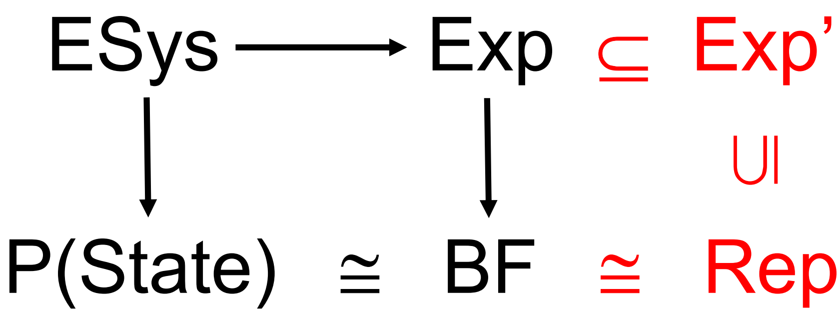
Phase 1: equation system → expression

Solving Equation Systems (3)



Phase 2: expression → good rep of BF

Solving Equation Systems (4)



Extend expressions to contain good reps of BFs

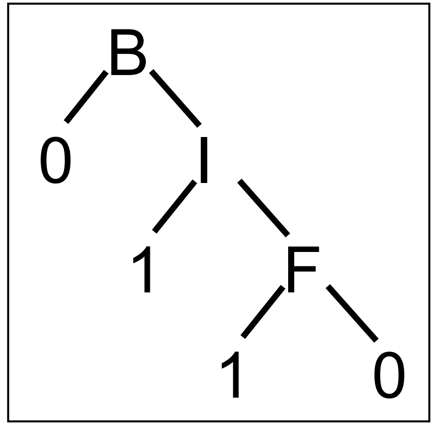
Equation System $\rightarrow$ Expression

$a=b$	$\Leftrightarrow$	$a \leftrightarrow b=1$
$a \neq b$	$\Leftrightarrow$	$\neg a \leftrightarrow b=1$
$a \leq b$	$\Leftrightarrow$	$a \rightarrow b=1$
$a<b$	$\Leftrightarrow$	$\neg a \wedge b=1$
$a=1$ and $b=1$	$\Leftrightarrow$	$a \wedge b=1$
$a=1$ or $b=1$	$\Leftrightarrow$	$a \vee b=1$

Example

$$\neg B \rightarrow F = 1$$
$$B \wedge F \rightarrow \neg I = 1$$
$$I \vee \neg B \rightarrow \neg F = 1$$

Equation system



Prime tree

$$(\neg B \rightarrow F) \wedge (B \wedge F \rightarrow \neg I) \wedge (I \vee \neg B \rightarrow \neg F) = 1$$

Normal equation

$$B \wedge (\neg F \vee \neg I)$$

Conjunctive prime form

$$(B \wedge \neg I) \vee (B \wedge \neg F)$$

Disjunctive prime form

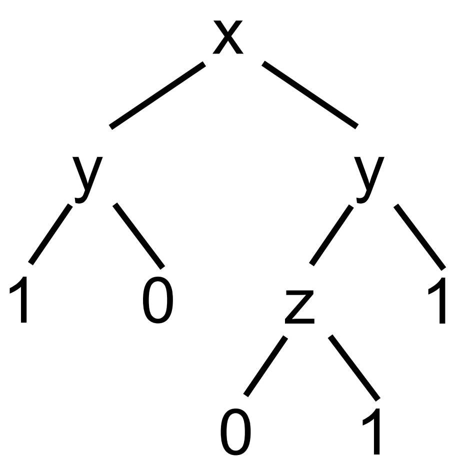
Overview

- Intro
- BDDs [Bryant 1986]
- Clause forms

BDDs

- Decision trees
- Prime trees
- Algorithms
- Minimal Graph Representation

Decision Trees



Graphical Representation of Nested Conditionals

```
if x=0
then if y=0
        then 1
        else 0
else if y=0
        then if z=0
                then 0
                else 1
        else 1
```

Conditonal as new Operation

$Bool^3 \to Bool$

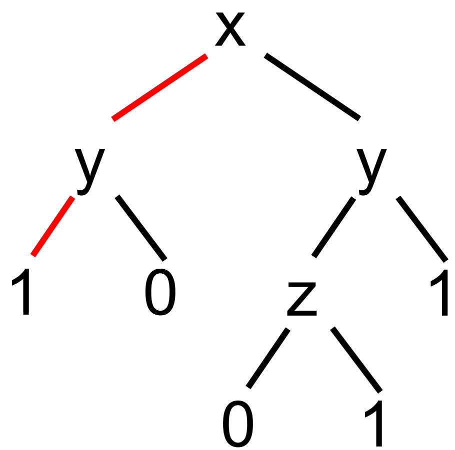
$(x,y,z) = \text{if } x=0 \text{ then } y \text{ else } z$

