



Sudoku Solutions Using Logic Equations

Christian Posthoff

The University of The West Indies, Trinidad & Tobago

Bernd Steinbach

Freiberg University of Mining and Technology, Germany



Outline

- Introduction
- Modeling Using a SAT – Instance
- Modeling Using a System of Logic Equations
- Basic Approaches of Parallel SAT-Solving
- New Approach: Implicit 2-Phase SAT-Solver
 - Basic Idea
 - First Phase
 - Second Phase
- Experimental Results
- Conclusions



Introduction

- Basic Definitions

- Boolean Space

$$B^n = \{(x_1, \dots, x_n) \mid x_i \in \{0,1\}, \forall i = 1, \dots, n\}$$

- disjunction (clauses) of variables or complemented variables (literals)

$$D = x_1 \vee \dots \vee \bar{x}_k$$

- SAT-Problem

- most general: Find all solutions of this equation!

$$C = D_1 \wedge \dots \wedge D_l = 1$$



Introduction

- SAT-Subproblems
 - Find some solutions!
 - Confirm that there are solutions!
 - Confirm that there is no solution!
- Complexity of the SAT-Problem
 - SAT is an NP-complete problem!
 - Cook-Levin theorem
- Application of SAT-Solver
 - verification in circuit design
 - decision in artificial intelligence

Introduction

- Logic Function f of n Variables \underline{x}



$$f(\underline{x}): B^n \rightarrow B$$

- Logic Equation



$$f(\underline{x}) = g(\underline{x})$$

- no restriction for the expressions of the functions

- Solution: set of Boolean vectors with $0 \equiv 0$ or $1 \equiv 1$

- System of Logic Equations



$$\left. \begin{array}{l} f_1(\underline{x}) = g_1(\underline{x}) \\ \vdots \\ f_k(\underline{x}) = g_k(\underline{x}) \end{array} \right|$$

- Solution: set of Boolean vectors that solves each equation of the system

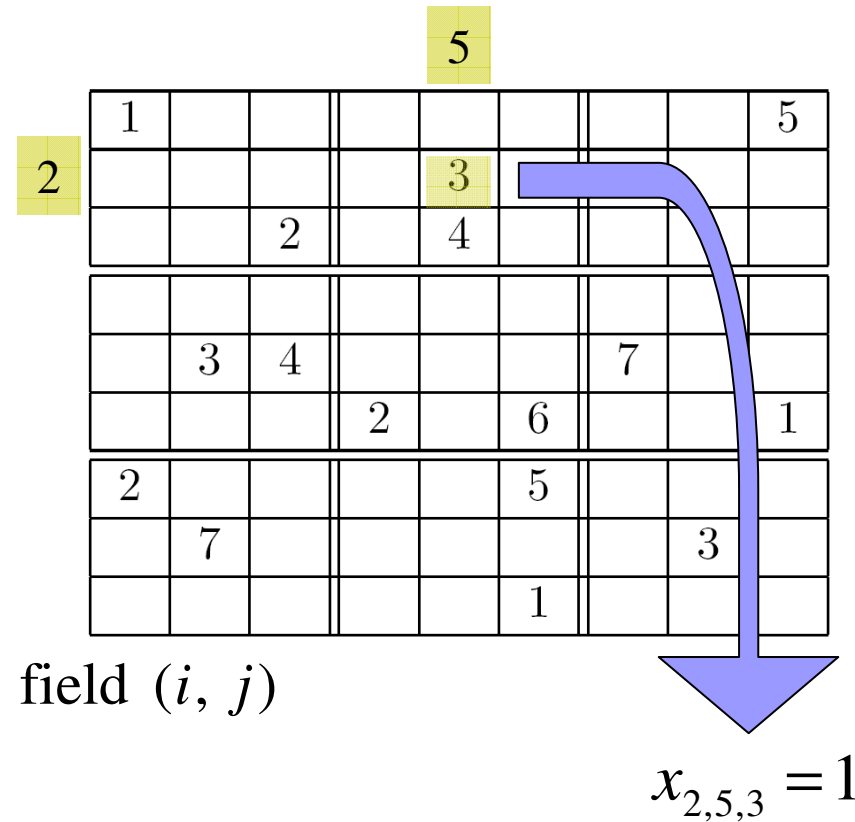
Modeling Using a SAT – Instance

- Example - Sudoku

- Boolean model

- row: i
 - column: j
 - value: k

$$x_{i,j,k} = \begin{cases} 1 & \text{if } k \text{ is on the field } (i, j) \\ 0 & \text{otherwise} \end{cases}$$



