

**Infineon**

# Foundations of Hierarchical SAT-Solving

*6<sup>th</sup> International Workshop on “Boolean Problems” 23-24.8.2004, Freiberg*

Presented by Yakov Novikov

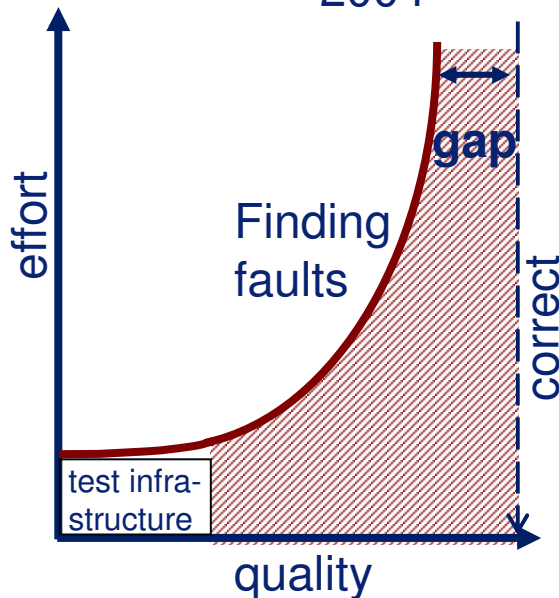
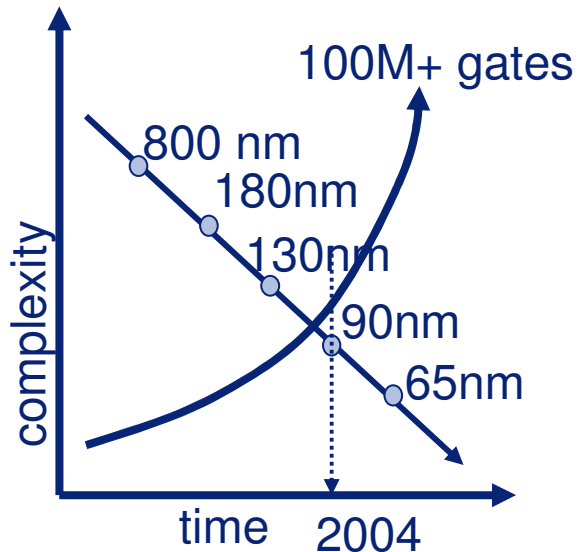
**Yakov Novikov and Raik Brinkmann**

Corporate Logic, Formal Verification



Never stop thinking

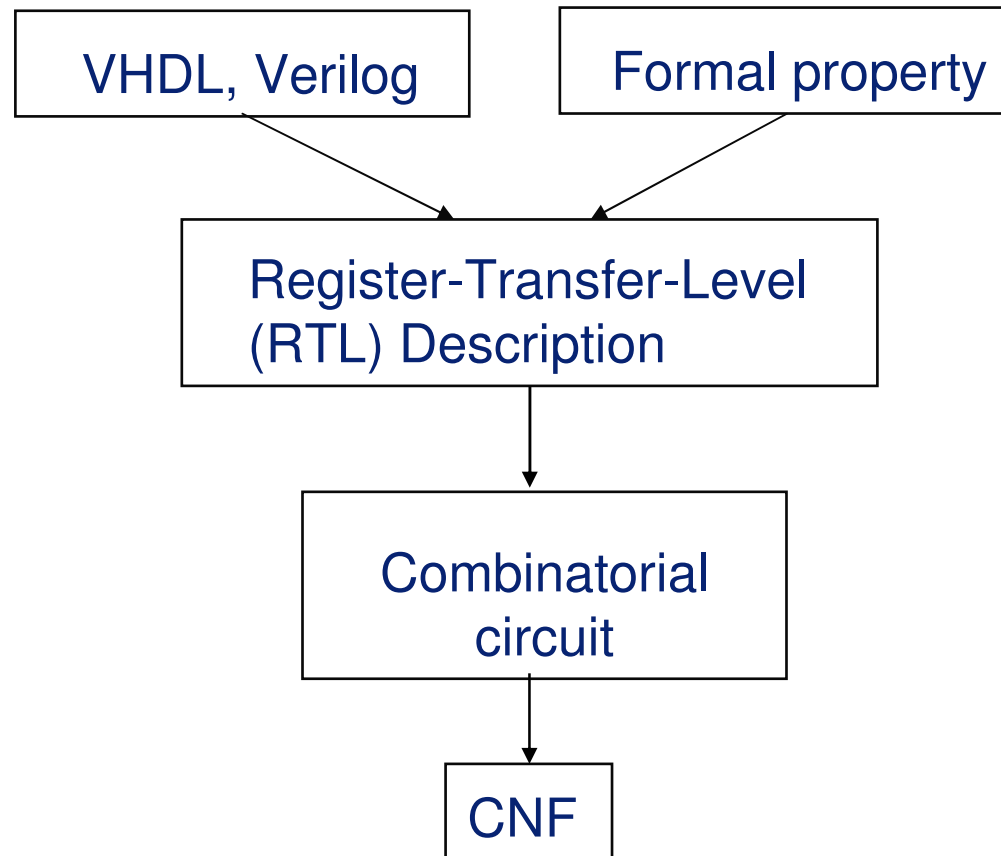
# Introduction: The Importance of Property Checking for Chip Makers



- Moore's Law continues.
- Soon chips will comprise 100M+ gates.
- Poor adaptivity in late design stages requires extreme quality.
- On average every chip requires 3+ redesigns (a few million € each!).
- Already more than 60% of project budget is spent on verification.
- Due to inherent limitations of simulation this percentage will grow but gap still opens.
- Formally proving properties becomes mandatory to close systematic gap!

## Introduction: Property checking flow

---



## Introduction: How to join the research?

---



The Boolean Satisfiability (SAT) problem is a central problem of discrete mathematics.



Currently SAT-solving is a hot topic of research.



The best SAT-solvers are commercially valid.



How can one join this research?



Know-how hidden in best SAT-solvers makes it difficult to develop sound results.



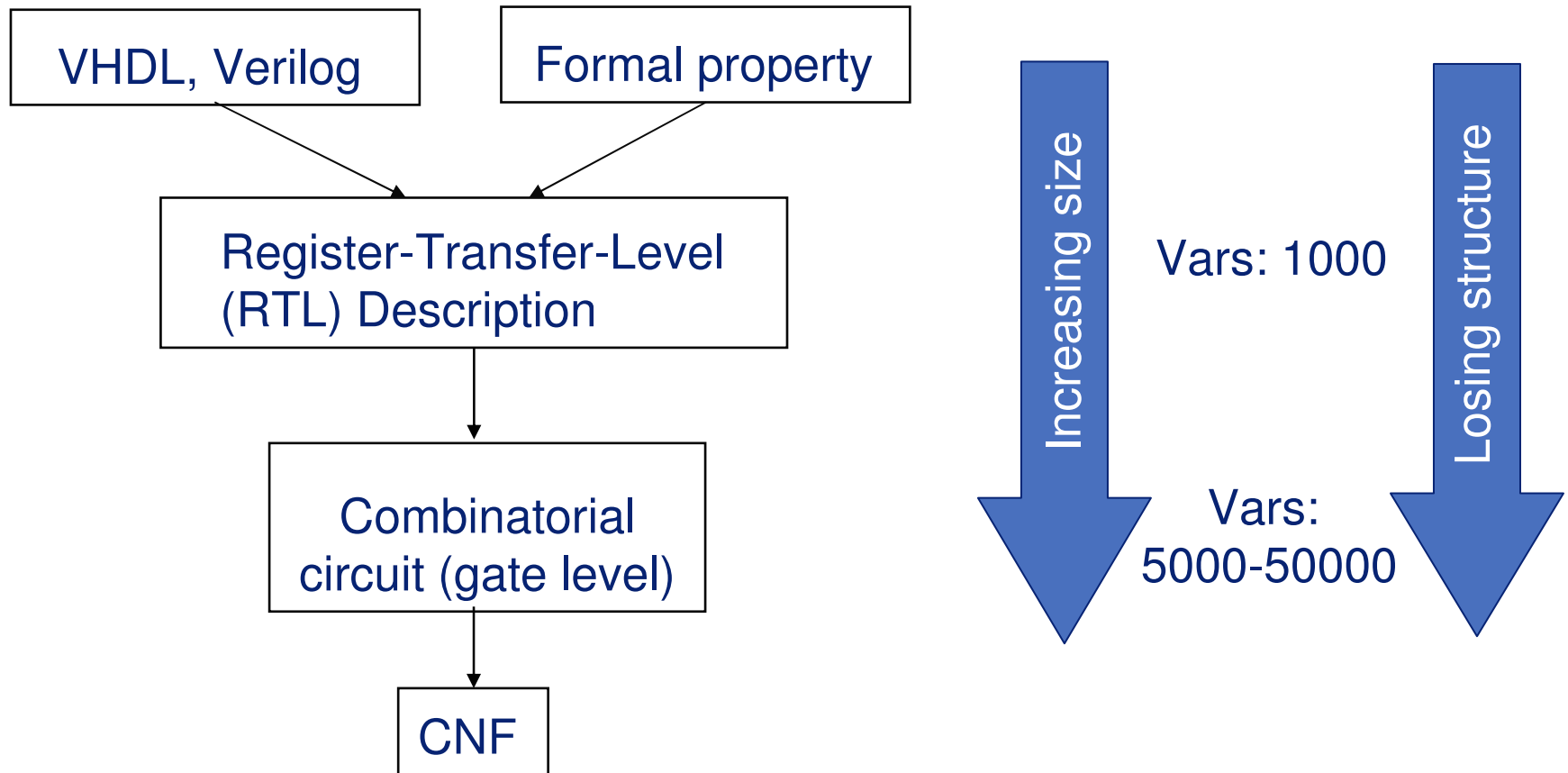
We propose a new theory opening a clear and easy way to be involved into practical SAT-solving.



Researchers are free to use any mathematics (including but not limited to traditional CNFs and BDDs)!