$= (\neg x \to y) \wedge (x \to z)$

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

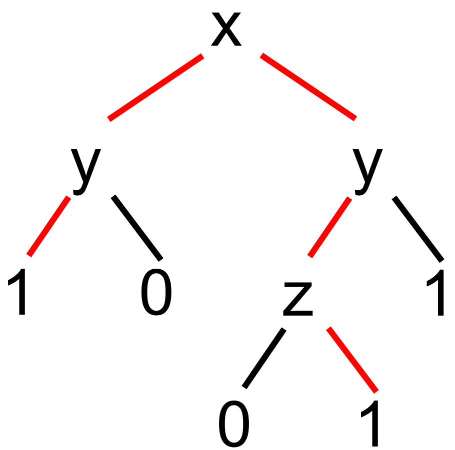
[Löwenheim 1910]

Decision Tree → DNF



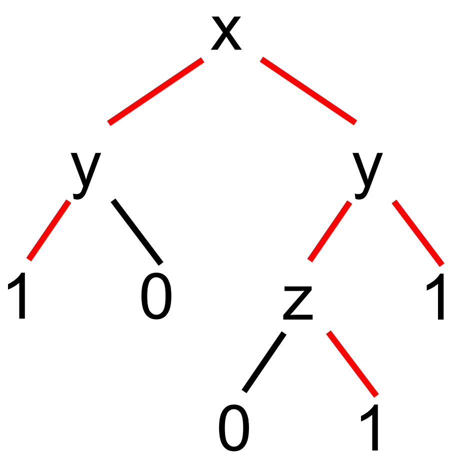
$(\neg x \wedge \neg y) \vee \ldots$

Decision Tree → DNF



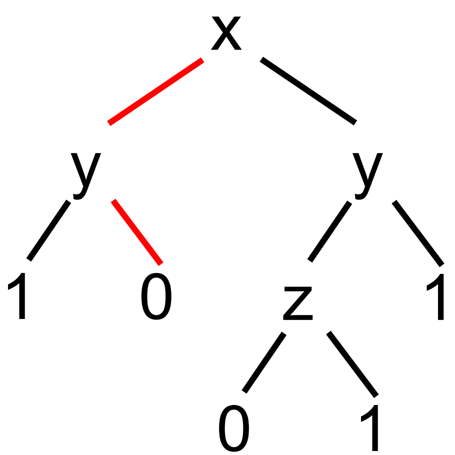
$(\neg x \wedge \neg y) \vee (x \wedge \neg y \wedge z) \vee \ldots$

Decision Tree → DNF



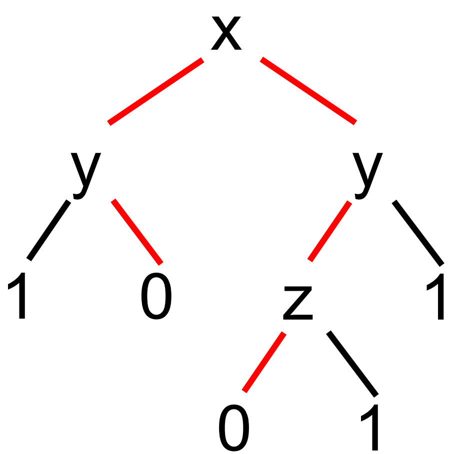
$$(\neg x \wedge \neg y) \vee (x \wedge \neg y \wedge z) \vee (x \wedge y)$$