Modeling Using a SAT – Instance

- Two Types of Laws
 - 1. requirements

Rule	Clause
There must be a value 1 in the first row.	$(x_{1,1,1} \vee x_{1,2,1} \vee x_{1,3,1} \vee x_{1,4,1}) = 1$
There must be a value 1 in the first column.	$(x_{1,1,1} \vee x_{2,1,1} \vee x_{3,1,1} \vee x_{4,1,1}) = 1$
There must be a value 1 in the top left subsquare.	$(x_{1,1,1} \vee x_{1,2,1} \vee x_{2,1,1} \vee x_{2,2,1}) = 1$
There must be one of the values in the field (1,1).	$(x_{1,1,1} \vee x_{1,1,2} \vee x_{1,1,3} \vee x_{1,1,4}) = 1$

- defined for

	1	2	3	4
1				
2				
3				
4				

- each row and value according to the first line
- each column and value according to the second line
- each subsquare and value according to the third line
- each field (row, column) according to the fourth line

- number of literals in requirement clauses: n

Modeling Using a SAT – Instance

- Two Types of Laws
 - 2. restrictions

	1	2	3	4
1				
2				
3				
4				

Rule	Implication
The value 1 in the field (1,1) prohibits other values in this field.	$(x_{1,1,1} \rightarrow \bar{x}_{1,1,2})(x_{1,1,1} \rightarrow \bar{x}_{1,1,3})$ $(x_{1,1,1} \rightarrow \bar{x}_{1,1,4}) = 1$
The value 1 in the field (1,1) prohibits the value 1 in other fields of the first row.	$(x_{1,1,1} \rightarrow \bar{x}_{1,2,1})(x_{1,1,1} \rightarrow \bar{x}_{1,3,1})$ $(x_{1,1,1} \rightarrow \bar{x}_{1,4,1}) = 1$
The value 1 in the field (1,1) prohibits the value 1 in other fields of the first column.	$(x_{1,1,1} \rightarrow \bar{x}_{2,1,1})(x_{1,1,1} \rightarrow \bar{x}_{3,1,1})$ $(x_{1,1,1} \rightarrow \bar{x}_{4,1,1}) = 1$
The value 1 in the field (1,1) prohibits the value 1 in other fields of the subsquare.	$(x_{1,1,1} \rightarrow \bar{x}_{1,2,1})(x_{1,1,1} \rightarrow \bar{x}_{2,1,1})$ $(x_{1,1,1} \rightarrow \bar{x}_{2,2,1}) = 1$

□ using $a \rightarrow b = \bar{a} \vee b$

□ it follows for a single basic value

$$(\bar{x}_{1,1,1} \vee \bar{x}_{1,1,2})(\bar{x}_{1,1,1} \vee \bar{x}_{1,1,3})(\bar{x}_{1,1,1} \vee \bar{x}_{1,1,4})(\bar{x}_{1,1,1} \vee \bar{x}_{1,2,1})(\bar{x}_{1,1,1} \vee \bar{x}_{1,3,1})(\bar{x}_{1,1,1} \vee \bar{x}_{1,4,1})$$

$$\wedge (\bar{x}_{1,1,1} \vee \bar{x}_{2,1,1})(\bar{x}_{1,1,1} \vee \bar{x}_{3,1,1})(\bar{x}_{1,1,1} \vee \bar{x}_{4,1,1}) (\bar{x}_{1,1,1} \vee \bar{x}_{2,2,1}) = 1.$$

□ number of literals in restrictive clauses: 2



Modeling Using a SAT – Instance

■ Numbers of clauses for a 9 x 9 Sudoku

(known from the literature)

- Lynce, I. and Ouaknine, J.: Sudoku as a SAT Problem. 9th International Symposium on Artificial Intelligence and Mathematics, January 2006

- minimal encoding 8829 clauses
- extended encoding 11988 clauses

- Weber, T: A SAT-based Sudoku Solver. TU Munich, November 2005.