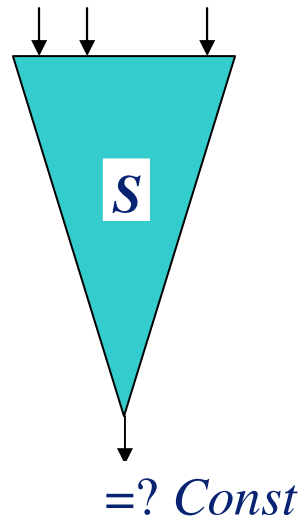
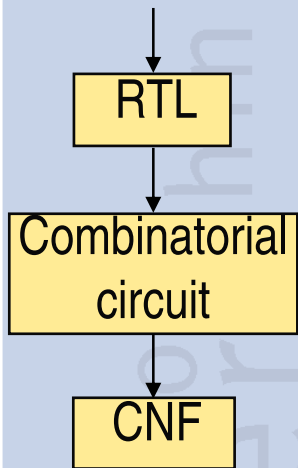
## Introduction: Drawbacks of top-to-down translation

---



## Introduction: Structural satisfiability problem

---

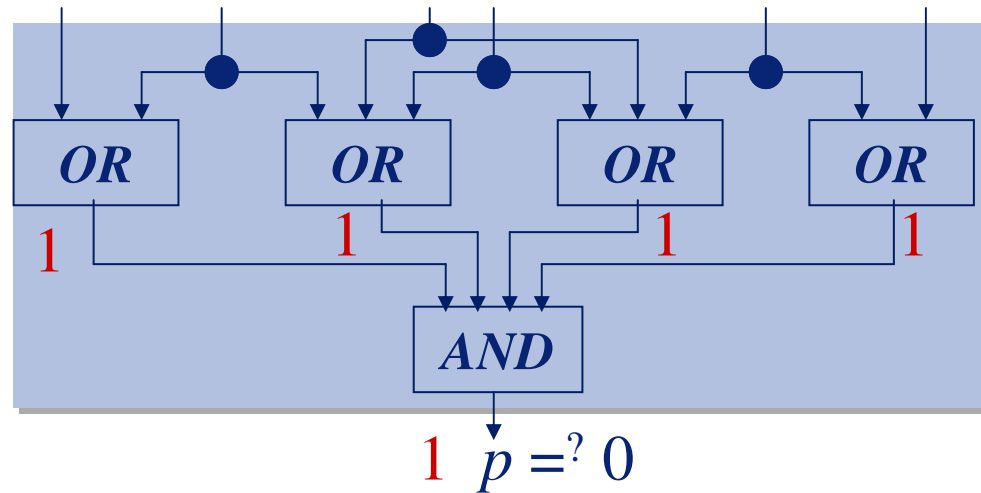
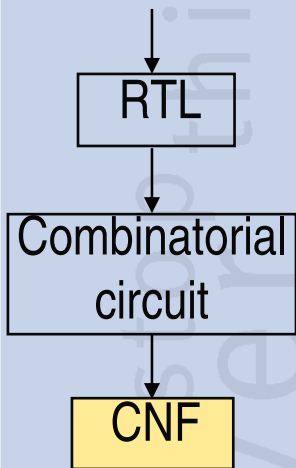


Starting at RT-level property-checking tasks can be reduced to providing answer to the question:

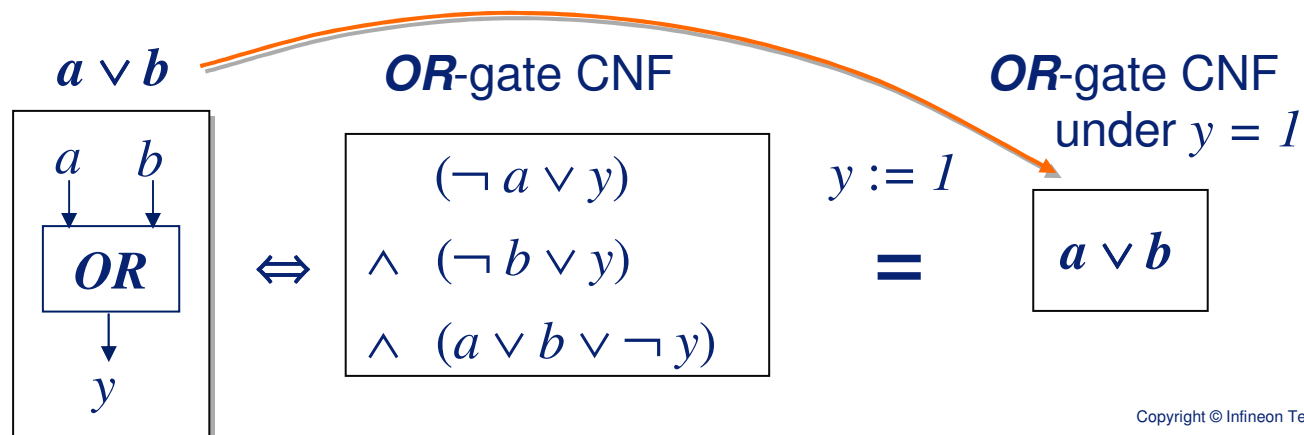
Given a combinatorial circuit **S**, does **S** implement a given constant function?

Our terminology :      combinatorial circuit = system  
                                  gate    = block

# Introduction: Checking satisfiability of CNF as structural problem

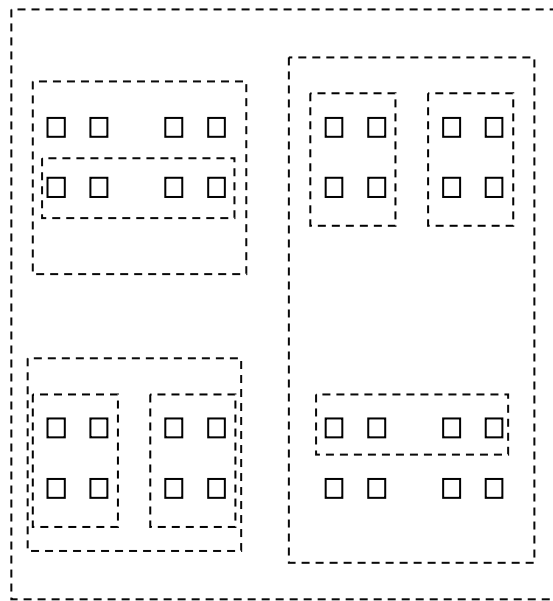
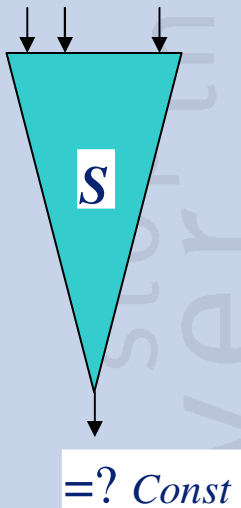


Propagation of constraint  $p = 1$  into system:

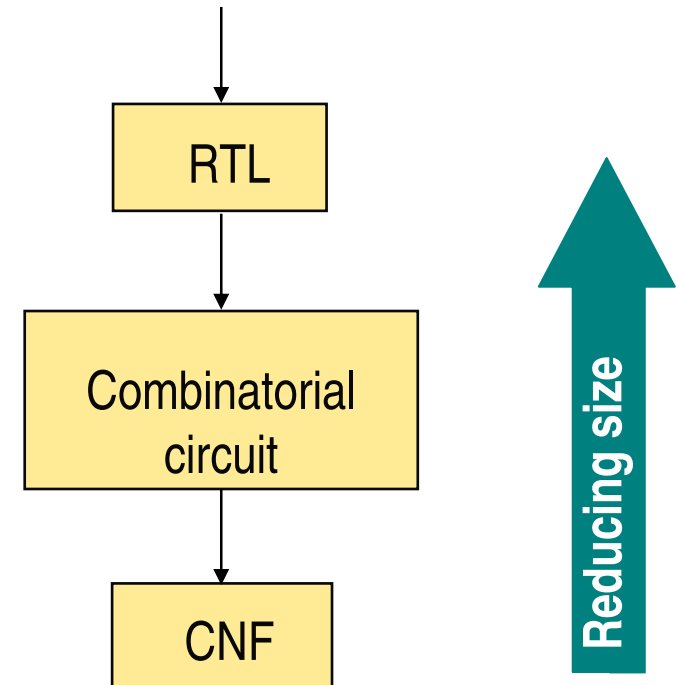
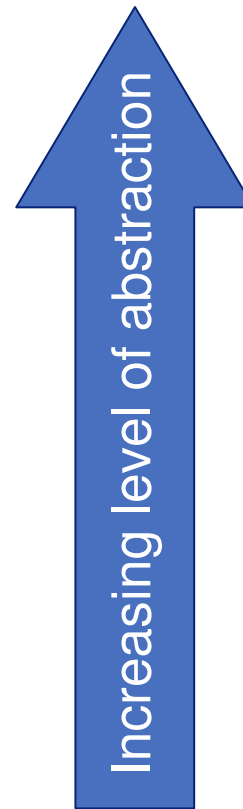


# Introduction: Block enlargement

We need a theory for solving structural SAT- instances when blocks implement complex functions.

