

Planning Sudoku für SAT

M. Fareed Arif

s9065145@inf.tu-dresden.de
Center for Computational Logic (ICCL)
Technische Universität Dresden
Tel.: +123-45-678910, Fax: +123-45-678910

Abstract

Sudoku is a very simple and well-known puzzle. Many encoding schemes such as: Minimal, Efficient, Extended and Pre-Processed are known for this puzzle. Any Sudoku puzzle is formulated as a CNF formula ϕ which is satisfiable iff the Sudoku has a solution. We restrict our attention to consider only the instance of Sudoku which has only one solution and can only be solved using reasoning (i.e. with no search). Pre-Processed encoding suffices for this characterization, where other encoding schemes add redundant clauses to minimal satisfying formulae; We observe that updating Pre-Processed encoding regarding pre-assigned cells configurations facilitate in higher degree of pre-processing and unit propagation can reduce the number of final clauses generated to be checked for a model.

In recent years, there has been a considerable renewed interest in the satisfiability problem (SAT) and many efficient SAT solvers are available. Traditionally planning has been formalized as deduction (0), but we'd formulated our model on satisfiability rather than on deduction. This approach not only provides a more flexible framework for stating different kinds of constraints on plans, but also more accurately reflects the theory behind modern constraint-based planning systems (0).

Preliminaries

Sudoku

Standard Sudoku puzzle usually appear in Newspapers is a 9×9 grid instance of a very easy problem;

Definition 0.1 *A Sudoku puzzle is represented by a $\mathbb{N} \times \mathbb{N}$ grid, which comprises of an $\sqrt{\mathbb{N}} \times \sqrt{\mathbb{N}}$ sub-grids (also called boxes). Some of the entries in the grid are filled with numbers from 1 to \mathbb{N} , whereas other entries are left blank (0);*

Puzzles are often assigned a difficulty level, which usually depends on the number of initial non-blank entries provided. This number may be as few as 17 to test expert players for 9×9 instance (0). But it could also be shown that with an automated planning system it can be reduced further to only one assignment and the planner still able to solve the problem correctly.

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Definition 0.2 *A Sudoku puzzle is solved by assigning numbers from 1 to \mathbb{N} (size of puzzle) to the blank entries such that every row, every column, and every $\sqrt{\mathbb{N}} \times \sqrt{\mathbb{N}}$ block (sub-grid) contains each of the any \mathbb{N} possible numbers;*

Propositional logic formalism is used to encode Sudoku puzzle in our encoding and other well known encoding schemes (0) (0) (0).

Planning as Satisfiability

The complementary problem to deducibility is satisfiability that is finding a model for set of axioms. The formalism of planning as satisfiability turns out to have a number of attractive properties like (0) (0):

- It is easy to state arbitrary facts about any state of the world not just the initial and goal states;
- It is likewise easy to state arbitrary constraints on the plan e.g. specifying an action performed at a any specific time interval;
- Finally the approach provides a more accurate formal model of modern constraint based planners.

Sudoku Axiomatization

Many Sudoku solvers are already available (0) (0) and since there are more than 6.10^{21} possible grids for a simple looking 9×9 puzzle instance (0); hence, a naive backtracking algorithm would be infeasible and the only feasible approach in this scenario is to combine backtracking with some form of constraint propagation. But we utilize a SAT-based approach in which a puzzle is translated into propositional formula Φ then we use any standard SAT solver to obtain the model (satisfying assignment) of Φ ; this model can then be readily translated into the solution of puzzle.

Sudoku Encoding Since puzzles are regarded as a propositional SAT problem, we'll provide an overview of various existing encoding schemes. A SAT problem is represented as a propositional formula Φ where each variable P_i is assigned 0 (\mathbb{F}) or 1 (\mathbb{T}) where $i \in (1, \dots, n)$.