Decision Tree → CNF



$(x \vee \neg y) \wedge \ldots$

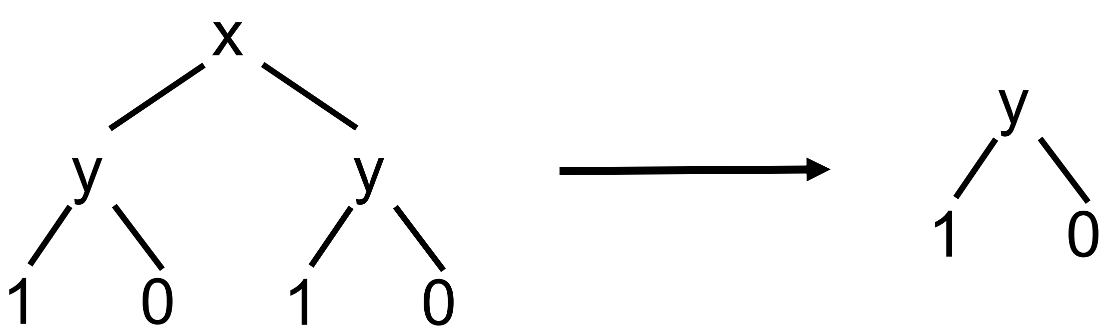
Decision Tree $\rightarrow$ CNF



$$(x \vee \neg y) \wedge (\neg x \vee y \vee z)$$

Reduction of Decision Trees

Based on (x,y,y) = y

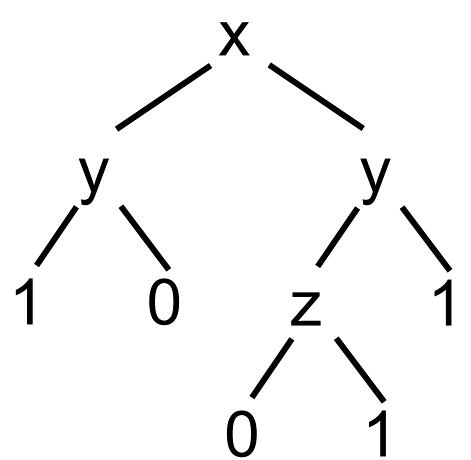


Ordered Decision Trees

- Fix linear order on variables

  $x < y < z < \ldots$

- Deeper variables must be larger



Prime Trees

- Ordered and reduced decision trees
- Isomorphic to Boolean functions
- Perfect representation of Boolean functions

$$\begin{array}{ccc} \text{Exp} & \subseteq & \text{Exp'} \\ \downarrow & & \cup\!| \\ \text{BF} & \cong & \text{PT} \end{array}$$

Theorem Different prime trees denote different Boolean functions.

Proof By induction on max of sizes. Case analysis:

1. a and b are both atomic.
2. Root variables of a and b are identical.
3. Root variable of a does not occur in b.

Theorem Every expression can be translated into equivalent prime tree.

Expansion Theorem

(Boole 1854, Löwenheim 1910, Shannon 1938)

$$a \equiv (x, a[x:=0], a[x:=1])$$

Operations on Prime Trees

not: $PT \to PT$
not a = $\pi(\neg a)$

and: $PT \times PT \to PT$
and(a,b) = $\pi(a \wedge b)$

Will see efficient algorithms

Constructors for PTs (ADT)

0: PT

1: PT

cond: $Var \times PT \times PT \rightarrow PT$

$cond(x,a,b) = \pi(x,a,b)$ provided $x < Va \cup Vb$

If a,b prime trees and x variable:

$\pi(x,a,a) = a$

$\pi(x,a,b) = (x,a,b)$ if $x < Va \cup Vb$

All algorithms will be based on these constructors

Algorithm for not

- Based on

  $\neg 0 = 1$

  $\neg 1 = 0$

  $\neg(x,y,z) = (x,\neg y,\neg z)$

- Orderedness preserved since no new variables
- Reducedness preserved since not injective

Algorithm for and

- Based on

  $(x,a,b) \wedge 0 = 0$

  $(x,a,b) \wedge 1 = (x,a,b)$

  $(x,a,b) \wedge (x,a',b') = (x, a \wedge a', b \wedge b')$

  $(x,a,b) \wedge c = (x, a \wedge c, b \wedge c)$ (only used if $x < Vc$)

- Orderedness preserved since no new variables
- Reducedness preserved by cond

Expression $\rightarrow$ Prime Tree

trans: Exp $\rightarrow$ PT

trans 0 = 0

trans 1 = 1

trans x = cond(x,0,1)

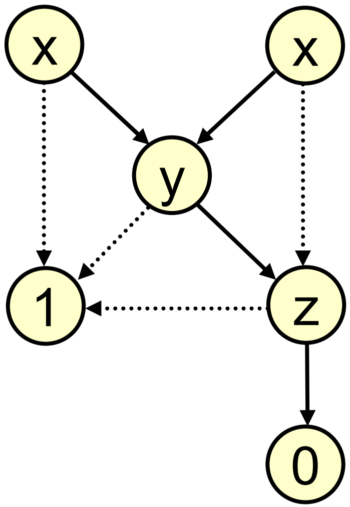
trans ($\neg$a) = not(trans a)

trans (a$\wedge$b) = and(trans a, trans b)