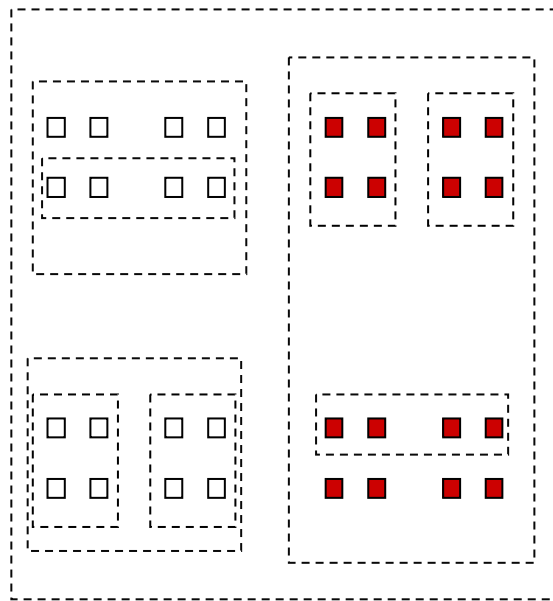
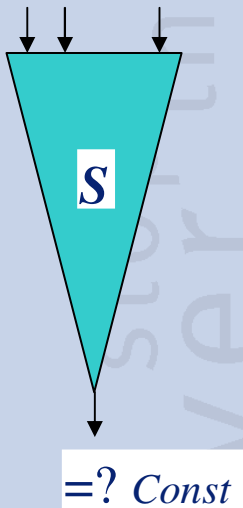


**Block enlargement**

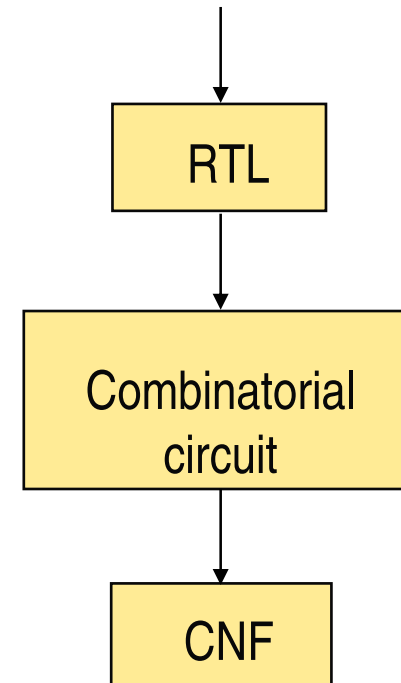
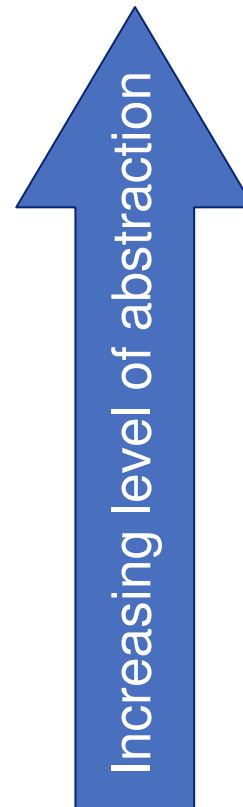


# Introduction: Block enlargement

We need a theory for solving structural SAT- instances when blocks implement complex functions.

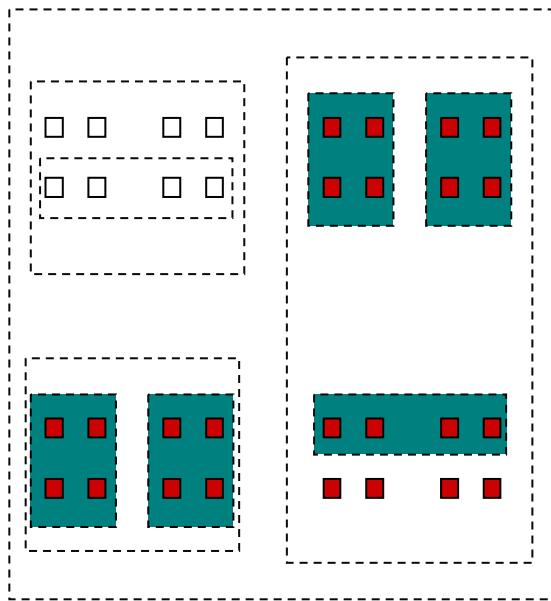
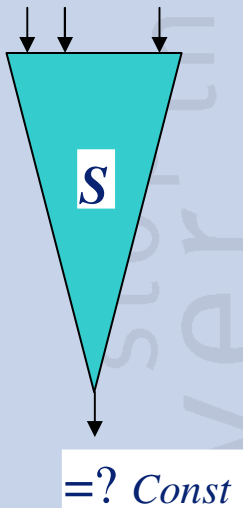


**Block enlargement**

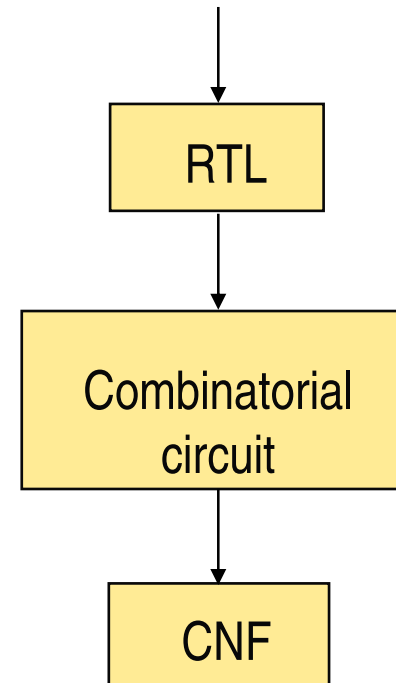
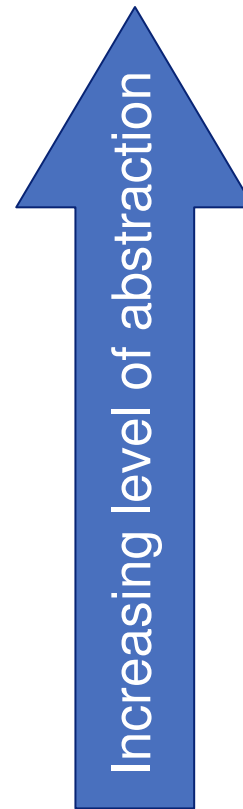


# Introduction: Block enlargement

We need a theory for solving structural SAT- instances when blocks implement complex functions.

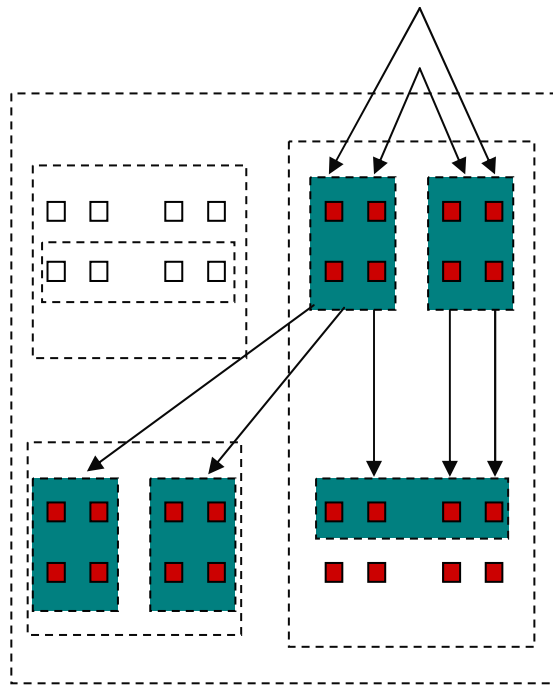
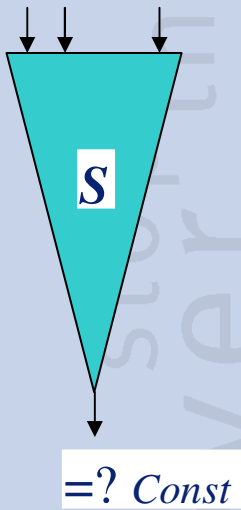


**Block enlargement**

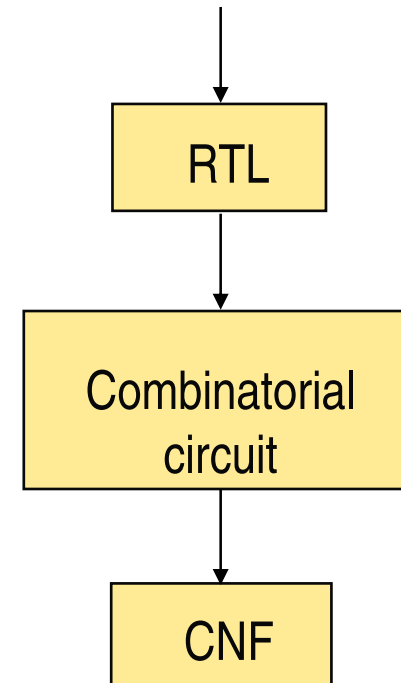
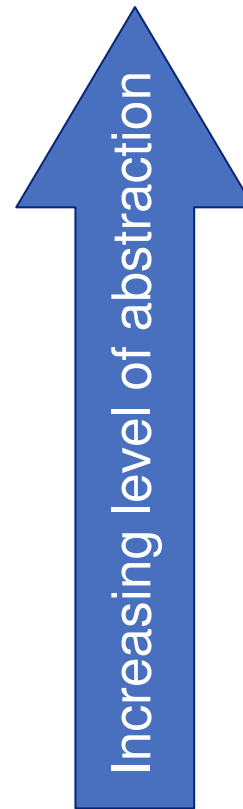


# Introduction: Block enlargement

We need a theory for solving structural SAT- instances when blocks implement complex functions.



**Block enlargement**



# Introduction: Basic idea

---

Operate on **blocks** in a similar way as **clauses** are processed in CNF-oriented SAT-solvers.

Using clauses

$a \vee b \vee \neg y$	
$a = b = 0 \Rightarrow y = 0$	<i>implication</i>
$a = b = 0, y = 1$	<i>conflict</i>

Using blocks

providing	<i>implications</i>
fixing	<i>conflicts</i>

# Table of contents

---

**1**

**Block Models**

**2**

**Implicativity**

**3**

**Systems as Blocks and Hierarchical SAT-Solving**

**4**

**Examples of models and modern advanced techniques in SAT**

**5**

**Conclusion**

## Block Models

---

### Principles of the theory

1

*Encapsulation:* storing conflicts and implications in block models

2

*Divide and conquer:* branching on variables describing interconnections between blocks

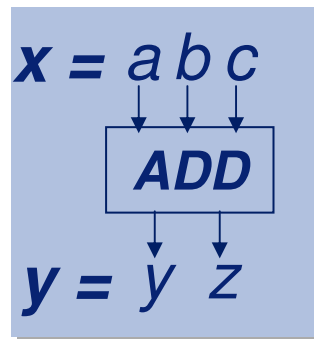
3

*Using different models for different kinds of blocks*

## Gate axioms

A block-model behaves like a combinatorial gate.

