

### 1.Assignment: Finding Hard Instances of the $\operatorname{SUDOKU}$ Puzzle

- ► Introducing the Assignment Motivation: Solving Sudoku puzzles
- Background Information
- Theoretical background: "What are hard SAT problems?"
- Practical issues: "How to represent a Sudoku as a PL SAT problem."
- Technical Information
- Collections & Formats
- SAT solvers.
- Organizational Matters



# Finding Hard Instances of the SUDOKU Problem

- ightharpoonup Sudoku is a puzzle game, where you have to fill a  $9 \times 9$  grid with numbers from 1 to 9. There are also sub-grids of size 3, sometimes called regions
- ► The puzzle has to be filled so that no number occurs more than once in any row, column or region.
- Sukoku puzzled can be solved by SAT solving.
- translate puzzle into PL formula (ideally in CNF)
- solve SAT problem for this formula
- using an existing SAT solver.
- Every model corresponds to a solution of the puzzle.



### Research question:

### Is there a relation between types of puzzles and the computational hardness of the related SAT problem?

- ► There are different types of puzzles, e.g. easy, difficult and fiendish one's.
- Some have many givens, some fewer.
- Proper SUDOKUs have unique solutions. Is there a difference in runtime between proper and improper SUDOKUs?

The task of the assignment is to develop and empirically test such theories of computational hardness of SUDOKU puzzles.

### Let's play the game

		_			_			_
	2	4				П	7	
		8	4		_	6		
2				2				9
	4		2		3		$\infty$	
$\infty$								1
	2		1		$\infty$		2	
2				8				3
		2	9		7	2		
	2	3				4	9	



### 1.Assignment: Finding Hard Instances of the $\operatorname{SUDOKU}$ Puzzle

- ► Introducing the Assignment Motivation: Solving Sudoku puzzles
- Background Information
- Theoretical background: "What are hard SAT problems?"
- Practical issues: "How to represent a Sudoku as a PL SAT problem."
- Technical Information
- Collections & Formats
- SAT solvers.
- Organizational Matters



# How to represent a SUDOKU as a SAT problem.

- 1. Each cell has to be filled with exactly one number.
- 2. In each row all numbers from 1 to 9 have to occur (this is equivalent to the fact that each number can only occur once per row)
- 3. In each column all numbers from 1 to 9 have to occur (this is equivalent to the fact that each number can only occur once per column)
- 4. In each  $3\times3$ -square all numbers from 1 to 9 have to occur (this is equivalent to the fact that each number can only occur once per  $3\times3$ -square)
- 5. Game specific constraints



## A possible Propositional Encoding

Choose variable names: A11 to 199

 $|91 \rightarrow (\neg |92 \land \neg |93 \land \dots \land \neg |99) \land |92 \rightarrow (\neg |93 \land \dots \land \neg |99) \land$ (108 → 109) we also need a constraint to specify that one of the values must be true, i.e.

A11  $\vee$  A12  $\vee$  . . .  $\vee$  A19, and the same for all cells.

2. A11→(¬A21 ∧ ¬A31 ∧ ... ∧ ¬A91) ∧ A21→(¬A31 ∧ ... ∧ ¬A91) ∧ ... ∧ (A81

**Exercise:** Encode on paper a 3 imes 3 Sudoku.



### 1.Assignment: Finding Hard Instances of the $\operatorname{SUDOKU}$ Puzzle

- ► Introducing the Assignment Motivation: Solving Sudoku puzzles
- Background Information
- Theoretical background: "What are hard SAT problems?"
- Practical issues: "How to represent a Sudoku as a PL SAT problem."
- ► Technical Information
- Collections & Formats
- SAT solvers.
- ► Organizational Matters



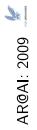
#### Collections

- ► There are numerous websites for SUDOKU with examples.
- ► Mostly, in graphical format, but also some txt outputs.
- Research page: http://www.research.att.com/~gsf/ with huge test-sets (already computationally biased).
- google!



## SAT solving: DIMACS and ZChaff

- ► There have been SAT solving competitions for years.
- See at http://www.satcompetition.org/ for details
- ▶ Input format is DIMACS (see blackboard)
- ▶ One of the contributors is ZChaff, which I've applied with success. But you are not restricted to using ZChaff (as long as you explain us in a README how to use the other prover).



#### Using ZCHAFF

- $lacktrianglerapprox = (P ee \lnot Q) \& (Q ee R) \& (\lnot R ee \lnot P)$
- ► Now this is in CNF
- P=1
- Q=2
- R=3
- C1: 1
- C2: 2 C3: -1
- ► DIMACS: representation
- c a diplomatic problem
- c created: 06/01/03
- p cnf 3 3 (#variable #clauses)

- AR@AI: 2009 🥌

- 2 3 0 -1 -3 0
- ► Solution: Instance satisfiable: -1 -2 3