In an $\mathbb{N} \times \mathbb{N}$ puzzle each cell can contain a number from 1 to n . Each tuple (r, c, v) denotes a variable which is true iff the cell in row r and column c is assigned a number v ; $[r, c] = v$. The resulting set of formulas turn out to be $V = \{(r, c, v) \mid 1 \leq r, c, v \leq n\}$. Exactly one number for each

cell, row, column and block can be viewed as a *definedness* and *Uniqueness constraints*. Following is the formulae:

- There is at exactly one number in each cell
 $\phi_{cell.ex} := \phi_{cell.def} \wedge \phi_{cell.uniq}$
 There is at least one number for each cell
 $\phi_{cell.def} := \bigwedge_{r=1}^n \bigwedge_{c=1}^n \bigvee_{v=1}^n (r, c, v)$
 Each number appears at most one in each cell
 $\phi_{cell.uniq} := \bigwedge_{r=1}^n \bigwedge_{c=1}^n \bigwedge_{v_i=1}^n \bigwedge_{v_j=v_i+1}^n \neg(r, c, v_i) \vee \neg(r, c, v_j)$
- There is at exactly one number in each row
 $\phi_{row.ex} := \phi_{row.def} \wedge \phi_{row.uniq}$
 There is at least one number for each row
 $\phi_{row.def} := \bigwedge_{r=1}^n \bigwedge_{v=1}^n \bigvee_{c=1}^n (r, c, v)$
 Each number appears at most one in each row
 $\phi_{row.uniq} := \bigwedge_{r=1}^n \bigwedge_{v=1}^n \bigwedge_{c_i=1}^{(n-1)} \bigwedge_{c_j=c_i+1}^n \neg(r, c_i, v) \vee \neg(r, c_j, v)$
- There is at exactly one number in each column
 $\phi_{col.ex} := \phi_{col.def} \wedge \phi_{col.uniq}$
 There is at least one number for each column
 $\phi_{col.def} := \bigwedge_{c=1}^n \bigwedge_{v=1}^n \bigvee_{r=1}^n (r, c, v)$
 Each number appears at most one in each column
 $\phi_{col.uniq} := \bigwedge_{c=1}^n \bigwedge_{v=1}^n \bigwedge_{r_i=1}^{(n-1)} \bigwedge_{r_j=r_i+1}^n \neg(r_i, c, v) \vee \neg(r_j, c, v)$
- There is at exactly one number in each block
 $\phi_{blk.ex} = \phi_{blk.def} \wedge \phi_{blk.uniq}$
 There is at least one number for each block
 $\phi_{blk.def} = \bigwedge_{rof=1}^m \bigwedge_{cof=1}^m \bigwedge_{v=1}^n \bigvee_{r=1}^m \bigvee_{c=1}^m (rof * m + r, cof * m + c, v)$ where $m := \sqrt{n}$;
 Each number appears at most one in each block
 $\phi_{blk.uniq} = \bigwedge_{rof=1}^m \bigwedge_{cof=1}^m \bigwedge_{v=1}^n \bigwedge_{r=1}^n \bigwedge_{c=r+1}^n \neg(rof * m + (r \bmod m), cof * m + (r \bmod m), v) \vee \neg(rof * m + (c \bmod m), cof * m + (c \bmod m), v)$ where $m := \sqrt{n}$;

The fixed cells are naturally represented as unit clauses in encoding of puzzle:

$V^+ = \{(r, c, v) \in V \mid [r, c] \text{ is a fixed cell and has a pre-assigned value } v\}$.

$\phi_{assign} = \bigwedge_{i=1}^k (r, c, v)$ where $(r, c, v) \in V^+$

Now encoding is nothing more than the following set of clauses:

- $\Phi_{extended} := \phi_{cell.ex} \wedge \phi_{row.ex} \wedge \phi_{col.ex} \wedge \phi_{blk.ex} \wedge \phi_{assign}$.
- $\Phi_{efficient} := \phi_{cell.ex} \wedge \phi_{row.uniq} \wedge \phi_{col.uniq} \wedge \phi_{blk.uniq} \wedge \phi_{assign}$.
- $\Phi_{minimal} := \phi_{cell.def} \wedge \phi_{row.uniq} \wedge \phi_{col.uniq} \wedge \phi_{blk.uniq} \wedge \phi_{assign}$.