For each Block **B** there is:



1 Bit-Adder

**Axiom 1.** A finite nonempty set of **Boolean pin variables** (inputs  $\mathbf{x}$  and outputs  $\mathbf{y}$ ).

**Axiom 2.** A corresponding **Boolean vector function**  $\mathbf{y} = \Psi(\mathbf{x})$ .

<i>a</i>	<i>b</i>	<i>c</i>	<i>z</i>	<i>y</i>
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

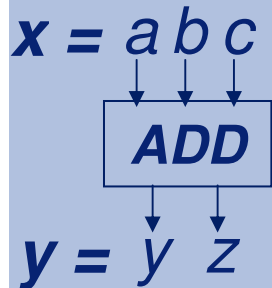
$$(y,z) = \Psi(a,b,c)$$

# Block Models: Axiomatic system (2)

## Assignment classification

A block-model can propagate partial assignments between the block pins and also classify them.

$$\alpha \in \{\text{conflicting, implying, not conflicting/implying}\}$$



1 Bit-Adder

For each Block  $B$  there is:

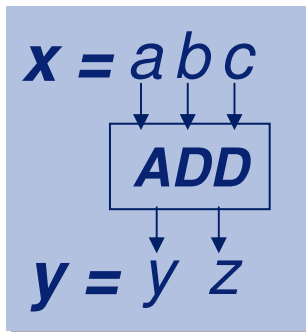
**Axiom 3.** A procedure “Boolean Constraint Propagation” **BCP** on the set  $A$  of all partial value assignments to the pin variables of  $B$  s.t. for each  $\alpha$  of  $A$ :

- BCP( $\alpha$ ) **does not change**  $\alpha$ .
- BCP( $\alpha$ ) classifies  $\alpha$  either as **conflicting** or **not conflicting**.
- If  $\alpha$  is not conflicting,  $\alpha$  is **extended** to  $\gamma = \alpha \cup \beta$  ( $\beta = \text{BCP}(\alpha)$ ) and, if  $\beta \neq \emptyset$ , then  **$\alpha$  implies  $\beta$**  ( $\alpha \Rightarrow \beta$ ).

$a$	$b$	$c$	$y$	$z$
-	1	-	0	0
<i>conf</i>				
-	0	-	0	0
$\Rightarrow a = 1$				

## Block-Models: Axiomatic system (3)

The classification of partial assignments of the BCP-procedure is consistent with the vector function  $\mathbf{y} = \Psi(\mathbf{x})$  correspondent to the block.



1 Bit-Adder

The characteristic function  $f(\mathbf{x}, \mathbf{y})$  of the set of all permissible complete assignments to the pin variables of a block  $\mathbf{B}$  is called **permission** (or characteristic) **function** for  $\mathbf{B}$ .

$a$	$b$	$c$	$z$	$y$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Permissible full assignments of

1-Bit-Adder

The clause  $\mathbf{c} = a \vee b \vee \neg y$  represents:

- the assignment  $a = 0, b = 0, y = 1$  (falsifying  $\mathbf{c}$ );
- implications  $\{a=0, b=0\} \Rightarrow \{y=0\}$ ,  $\{a=0, y=1\} \Rightarrow \{b=1\}$ ,  $\{b=0, y=1\} \Rightarrow \{a=1\}$ .

# Block-Models: Axiomatic system (4)

## Consistency of classifying assignment with permission function $f(x,y)$

<i>a</i>	<i>b</i>	<i>c</i>	<i>z</i>	<i>y</i>
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Permissible full ass. of 1-bit-Adder

**Axiom 4.** Let  $f(x,y)$  be the permission function for a block  $B$  then:

- If  $\alpha$  is classified as **conflicting**, then the clause  $c$  representing  $\alpha$  is **implicate** of  $f$ , i.e.  $(f \rightarrow c) = 1$ .
- If  $\alpha$  **implies**  $\beta$  ( $\beta = \text{BCP}(\alpha)$ ), then for each elementary assignment  $\beta_i$  of  $\beta$ , the clause representing the implication  $\alpha \Rightarrow \beta_i$  is **implicate** of  $f$ .

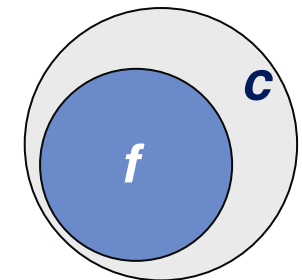
- Let  $b = 1, y = 0, z = 0$  is conflicting, then

$\neg b \vee y \vee z$  is implicate of  $f$

- Let  $(b = 0, y = 0, z = 0)$  implies  $(a = 1, c = 1)$  then

$b \vee y \vee z \vee a$  is implicate of  $f$

$b \vee y \vee z \vee c$  is implicate of  $f$



$(f \rightarrow c) = 1$

# Block-Models: Axiomatic system (5)

## Simulating and recognizing vector function $y = \Psi(x)$

BCP simulates the vector function  $y = \Psi(x)$  :

**Axiom 5.** For each full assignment  $\alpha$  to the inputs  $x$ ,  $\beta = \text{BCP}(\alpha) = \Psi(\alpha)$ .

$$(z = 0, y = 0) = \text{BCP}(a = 0, b = 0, c = 0) = \Psi(a = 0, b = 0, c = 0)$$

BCP recognizes non-permissible complete assignments:

**Axiom 6.** Let  $(\alpha, \beta)$  be a permissible complete assignment where  $\alpha$  is a full assignment to the inputs and  $\beta$  is a full assignment to the outputs, then for any elementary assignment  $\beta_i$  of  $\beta$  the assignment  $\alpha \cup \neg \beta_i$  is classified as conflicting.

$\alpha$                        $\beta$

$$a = 0, b = 0, c = 0 \quad z = 0, y = 0$$

$a = 0, b = 0, c = 0, z = 1$	conflicting
$a = 0, b = 0, c = 0, y = 1$	

a	b	c	z	y
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Permissible full ass. of 1-

Bit-Adder

## Block-Models: Axiomatic system (6)

---

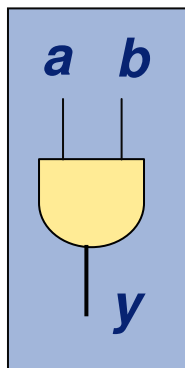
### Summary

- Two gate axioms  
(making a block similar to a gate)
- Four **BCP** axioms:
  - assignment classification (*conflicting, implying, not conflicting/implying*)
  - consistency of the classification with the *permission function*  $f(\mathbf{x}, \mathbf{y})$
  - simulating (implementing) the *vector function*  $\mathbf{y} = \Psi(\mathbf{x})$
  - recognizing  $\mathbf{y} = \Psi(\mathbf{x})$

# Block-Models: An example

## CNF-Based Block-Model

- A CNF  $\mathbf{C}(\mathbf{x})$  represents a Boolean function  $\mathbf{f}(\mathbf{x})$  iff  $\mathbf{C}(\mathbf{x}) = \mathbf{f}(\mathbf{x})$ .



- Consider an AND-Gate with  $y = a \wedge b$ .
- The CNF  $\mathbf{C} = (a \vee \neg y) \wedge (b \vee \neg y) \wedge (\neg a \vee \neg b \vee y)$  represents the permission function  $\mathbf{f}(a,b,y)$  of the gate.

$a$	$b$	$y$
0	0	0
0	1	0
1	0	0
1	1	1

ON-set of  $\mathbf{f}(a,b,y)$

**Theorem.** Given a CNF representing the permission function of a Block, the CNF is a model under CNF-BCP.

# Table of contents

---

1

Block Models

2

Implicativity

3

Systems as Blocks and Hierarchical SAT-Solving

4

Examples of models and modern advanced techniques in SAT

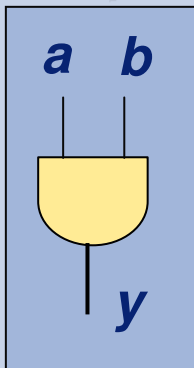
5

Conclusion

# Implicativity: Definition

**Implicativity** = Number of conflicting and implying partial assignments.

**Implicativity** can be measured by simulation.



Affecting assignments

<b>a</b>	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
<b>b</b>	-	-	-	0	0	0	1	1	1	-	-	-	0	0	0	1	1	1	-	-	-	0	0	0	1	1	1	
<b>y</b>	-	0	1	-	0	1	-	0	1	-	0	1	-	0	1	-	0	1	-	0	1	-	0	1	-	0	1	



Reactions

<b>a</b>	-	-	1	-	-	-	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
<b>b</b>	-	-	1	0	0	c	1	1	1	-	-	c	0	0	c	1	1	c	-	0	1	0	0	0	c	1	c	1
<b>y</b>	-	0	1	0	0	-	0	1	0	0	0	0	0	0	0	0	0	0	-	0	1	0	0	0	1	1	1	

CNF Model:

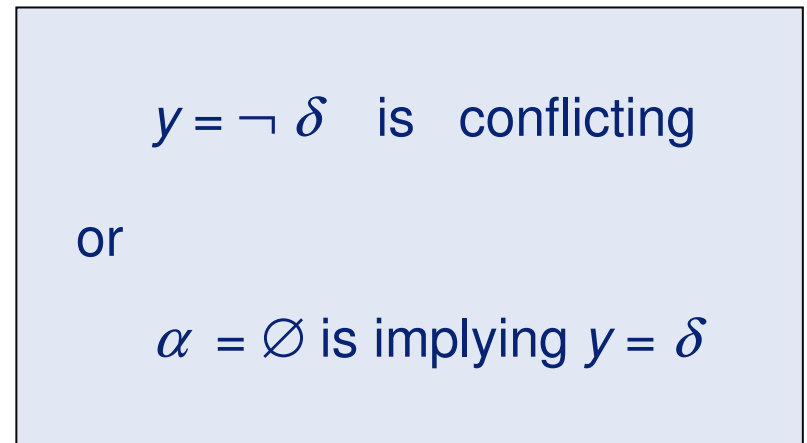
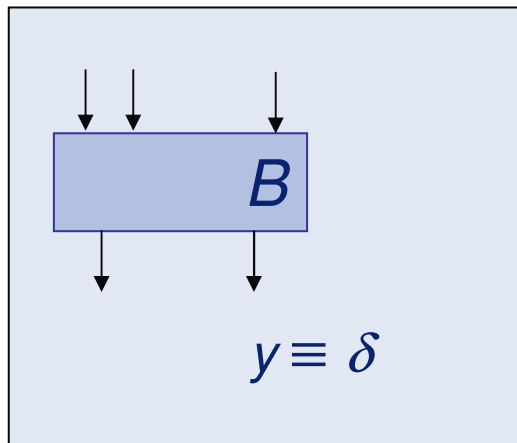
$$\begin{aligned}
 &(\neg a \vee \neg b \vee y) \\
 &\wedge (b \vee \neg y) \\
 &\wedge (a \vee \neg y)
 \end{aligned}$$

All partial assignments = 29  
 Implying assignments = 11  
 Conflicting assignments = 6  
**Implicativity** = 11 + 6 = 17

→ implication conflict  
 \*

# Implicativity: Fundamental theorem

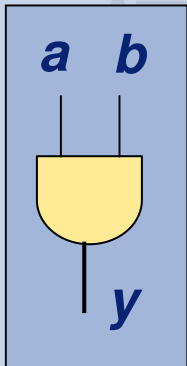
**Theorem.** Let a model  $M$  have maximal implicativity among all models for a block  $B$ . Let  $y$  be an output of  $B$ . The output  $y$  implements a constant Boolean function  $\delta$  where  $\delta \in \{0,1\}$  iff the elementary assignment  $y = \neg \delta$  is conflicting or the empty assignment  $\alpha = \emptyset$  implies  $y = \neg \delta$  for the model  $M$ .