- one encoding gives only 11745 clauses

Modeling Using a System of Logic Equations

■ Restrictions

- Constraints can be stated by one single conjunction for each number on each field.
- The conjunction

$$K_{111} = x_{111}\bar{x}_{112}\bar{x}_{113}\bar{x}_{114}\bar{x}_{115}\bar{x}_{116}\bar{x}_{117}\bar{x}_{118}\bar{x}_{119} \bar{x}_{121}\bar{x}_{131}\bar{x}_{141}\bar{x}_{151}\bar{x}_{161}\bar{x}_{171}\bar{x}_{181}\bar{x}_{191} \wedge \\ \wedge \bar{x}_{211}\bar{x}_{311}\bar{x}_{411}\bar{x}_{511}\bar{x}_{611}\bar{x}_{711}\bar{x}_{811}\bar{x}_{911}\bar{x}_{221}\bar{x}_{231}\bar{x}_{321}\bar{x}_{331}.$$

describes completely the setting of 1 on the field (1,1) and **all the consequences**.

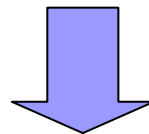
- There are 729 of such conjunctions for a 9 x 9 Sudoku which are defined uniquely.

Modeling Using a System of Logic Equations

■ Requirements

- Each of the four types of requirements can be expressed in terms of the 9 variables $x_{i,j,k}$ but also in terms of the associated 9 conjunctions $K_{i,j,k}$, e.g.

$$(x_{ij1} \vee x_{ij2} \vee x_{ij3} \vee x_{ij4} \vee x_{ij5} \vee x_{ij6} \vee x_{ij7} \vee x_{ij8} \vee x_{ij9}) = 1$$



$$(K_{ij1} \vee K_{ij2} \vee K_{ij3} \vee K_{ij4} \vee K_{ij5} \vee K_{ij6} \vee K_{ij7} \vee K_{ij8} \vee K_{ij9}) = 1$$

- Taking into consideration the conjunctions $K_{i,j,k}$ **all the consequences** resulting from a given setting are used immediately.



Modeling Using a System of Logic Equations

■ Four Types of Requirements

- value in a selected field

$$(K_{111} \vee K_{112} \vee K_{113} \vee K_{114} \vee K_{115} \vee K_{116} \vee K_{117} \vee K_{118} \vee K_{119}) = 1$$

- fixed value in a selected row

$$(K_{111} \vee K_{121} \vee K_{131} \vee K_{141} \vee K_{151} \vee K_{161} \vee K_{171} \vee K_{181} \vee K_{191}) = 1$$

- fixed value in a selected column

$$(K_{111} \vee K_{211} \vee K_{311} \vee K_{411} \vee K_{511} \vee K_{611} \vee K_{711} \vee K_{811} \vee K_{911}) = 1$$

- fixed value in a selected subsquare

$$(K_{111} \vee K_{121} \vee K_{131} \vee K_{211} \vee K_{221} \vee K_{231} \vee K_{311} \vee K_{321} \vee K_{331}) = 1$$

- number of logic equations in the system: $4 * 81 = 324$



Modeling Using a System of Logic Equations

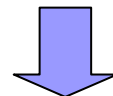
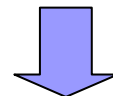
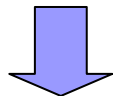
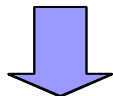
■ Simplification

- Due to the used conjunctions of restrictions $K_{i,j,k}$ each type of requirements solves the problem completely.
- Hence, the system of logic equations consists of **81 equations** (disjunctions of 9 conjunctions) completed by a **single** characteristic equation of the given setting.