As is, and is exponential

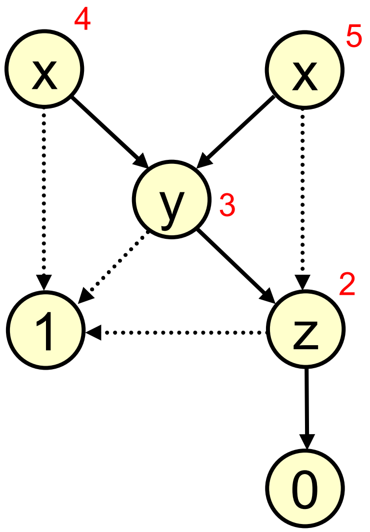
- Can make it quadratic by
  - dynamic programming (hashing over PTs)
  - constant time equality test for PTs

Minimal Graph Representation



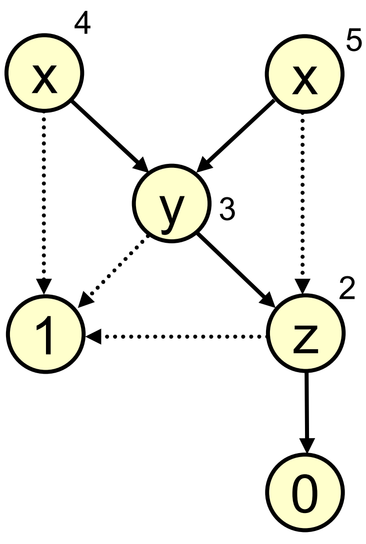
- Every node describes a prime tree
- Graph describes a subtree-closed set of prime trees
- Graph minimal iff different nodes describe different trees

Graph → Table



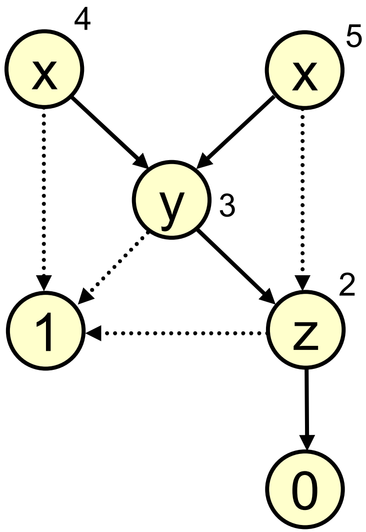
Number nodes of graph

Graph → Table



2	(z,1,0)
3	(y,1,2)
4	(x,1,3)
5	(x,2,3)

Graph → Table → Function



i	tab(i)
2	(z,1,0)
3	(y,1,2)
4	(x,1,3)
5	(x,2,3)

Graph minimal iff tab injective

Constant Time Realization of cond

```
cond(x,n,n') =
   if n=n' then n
   else if (x,n,n') ∈ Dom(tab⁻¹)
         then tab⁻¹ (x,n,n')
         else let n'' = least number not in Dom tab
              in tab := tab[n'':=(x,n,n')] ;
                 n''
```

Implement tab^{-1} with hashing

Overview

- Intro
- BDDs
- Clause forms [Quine 1959]

Conjunctive Normal Forms

literal	x, ¬x
clause C	finite set of literals, not x and ¬x
clause set S	finite set of clauses
cnf S	new expression form

$$(\text{cnf } S)s \;=\; \bigwedge_{C \in S} \; \bigvee_{a \in C} \; as \qquad\qquad (\wedge\varnothing = 1,\ \vee\varnothing = 0)$$

Conjunctive Prime Forms

- C implicate of a $\Leftrightarrow$ $a \leq \vee C$
- C prime implicate of a $\Leftrightarrow$ C minimal implicate of a
- Formula has only finitely many prime implicates
- $a \equiv cnf\ \{C \mid C \text{ prime implicate of } a\}$