# Implicativity: Example of a model with maximal implicativity

A clause  $c$  is **prime implicate** of  $f$  if  $c$  is not covered by (is part of) any other implicate of  $f$ .

The **characteristic CNF** consists of all prime implicates of the permission function  $f$ .



The conventional CNF  $C = (a \vee \neg y) \wedge (b \vee \neg y) \wedge (\neg a \vee \neg b \vee y)$  for an AND-gate is the characteristic CNF.

**Theorem.** The **characteristic CNF** of a block is a model with **maximal implicativity** under CNF-BCP.

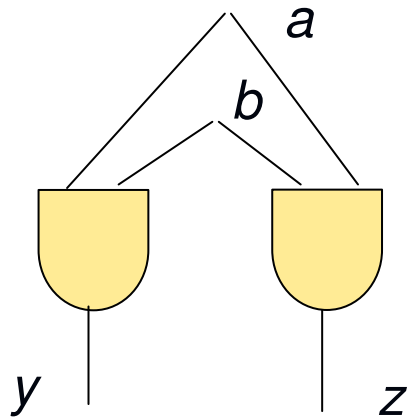
$a$	$b$	$y$
0	0	0
0	1	0
1	0	0
1	1	1

ON-set of  $f(a,b,y)$

CNF-BCP:  
recursive application  
of the unit clause rule

$$\begin{array}{l}
 a \vee b \vee \neg y \\
 a = b = 0 \Rightarrow y = 0 \quad \textit{implication} \\
 a = b = 0, y = 1 \quad \textit{conflict}
 \end{array}$$

# Implicativity: Conventional CNF is not a model with maximal implicativity in the general case



Equivalent gates

Conventional CNF :

$$C = (a \vee \neg y) \wedge (b \vee \neg y) \wedge (\neg a \vee \neg b \vee y) \wedge (a \vee \neg z) \wedge (b \vee \neg z) \wedge (\neg a \vee \neg b \vee z)$$

CNF-BCP doesn't classify the partial assignment  $y = 0$  as implying  $z = 0$ .

However, adding the resolvent  $y \vee \neg z$  of  $\neg a \vee \neg b \vee y$ ,  $a \vee \neg z$ , and  $b \vee \neg z$  provides the implication:

$$y = 0 : y \vee \neg z \Rightarrow z = 0$$

# Table of contents

---

**1**

**Block Models**

**2**

**Implicativity**

**3**

**Systems as Blocks and Hierarchical SAT-Solving**

**4**

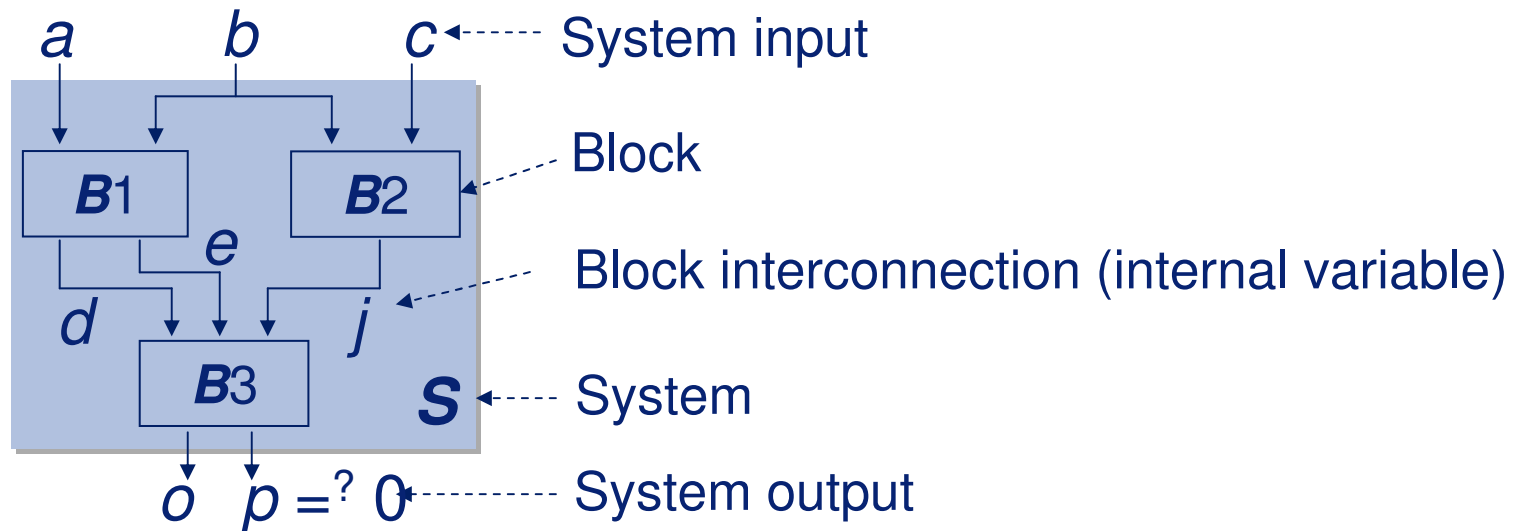
**Examples of models and modern advanced techniques in SAT**

**5**

**Conclusion**

# Systems as Blocks and Hierarchical SAT-solving

## A normal system



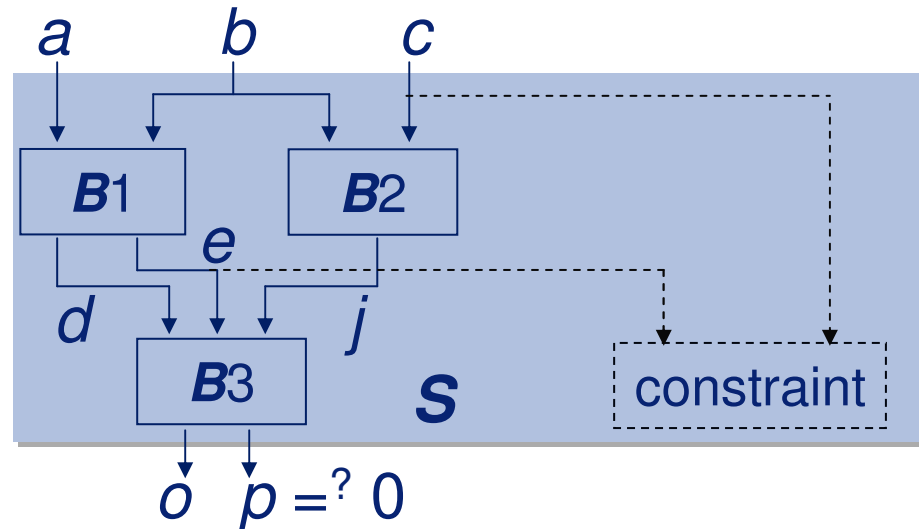
A **normal** block  $B$  has **at least one** output.

A **normal** system  $S$  consists of normal blocks.

The vector function  $\mathbf{y} = \Psi(\mathbf{x})$  of a normal system is equal to the superposition of vector functions implemented by its blocks.

## Systems as Blocks and Hierarchical SAT-solving (2)

### A constrained system



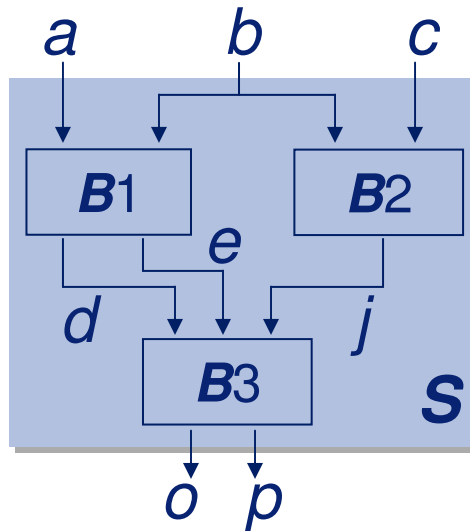
A **block-constraint**  $B$  has no outputs.

A **constrained system** = a normal system + block-constraints

Constrained system is called **system** (for short)

# Systems as Blocks and Hierarchical SAT-solving (3)

## Extended permission function



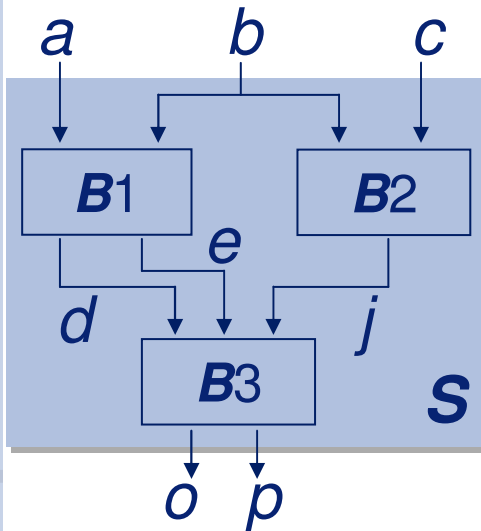
- Let  $\mathbf{S}$  consist of
  - Blocks  $B_1 \dots B_k$  with
  - Permission functions  $f_1 \dots f_k$ , connected by
  - Internal variables  $\mathbf{z}$ .
- Then  $f^*(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \bigwedge_{i=1..k} f_i$  is called **extended permission function.**

### **Fitting axiom:**

Let  $B$  be a block-constraint of a system  $\mathbf{S}$ , and let  $f_i$  be its permission function, and let  $f^*$  be the extended permission function of  $\mathbf{S}$ . Then  $(f^* \rightarrow f_i) = 1$  ( $f^*$  implies  $f_i$ ).