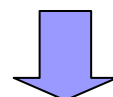
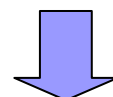
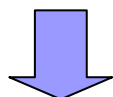
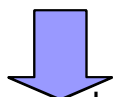
Basic Approaches of Parallel SAT-Solving

- Processing of variables in parallel
- SAT – example:

$$(a \vee \bar{b} \vee \bar{c}) \wedge (b \vee \bar{d} \vee \bar{e}) \wedge (\bar{a} \vee d \vee e) \wedge (b \vee c \vee \bar{e}) = 1$$



$$(a \vee \bar{b} \vee \bar{c}) = 1 \quad (b \vee \bar{d} \vee \bar{e}) = 1 \quad (\bar{a} \vee d \vee e) = 1 \quad (b \vee c \vee \bar{e}) = 1$$



a	b	c	d	e
1	-	-	-	-
0	0	-	-	-
0	1	0	-	-

a	b	c	d	e
-	1	-	-	-
-	0	-	0	-
-	0	-	1	0

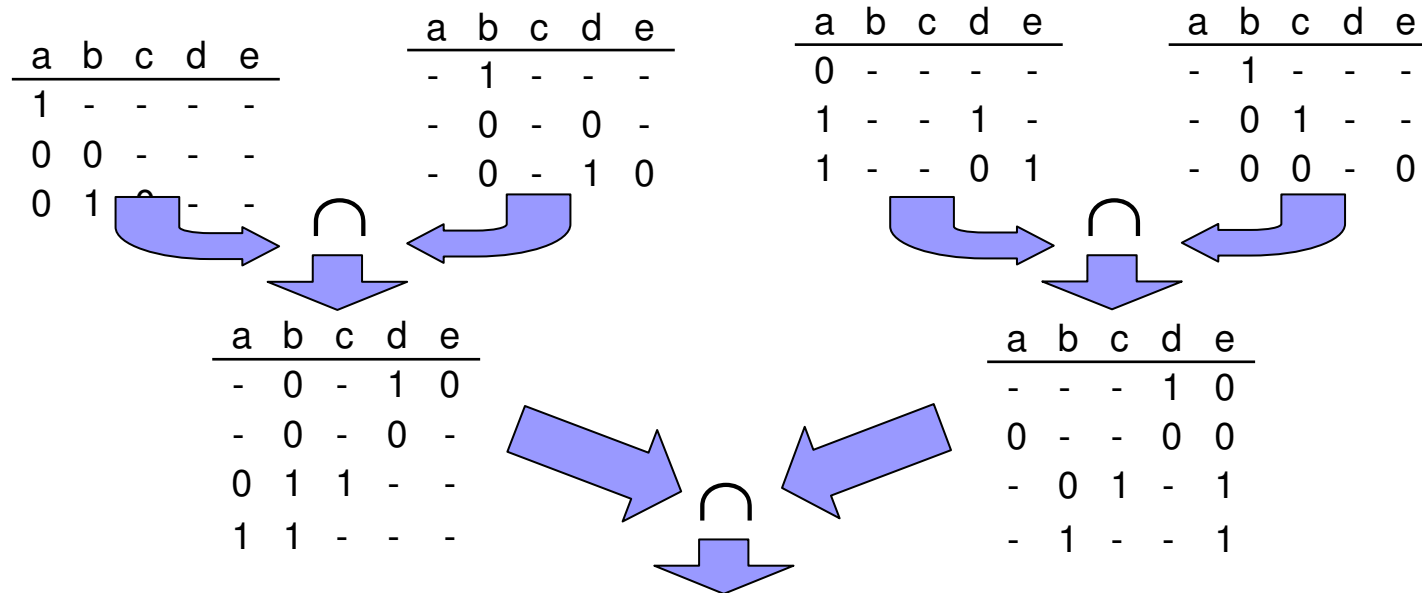
a	b	c	d	e
0	-	-	-	-
1	-	-	1	-
1	-	-	0	1

a	b	c	d	e
-	1	-	-	-
-	0	1	-	-
-	0	0	-	0

- Partial solutions expressed by orthogonal TVLs

Basic Approaches of Parallel SAT-Solving

- All global solutions calculated by intersections in parallel



- constructive solution process
- finds all solutions without any search



New Approach: Implicit 2-Phase SAT-Solver

■ Basic Idea

- the problem knowledge is widely distributed over the clauses of the SAT formula
- preserve this knowledge by
 - substituting the huge number of clauses by a much smaller amount of partial solution sets of logic equations
- requires the new two-phase SAT-solver
 - first phase: create partial solution sets
 - second phase: combine these partial solution sets to the final solution

New Approach: Implicit 2-Phase SAT-Solver

■ First Phase

- all literal in requirement clauses: not negated
- all literal in restrictive clauses: negated
- n^3 particular SAT- problems, e.g.