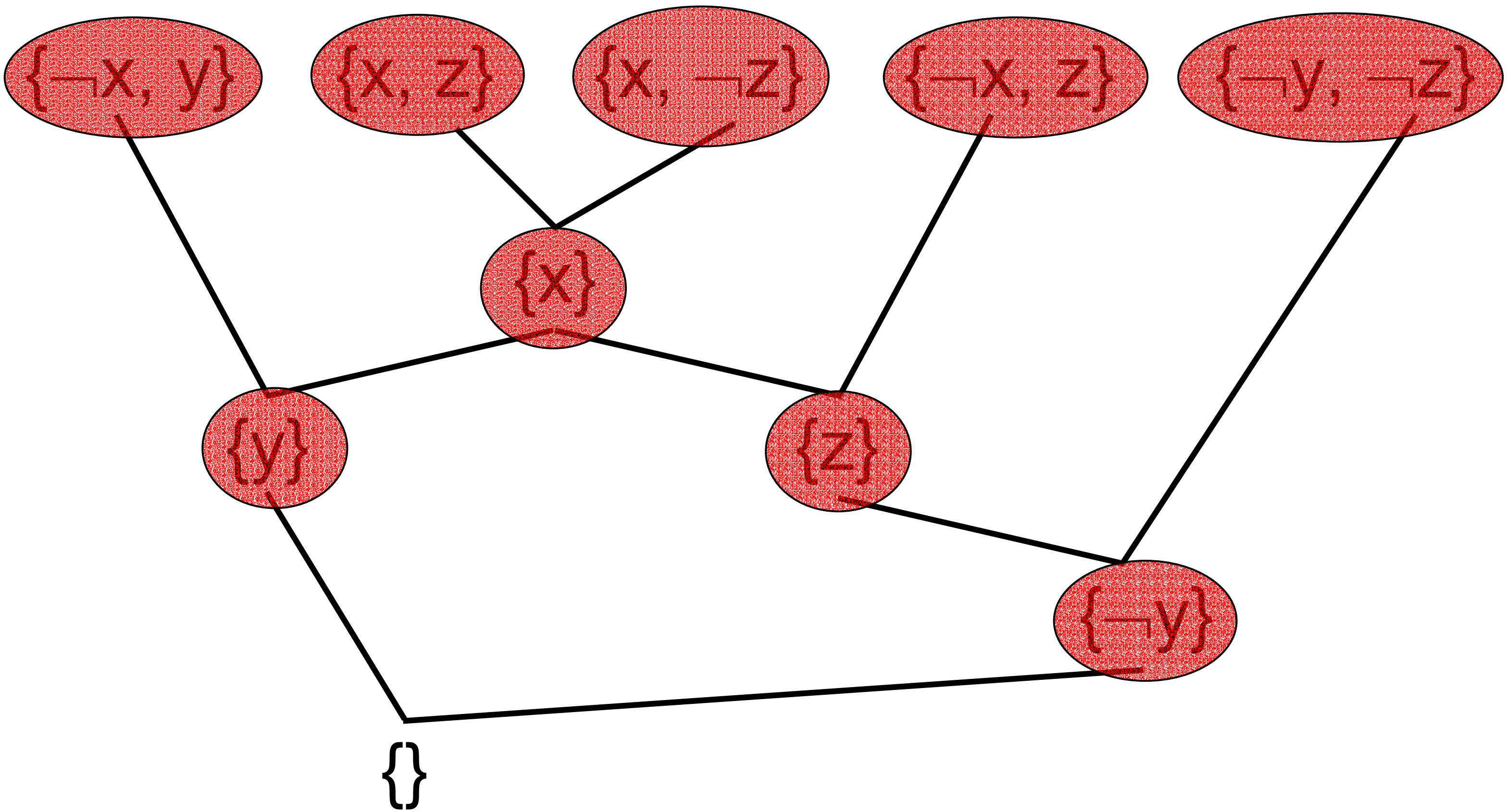
$$\begin{array}{ccc} \text{Exp} & \subseteq & \text{Exp'} \\ \downarrow & & \cup\!| \\ \text{BF} & \cong & \text{CPF} \end{array}$$

CNF $\rightarrow$ CPF

- CPF can be computed from CNF by 2 rules:
  - delete subsumed clause
  - add resolvent that is not subsumed

    $(a \vee b) \wedge (\neg a \vee c) \leq (b \vee c)$
- Equivalence transformations
- Terminate with CPF

Example : CNF → CPF



CNF $\rightarrow$ CPF

- Nice for few variables
- Explosive in number of variables
- By duality: DNF $\rightarrow$ DPF
- Application: truth maintainance in AI (CPF)
  - Reiter and de Kleer 1987
- Application: circuit minimization (DPF)
  - Quine 1959
  - Minimal size DNFs are subsets of DPF

Summary and Remarks

- 2 Methods for solving Boolean equations
  - BDDs [Bryant 1986]
  - clause forms [Quine 1959]
- Generalizes to Boolean algebras
- Generalizes to infinitely many variables
- There are other methods, e.g.
  - Complete normal forms [Boole 1854]
  - [Löwenheim 1910]

References

- Willard V. Quine.
  On Cores and Prime Implicants of Truth Functions.
  American Mathematical Monthly, 1959.
- Randal E. Bryant.
  Graph-based Algorithms for Boolean Function Manipulation. IEEE Transactions on Computers, 1986.
- Gert Smolka.
  Skript zur Vorlesung Einführung in die Computationale Logik, 2003. www.ps.uni-sb.de/courses/cl-ss03