# Systems as Blocks and Hierarchical SAT-solving (4)

## Permission function



The operator  $\exists a$  for *existential quantification* of a Boolean function  $f$  w.r.t. a variable  $a$  is defined as follows:

$$\exists a f(x_0, \dots, a, \dots, x_n) = f(x_0, \dots, 0, \dots, x_n) \vee f(x_0, \dots, 1, \dots, x_n).$$

**Theorem.** The permission function  $f(x, y)$  of a system  $S$  is equal to its extended permission function existentially quantified by internal variables  $z$  of  $S$

$$\exists z : f^*(x, y, z) = f(x, y)$$

$$f(a, b, c, o, p) = \exists d, e, j : f_1(a, b, d, e) \wedge f_2(b, c, j) \wedge f_3(d, e, j, o, p)$$

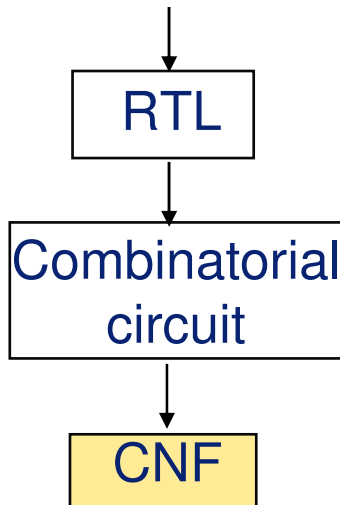
# Systems as Blocks and Hierarchical SAT-solving (5)

## SYSTEM-BCPs

We define two SYSTEM-BCP procedures P1 and P2.

**Theorem.** A system of blocks is normal block under the procedures P1 and P2.

**P1** A class of procedures **P2**



On CNF-level: P1 and P2 are similar to the BCP procedures used in modern SAT-solvers.

## Systems as Blocks and Hierarchical SAT-solving (6)

---

### Hierarchical SAT-solving

Since block-models are used for the same activity as clauses in CNF-oriented SAT-solvers (i.e. for fixing conflicts and producing implications) hierarchical SAT-solving can be organized **in a similar way** as CNF SAT-solving.

#### CNF

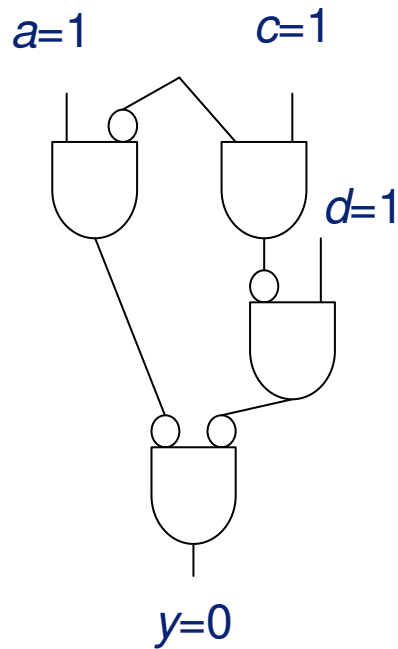
- Search tree traversal
- Restarts
- Decision strategies
- BCP (watched literals)
- Conflict analysis and non-chronological backtracking
- Data Base Management

#### General case

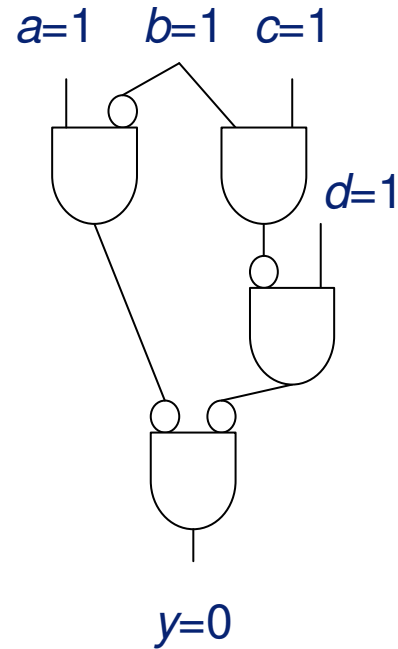
- Using structure
- Accessing functions
- Reason reduction
- Block-constraints

# Systems as Blocks and Hierarchical SAT-solving (6)

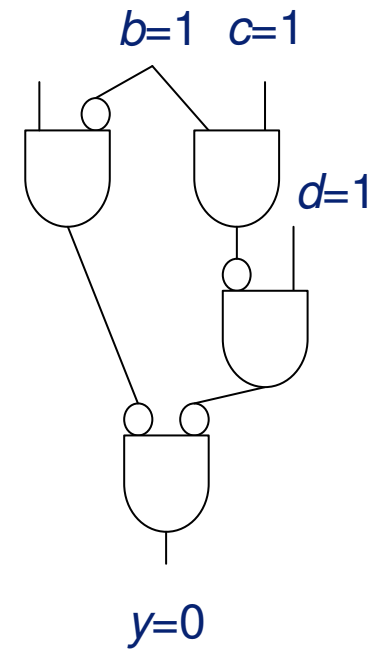
## Reason reduction



a)  $a=1, c=1, d=1, y=0$   
(not conflicting/implying)



b)  $a=1, b=1, c=1, d=1, y=0$   
(conflicting)



c)  $a=1, c=1, d=1, y=0$   
(reduced conflicting)

## Systems as Blocks and Hierarchical SAT-solving (6)

---

### Learning

**Learning:** Adding block-constraints satisfying the fitting axiom

**Theorem 1.** Learning does not reduce the implicativity of a system.

**Theorem 2.** There always exists a block-constraint increasing the implicativity of a system up to the maximum.

**Theorem 3.** If there is no counter-example, hierarchical SAT-solving results in reaching maximal implicativity for the system.

**Hierarchical SAT-solving is reduced  
to increasing implicativity**

## Systems as Blocks and Hierarchical SAT-solving (6)

---

### SAT-models

A way of increasing implicativity of a system is increasing implicativity of its blocks.

Constructing block models **is a new direction of research.**

#### Block-models

- **Increased implicativity** (compared to conventional CNF-based models)
- **Fast BCP** (to quickly extract information stored in the model)
- **Acceptable size** (for practical use)

# Table of contents

---

**1**

**Block Models**

**2**

**Implicativity**

**3**

**Systems as Blocks and Hierarchical SAT-Solving**

**4**

**Examples of models and modern advanced techniques in SAT**

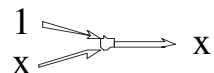
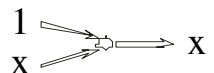
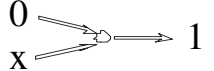
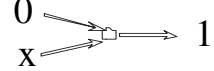
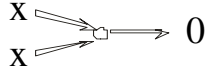
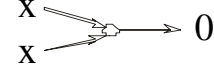
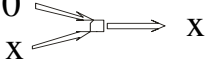
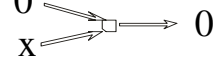
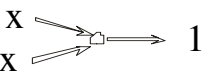
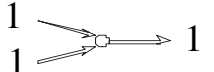
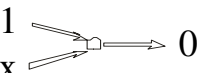
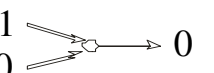
**5**

**Conclusion**

# Examples of models and modern advanced techniques in SAT

## Circuit-oriented SAT-solving

[M.K.Ganai et al, 2002], [F.Lu et al, 2003]

Current	Next	Action
		$\emptyset$
		conflicting
		$\emptyset$
		implying
		implying
		implying
...	...	...

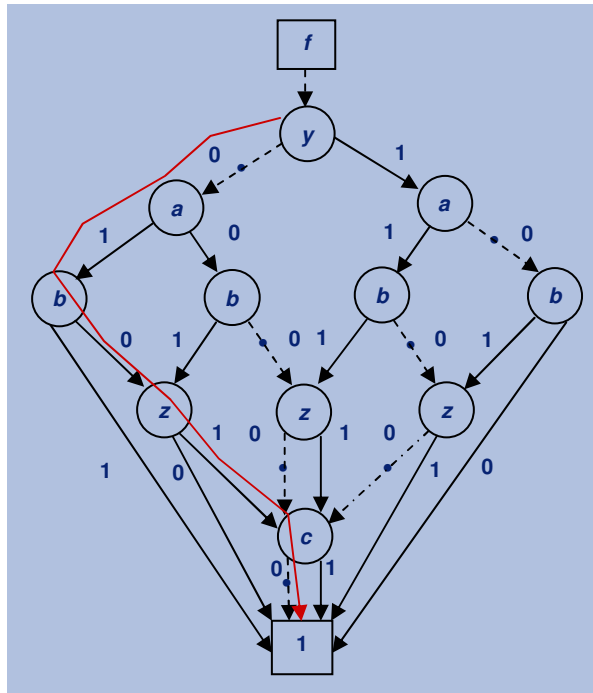
- Maximal implicativity
- Fast access
- Exponential growth of size

2-input AND lookup table [A.Kühlmann et al, 2001]

# Examples of models and modern advanced techniques in SAT (2)

## BDD-based models

[A.Gupta et al, 1997], [S.Redha et al, 2001], [R.Damiano et al, 2003]



BDD for the permission function of 1-bit-Adder

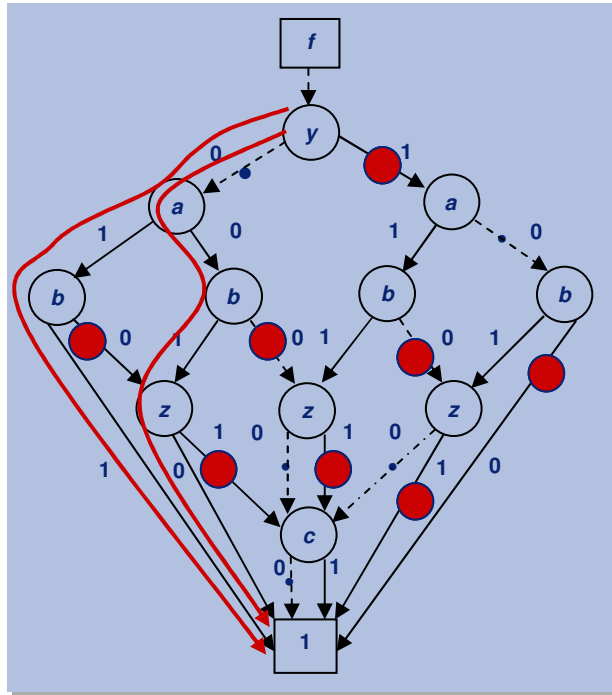
We extend their results by considering reduced ordered BDDs with complemented edges.