$$(\bar{x}_{1,1,1} \vee \bar{x}_{1,1,2})(\bar{x}_{1,1,1} \vee \bar{x}_{1,1,3})(\bar{x}_{1,1,1} \vee \bar{x}_{1,1,4})(\bar{x}_{1,1,1} \vee \bar{x}_{2,1,1})(\bar{x}_{1,1,1} \vee \bar{x}_{3,1,1})(\bar{x}_{1,1,1} \vee \bar{x}_{4,1,1}) \\ \wedge (\bar{x}_{1,1,1} \vee \bar{x}_{1,2,1})(\bar{x}_{1,1,1} \vee \bar{x}_{1,3,1})(\bar{x}_{1,1,1} \vee \bar{x}_{1,4,1})(\bar{x}_{1,1,1} \vee \bar{x}_{2,2,1}) \wedge x_{1,1,1} = 1.$$

- solution of a particular SAT- problem

$$x_{1,1,1} \bar{x}_{1,1,2} \bar{x}_{1,1,3} \bar{x}_{1,1,4} \bar{x}_{2,1,1} \bar{x}_{3,1,1} \bar{x}_{4,1,1} \bar{x}_{1,2,1} \bar{x}_{1,3,1} \bar{x}_{1,4,1} \bar{x}_{2,2,1} = 1$$

- n^3 such solutions stored as matrix of ternary vectors
- number of zeros: $q_{sfrc} = 3 * (n - 1) + (\sqrt{n} - 1)^2$

New Approach: Implicit 2-Phase SAT-Solver

■ First Phase

- solution of the first phase:
matrix of
partial
solution
sets
(*mpss*) or the
simple case
of a 4×4
Sudoku

The screenshot shows a software window titled "XBOOLE-Monitor 32-bit" with a menu bar (File, Objects, Derivatives, Metrics, Sets, Extras, View, ?) and a toolbar. The main area displays a large matrix of 0s and 1s, representing a solution set. The matrix has 64 rows and 64 columns, with the first 16 columns labeled 1-16 and the last 16 columns labeled 49-64. The matrix is mostly filled with 0s, with some 1s scattered throughout, indicating a partial solution. The interface also shows a "Protocol" window at the bottom with "4-fold View" and "1-fold View" options.

New Approach: Implicit 2-Phase SAT-Solver

■ First Phase

- restrictive problem laws are captured by an algorithm that creates the matrix *mpss* of partial solutions

Algorithm 1 Create the $n^3 \times n^3$ matrix of partial solution sets: *mpss* CPSS(*Size* *n*)

Require: number *n* of rows, columns or values of a Sudoku game
Ensure: $n^3 \times n^3$ matrix *pss* that includes for each possible local assignment the associated partial solution set

```
1: mpss ← ∅
2: for all i such that  $1 \leq i \leq n$  do {iterate over all rows}
3:   for all j such that  $1 \leq j \leq n$  do {iterate over all columns}
4:     for all k such that  $1 \leq k \leq n$  do {iterate over all values}
5:        $x[i, j, k] \leftarrow 1$ 
6:       for all v such that  $1 \leq v \leq n$  do {iterate over all values}
7:         if  $v \neq k$  then {other values}
8:            $x[i, j, v] \leftarrow 0$ 
9:         end if
10:      end for
11:    for all c such that  $1 \leq c \leq n$  do {iterate over all columns}
12:      if  $c \neq k$  then {other columns}
13:         $x[i, c, k] \leftarrow 0$ 
14:      end if
15:    end for
16:    for all r such that  $1 \leq r \leq n$  do {iterate over all rows}
17:      if  $r \neq i$  then {other rows}
18:         $x[r, j, k] \leftarrow 0$ 
19:      end if
20:    end for
21:     $ssn \leftarrow \sqrt{n}$ 
22:     $dssi \leftarrow ((i - 1) \bmod ssn) + 1$ 
23:     $bssi \leftarrow ((i - dssi) / ssn)$ 
24:     $dssj \leftarrow ((j - 1) \bmod ssn) + 1$ 
25:     $bssj \leftarrow ((j - dssj) / ssn)$ 
26:    for all r such that  $1 + bssi * ssn \leq r < (bssi + 1) * ssn$  do {rows of subsquare}
27:      for all c such that  $1 + bssj * ssn \leq c < (bssj + 1) * ssn$  do {columns of subsquare}
28:        if  $(r \neq i) \wedge (c \neq j)$  then {other rows and columns of the subsquare}
29:           $x[r, j, k] \leftarrow 0$ 
30:        end if
31:      end for
32:    end for
33:     $mpss[((i - 1) * n) + (j - 1) * n + k - 1] \leftarrow x$ 
34:  end for
35: end for
36: end for
37: return mpss{the  $n^3 \times n^3$  matrix of partial solution sets}
```