---●---> complemented edge

$f = 1$  on  $(y = 0, a = 1, b = 0, z = 1, c = 0)$  as the number of complemented edges on the red path is even.

# Examples of models and modern advanced techniques in SAT (3)

## BDD-based models: fixing conflicts



● - removed edge

The assignment  $(y, a, b, z, c) = (0, x, 1, 0, x)$ , is conflicting.

Graph reduction:

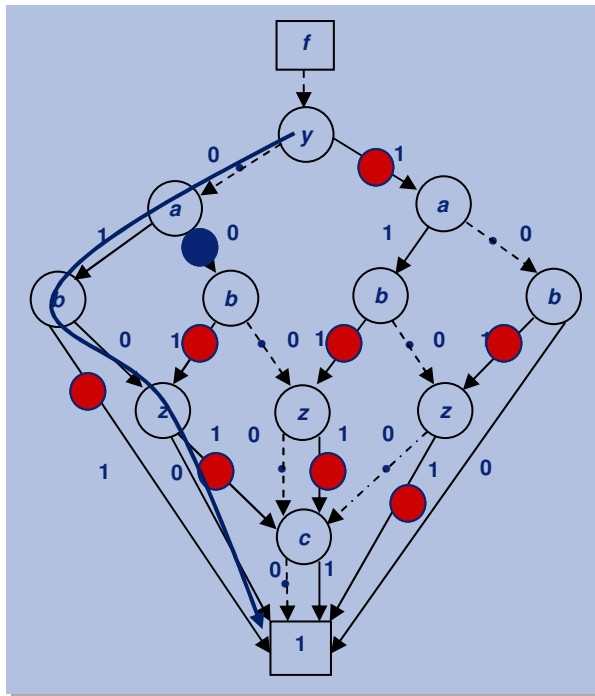
If  $x$  is assigned to 0, remove edges  $x = 1$

Under assignment  $\alpha$  the graph is transformed to  $D_\alpha$

An assignment  $\alpha$  is conflicting, if **any path** leading from the source node  $f$  to the sink node 1 in the reduced graph  $D_\alpha$  contains an **odd number** of complemented edges.

# Examples of models and modern advanced techniques in SAT (4)

## BDD-based models: deducing implications



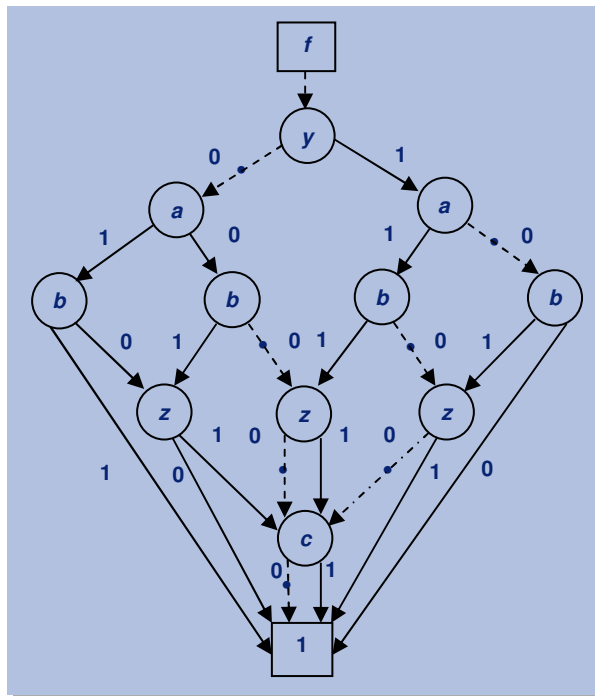
● - removed edge

The assignment  $(y, a, b, z, c)$   
 $(0, x, 0, 0, x)$ , implies  $a = 1$ .

An assignment  $\alpha$  implies an elementary assignment  $\beta_i$ , if  $\alpha$  is not conflicting and the assignment  $(\alpha, \neg\beta_i)$  is conflicting.

An assignment  $\alpha$  implies an elementary assignment  $x=\delta$ , if  $\alpha$  is not conflicting and **any path** leading from the source node  $f$  to the sink node 1 in  $D_{\alpha \cup (x=\neg\delta)}$  contains an **odd number** of complemented edges and passes through a node marked with  $x$ .

## BDD-based models: conclusion



BDD for the permission function of 1-Bit-Adder

- Implicativity: maximal
- Access: three traversals of the graph
- Size: depends on functions used

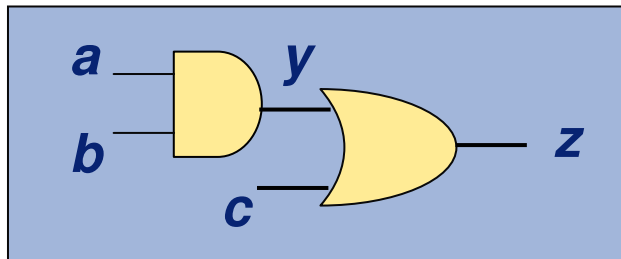
We need models for typical blocks or structures of the application domain

# Examples of models and modern advanced techniques in SAT (6)

## Tree-like structures: Gate merging

[M.N.Velev, 2004]

$  \begin{array}{l}  a \vee \neg y \\  b \vee \neg y \\  \neg a \vee \neg b \vee y  \end{array}  $	$  \begin{array}{l}  \neg a \Rightarrow \neg y \\  \neg b \Rightarrow \neg y \\  a \wedge b \Rightarrow y  \end{array}  $	$  \begin{array}{l}  \neg y \vee z \\  \neg c \vee z \\  y \vee c \vee \neg z  \end{array}  $	$  \begin{array}{l}  y \Rightarrow z \\  c \Rightarrow z \\  \neg y \wedge \neg c \Rightarrow \neg z  \end{array}  $	$  \begin{array}{l}  c \Rightarrow z \\  a \wedge b \Rightarrow z \\  \neg a \wedge \neg c \Rightarrow \neg z \\  \neg b \wedge \neg c \Rightarrow \neg z  \end{array}  $	$  \begin{array}{l}  \neg c \vee z \\  \neg a \vee \neg b \vee z \\  a \vee c \vee \neg z \\  b \vee c \vee \neg z  \end{array}  $
CNF model for AND gate	Implication model for AND gate	CNF model for OR gate	Implication model for OR gate	Implication model for the system	CNF model for the system

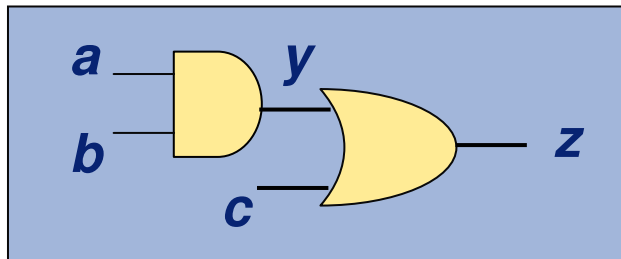


# Examples of models and modern advanced techniques in SAT (7)

## Tree-like structures: Resolution based method

[S. Subbarayan et al, 2004]

$a \vee \neg y$ $b \vee \neg y$ $\neg a \vee \neg b \vee y$	$\neg y \vee z$ $\neg c \vee z$ $y \vee c \vee \neg z$	$\neg a \vee \neg b \vee y, \neg y \vee z$ $a \vee \neg y, y \vee c \vee \neg z$ $b \vee \neg y, y \vee c \vee \neg z$	$\neg a \vee \neg b \vee z$ $a \vee c \vee \neg z$ $b \vee c \vee \neg z$	$\neg c \vee z$ $\neg a \vee \neg b \vee z$ $a \vee c \vee \neg z$ $b \vee c \vee \neg z$
CNF model for AND gate	CNF model for OR gate	resolved pair of clauses	resolvents	resulting CNF for the system



NiVER [2004]: VER [M. Davis et al, 1960]

# Examples of models and modern advanced techniques in SAT (8)

---

## Tree-like structures: CNF model

- Gate merging
- Resolution based method
- Existential quantification



**CNF model with  
maximal implicativity**

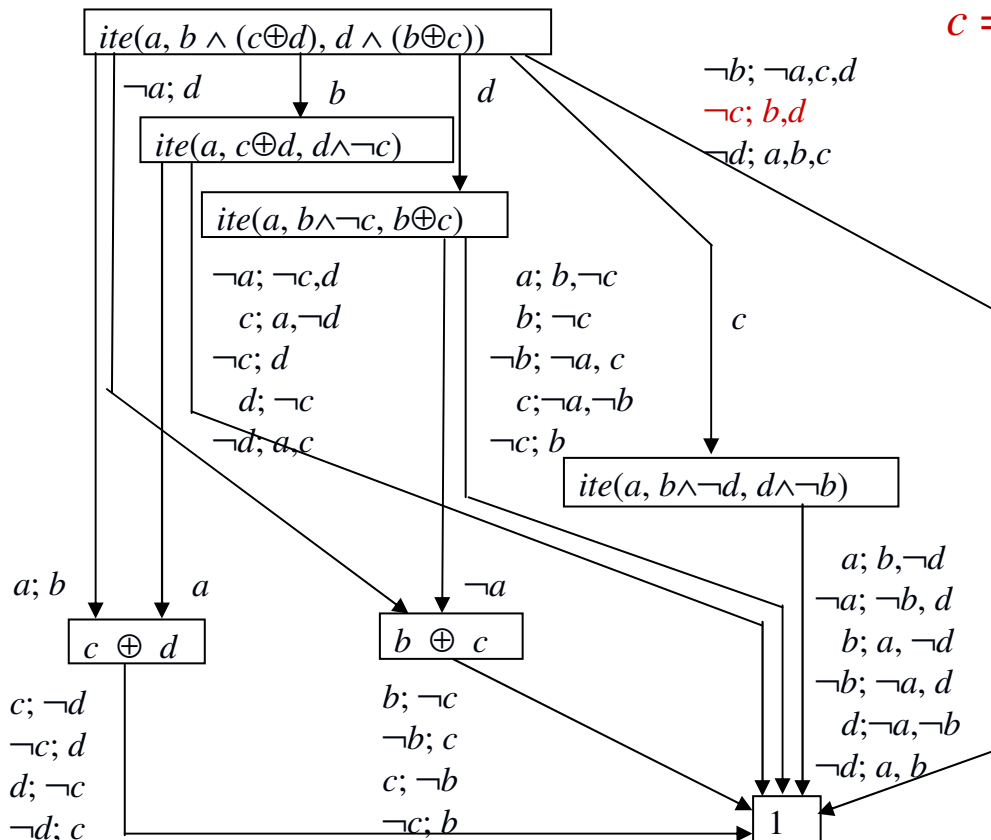
- Maximal implicativity
- Access: CNF-BCP
- Size: depends on functions used