New Approach: Implicit 2-Phase SAT-Solver

■ Second Phase

- each requirement law can be extended by the partial solution sets of *mpss* known from the first phase; e.g.

- the clause

$$(x_{1,1,1} \vee x_{1,2,1} \vee x_{1,3,1} \vee x_{1,4,1}) = 1$$

- can be extended to

$$\begin{aligned}
 & x_{1,1,1} \bar{x}_{1,1,2} \bar{x}_{1,1,3} \bar{x}_{1,1,4} \bar{x}_{2,1,1} \bar{x}_{3,1,1} \bar{x}_{4,1,1} \bar{x}_{1,2,1} \bar{x}_{1,3,1} \bar{x}_{1,4,1} \bar{x}_{2,2,1} \\
 \vee & x_{1,2,1} \bar{x}_{1,2,2} \bar{x}_{1,2,3} \bar{x}_{1,2,4} \bar{x}_{2,2,1} \bar{x}_{3,2,1} \bar{x}_{4,2,1} \bar{x}_{1,1,1} \bar{x}_{1,3,1} \bar{x}_{1,4,1} \bar{x}_{2,1,1} \\
 \vee & x_{1,3,1} \bar{x}_{1,3,2} \bar{x}_{1,3,3} \bar{x}_{1,3,4} \bar{x}_{2,3,1} \bar{x}_{3,3,1} \bar{x}_{4,3,1} \bar{x}_{1,1,1} \bar{x}_{1,2,1} \bar{x}_{1,4,1} \bar{x}_{2,4,1} \\
 \vee & x_{1,4,1} \bar{x}_{1,4,2} \bar{x}_{1,4,3} \bar{x}_{1,4,4} \bar{x}_{2,4,1} \bar{x}_{3,4,1} \bar{x}_{4,4,1} \bar{x}_{1,1,1} \bar{x}_{1,2,1} \bar{x}_{1,3,1} \bar{x}_{2,3,1} = 1
 \end{aligned}$$

- due to the repeated use of the same literal in different clauses, the associated partial solution set of *mpss* is used several times



New Approach: Implicit 2-Phase SAT-Solver

■ Second Phase

- the global solution is calculated straight forward controlled by the requirement laws of one type only
- applying the partial solution sets of the given settings first restricts the remaining solution space
- no search procedures are involved
- due to the fixed values in the restrictions, it is a disadvantage to take the requirement type of values in a selected field in the second phase

New Approach: Implicit 2-Phase SAT-Solver

■ Second Phase

- required problem laws are captured by an algorithm that uses the matrix *mpss* and calculates the final solutions
- main operations
 - **union** of selected partial solution sets
 - **intersection** of these sets

Algorithm 2 Solve SAT Problem of an $n \times n$ Sudoku: $s \text{ SSP}(\text{Size } n, \text{Settings } g, \text{MPS } mpss)$

Require: number n of rows, columns or values of a Sudoku game

vector g of given settings where the value g specified the associated value in $mpss$

matrix $mpss$ of partial solution sets as created in Algorithm 1

Ensure: set of all solution vectors of the length n^3 that hold both the given settings and the rule of the Sudoku game

```
1:  $s \leftarrow FULL(mpss)$ 
2: for all  $g_i$  in  $g$  do {iterate over all given settings}
3:    $s \leftarrow ISC(s, mpss[g_i])$ 
4: end for
5: for all  $k$  such that  $1 \leq k \leq n$  do {iterate over all values}
6:   for all  $i$  such that  $1 \leq i \leq n$  do {iterate over all rows}
7:      $rule \leftarrow \emptyset$ 
8:     for all  $j$  such that  $1 \leq j \leq n$  do {iterate over all columns}
9:        $constraint \leftarrow mpss[(((i-1)*n) + (j-1)*n) + k - 1]$ 
10:       $rule \leftarrow UNI(rule, constraint)$ 
11:    end for
12:     $s \leftarrow ISC(s, rule)$ 
13:  end for
14: end for
15: return  $s$  {set of all solutions of the Sudoku game}
```