# Examples of models and modern advanced techniques in SAT (9)

## SMURFs – State machines for representation of Boolean functions

SMURF for  
 $ite(a, b \wedge (c \oplus d), d \wedge (a \oplus b))$

[J. Franko et al, 2004]



$$c = 0 \Rightarrow b = 1, d = 1$$

- Maximal implicativity
- Access: fast
- Size: depends on functions used

BDD  $\leq$  SMURF

$$O(2^n)$$

$$O(3^n)$$

Never stop thinking

# Examples of models and modern advanced techniques in SAT (10)

## Using pseudo-Boolean constraints (PBCs)

[F.A.Aloul et al, 2002]

**PBC:**  $a_1y_1 + a_2y_2 + \dots + a_ny_n \leq b$

where  $a_i, b \in \mathbf{N}$  and  $y_i$  denotes either  $x_i$  or  $\neg x_i$

**Example:  $a + b + c + d \leq 2$**

$a = b = 1 \Rightarrow c = d = 0$

$a = b = c = 1$  - conflict

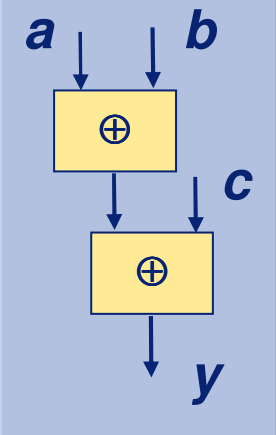
- Maximal implicativity
- Access: fast
- Size: linear
- Domain: threshold functions

# Examples of models and modern advanced techniques in SAT (11)

## Using multiple exclusive ORs

[J.A.Roy et al, 2004]

$$x_1 \oplus x_2 \oplus \dots \oplus x_n \oplus \delta, \delta \in \{0,1\}$$



EX-OR chain

**Permission function:**

$$a \oplus b \oplus c \leftrightarrow y$$

or

$$a \oplus b \oplus c \oplus y \oplus 1$$

**Example:**  $a \oplus b \oplus c \oplus d$

$$a = b = c = 1 \Rightarrow a \oplus b \oplus c = 1$$

$$\Rightarrow d = 0$$

$$a = b = c = d = 1 - \text{conflict}$$

- Maximal implicativity
- Access: fast
- Size: linear
- Domain: linear functions

# Examples of models and modern advanced techniques in SAT (12)

## Arithmetic reasoning

[M.Wedler et al, 2004]

	8	7	6	5	4	3	2	1
1					$a_{14} = 0$	$a_{13} = 1$	$a_{12} = 0$	$a_{11} = 1$
X				$a_{24} = X$	$a_{23} = X$	$a_{22} = X$	$a_{21} = X$	
1			$a_{34} = 0$	$a_{33} = 1$	$a_{32} = 0$	$a_{31} = 1$		
0		$a_{44} = 0$	$a_{43} = 0$	$a_{42} = 0$	$a_{41} = 0$			
	$c_{17} = 0$	$c_{16} = 0$	$c_{15} = 0$	$c_{14} = 0$	$c_{13} = 1$	$c_{12} = 0$		
		$c_{26} = 0$	$c_{25} = 0$	$c_{24} = 0$	$c_{23} = 0$			
			$c_{35} = X$	$c_{34} = X$				
	$b_8 = 0$	$b_7 = 0$	$b_6 = X$	$b_5 = X$	$b_4 = X$	$b_3 = X$	$b_2 = X$	$b_1 = 1$

$a_{ij}$  – bits of addends, 
 
 $c_{ij}$  – carry bits, 
 
 $b_i$  – bits of the product

Arithmetic reasoning for 4\*4-multiplier: multiplying 0 1 X 1 on 0 1 0 1

# Examples of models and modern advanced techniques in SAT (13)

---

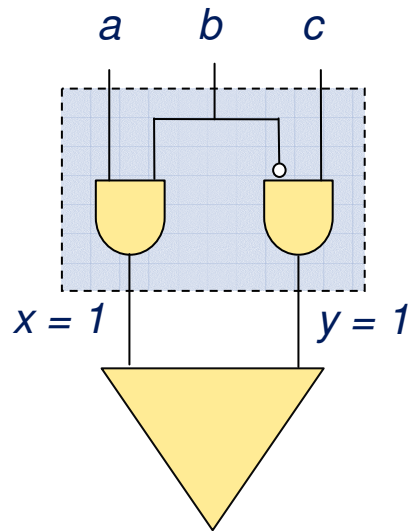
## Arithmetic reasoning: conclusion

- Implicativity: increased
- Access: fast
- Size: quadratic
- Domain: multipliers

# Examples of models and modern advanced techniques in SAT (14)

## Managing don't cares

[S.Safarpour et al, 2004]



Controllability  
don't care

### Conventional CNF

$$(a \vee \neg x) \wedge (b \vee \neg x) \wedge (\neg a \vee \neg b \vee x) \\ \wedge (\neg b \vee \neg y) \wedge (c \vee \neg y) \wedge (b \vee \neg c \vee x)$$

$b \vee \neg x$
$\neg b \vee \neg y$
$\neg x \vee \neg y$

adding  $\neg x \vee \neg y$

provides  $x = 1 \Rightarrow y = 0$

- Implicativity: increased
- Access: CNF-BCP
- Size: depends on the method used

# Examples of models and modern advanced techniques in SAT (15)

---

## Resume

- We have considered almost a dozen of newest advanced techniques in SAT-solving.
- All the authors report about substantial progress (sometimes speed-ups reach a few orders of magnitude).
- At first glance, all these techniques are disjoint, independent, and are based on different ideas and mathematics.
- However, they are all covered by the general theory of “Hierarchical SAT-solving”.

# Table of contents

---

**1**

**Block Models**

**2**

**Implicativity**

**3**

**Systems as Blocks and Hierarchical SAT-Solving**

**4**

**Examples of models and modern advanced techniques in SAT**

**5**

**Conclusion**

## Conclusion: Summary

---

- We propose a theoretical foundation for hierarchical SAT-solving.
- We introduce the notion of implicativity.
- We show that hierarchical SAT-solving reduces to increasing the implicativity of a system.
- We show that many research topics are cases of hierarchical SAT-solving, as:
  - They increase the level of abstraction of SAT-solving by constructing block models with maximal or increased implicativity.
- SAT-model construction is a new promising direction of research.
- This research is relevant for industrial application.

## Conclusion: Ideal Block-Models

---

- Maximal (high) implicativity
- Compact representation
- Fast BCP
- Good accessing function:
  - Close over-approximation of the set of implying or conflicting partial assignments.
  - Needed for replacing 2-Watched-Literal scheme to avoid blank processing.
- Fast and effective Reverse-BCP for reason reduction (efficient learning)

## Conclusion: Ideal Block-Models

---

- Maximal (high) implicativity
- Compact representation
- Fast BCP
- Good accessing function
- Fast and effective Reverse-BCP for reason reduction

**You are welcome to join in!**



Never stop thinking



Never stop thinking.



Never stop thinking.



Never stop thinking.

## A System-BCP-Procedure

---

- Given  $\alpha = (x_1=v_1, \dots, x_n=v_n)$  (partial ass. to system pins).
- 1** Let  $\gamma = (x_1=v_1, \dots, x_n=v_n)$ ,  $\rho = ()$ ,  $t = n$ ,  $b = 0$ ; (deduced ass., list of reasons, top, bottom pointers)
- 2** For each block  $B_j$  having  $x_b$  as pin variable, let  $\gamma_j \subseteq (\gamma_1, \dots, \gamma_b)$  be the corresponding partial assignment, run BCP( $\gamma_j$ )
  - a** On conflict, report  $\alpha$  as conflicting (with direct reason  $\gamma_j$ ).
  - b** On implication  $\beta = (\beta_1, \dots, \beta_k) = \text{BCP}_j(\gamma_j)$ 
    - i** If  $\beta_i = \neg\gamma_l$  for some  $(i=1..k, l=1..t)$  report  $\alpha$  as conflicting (with direct reason  $(\neg\beta_i, \beta_i)$ ).
    - ii** Else, add  $\beta$  at the end of  $\gamma$ , set  $\rho_{t+i} = \gamma_j$  ( $i=1..k$ ), and set  $t := t+k$ .
- 3** Set  $b := b+1$ . If  $b \leq t$  repeat step 2.
- 4** If new system pin variables assigned, report implication.
- Early implication: At step 2, let  $\gamma_j \subseteq (\gamma_1, \dots, \gamma_t)$  (take all deduced assignments)

## A Reverse System-BCP-procedure

---

- Use  $\rho$  of System-BCP for finding indirect reason for a given partial assignment (conflict analysis, reason reduction):
  - 1 Mark all elementary assignments of the direct reason in  $\gamma$
  - 2 Traverse backward through  $\gamma$  (let  $i=t$ ).
  - 3 If  $\gamma_i$  is marked,
    - If its reason is empty, add  $\gamma_i$  to the resulting indirect reason.
    - Otherwise mark all elementary assignments occurring in the reason of  $\gamma_i$
  - 4 If  $i = 0$  stop, otherwise  $i := i-1$  and repeat step 3.