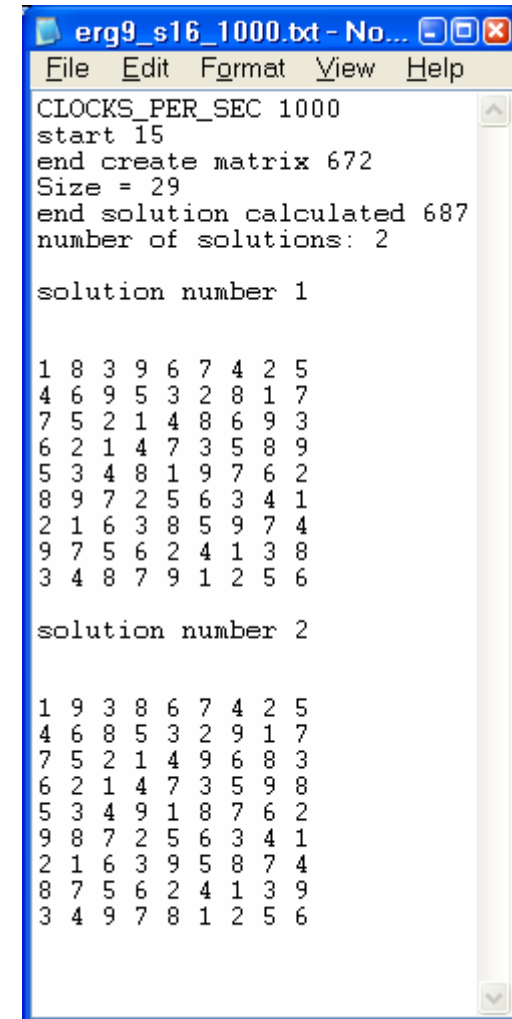
Experimental Results

■ 9 × 9 Sudoku

- 16 of 81 fixed values only
- 729 Boolean variable
- first phase: 667 msec
- second phase: 15 msec
- 2 solutions

■ 16 × 16 Sudoku

- 120 of 256 fixed values
- 4096** Boolean variables
- first phase: 0 msec
- second phase: 148,4 sec
- 1 solution



```
erg9_s16_1000.txt - No...
File Edit Format View Help
CLOCKS_PER_SEC 1000
start 15
end create matrix 672
Size = 29
end solution calculated 687
number of solutions: 2

solution number 1

1 8 3 9 6 7 4 2 5
4 6 9 5 3 2 8 1 7
7 5 2 1 4 8 6 9 3
6 2 1 4 7 3 5 8 9
5 3 4 8 1 9 7 6 2
8 9 7 2 5 6 3 4 1
2 1 6 3 8 5 9 7 4
9 7 5 6 2 4 1 3 8
3 4 8 7 9 1 2 5 6

solution number 2

1 9 3 8 6 7 4 2 5
4 6 8 5 3 2 9 1 7
7 5 2 1 4 9 6 8 3
6 2 1 4 7 3 5 9 8
5 3 4 9 1 8 7 6 2
9 8 7 2 5 6 3 4 1
2 1 6 3 9 5 8 7 4
8 7 5 6 2 4 1 3 9
3 4 9 7 8 1 2 5 6
```



Conclusions

- ❑ Many combinatorial problems can be transformed into satisfiability problems.
- ❑ advantages of SAT
 - ❑ unique representation of different problems
 - ❑ universal well explored SAT – solver
- ❑ disadvantage of SAT
 - ❑ large number of clauses
 - ❑ loss of problem knowledge
- ❑ our **new implicit two-phase SAT – solver** breaks this disadvantage
 - first phase: generate matrix of partial solution sets *mpss* based on the restrictive laws
 - second phase: solve the SAT – problem controlled by the required laws using the matrix *mpss*
- ❑ Several problems having up to more than 4000 Boolean variables were solved in a couple of seconds.