$V^- = \{(r, c, v) \in V \mid \exists (r', c', v') \in V^+ ((r = r') \wedge (c = c') \wedge (v \neq v')) \vee ((r = r') \wedge (c \neq c') \wedge (v = v')) \vee ((r \neq r') \wedge (c = c') \wedge (v = v')) \vee ((r \neq r') \wedge (c \neq c') \wedge (v = v'))\} \wedge \exists lRow, lCol (lRow \leq r, r' \leq lRow + m) \wedge (lCol \leq c, c' \leq lCol + m)\}$

where:

$$lRow = \begin{cases} 1 & \text{true;} \\ ((r-1)/m) * (m+1) & \text{false;} \end{cases}$$

$$lCol = \begin{cases} 1 & \text{true;} \\ ((c-1)/m) * (m+1) & \text{false;} \end{cases}$$

$$m = \sqrt{n}.$$

Reduction operators described in pre-assigned formulae is as follows:

$$U := \{V^+, V^-\} \phi \Downarrow U = \{C \in \phi \mid \neg \exists L \in C. (L = x \wedge x \in U)\}$$

$$\phi \Downarrow V^+ = \{\forall C \in \phi. \forall L \in C \mid \neg (L = \neg x \Rightarrow x \in V^+)\}$$

$$\phi \Downarrow V^- = \{\forall C \in \phi. \forall L \in C \mid \neg (L = x \Rightarrow x \in V^-)\}$$

Githown (0) encoding is formulated as:

- $\Phi_{pre.process} := (\phi_{cell.uniq} \cup \phi_{row.uniq} \cup \phi_{col.uniq} \cup \phi_{blk.uniq}) \Downarrow V^- \cup (\phi_{cell.def} \cup \phi_{row.def} \cup \phi_{col.def} \cup \phi_{blk.def}) \Downarrow V^- \cup \phi_{assign} \Downarrow V^+.$

Lemma 0.3 A satisfiability equivalence relation holds between $\Phi_{extended}$ and $\Phi_{pre.process}$ with respect to a satisfying assignment α , i.e. α satisfies $\Phi_{extended}$ iff α satisfies $\Phi_{pre.process}$.

Proposed Encoding

The phenomenon we'd used to further reduce redundancies is explained using an example for 4*4 puzzle instance. Suppose, we have some pre-assigned cells $[1, 1] = 4$, $[1, 2] = 3$, $[1, 3] = 2$:

- Literals in row *definedness clause* are $(1, 1, 1)$, $(1, 2, 1)$, $(1, 3, 1)$, $(1, 4, 1)$;
- Literals $(1, 1, 1)$ is falsified by pre-assigned $[1, 1] = 4$ because of same cell constraint; likewise $(1, 2, 1)$, $(1, 3, 1)$ are falsified because of same row constraint.
- $(1, 4, 1)$ is the only literal left where definedness constraint imposed it to be true thus assigning $[1, 4] = 4$.

The fact which we want to reveal from the above example is that by imposing all the above constraints consequence to obtaining new assigned values for the cells which are blank before and are not even in ϕ_{assign} . So, rendering this process to include these newly assigned cells within our ϕ_{assign} set and perform the deletion and removal of clauses level and literals up to saturation result in further reductions and the following formulae confers it's detail as followed:

$$\varphi_{new} := (\phi_{cell.def} \cup \phi_{row.def} \cup \phi_{col.def} \cup \phi_{blk.def}) \Downarrow V^-$$

$$V^* := \{\forall C \in \varphi_{new}. |C| = 1 \wedge C \notin \phi_{assign} \Rightarrow C \cup V^+\}$$

$$\Phi_{new} := \varphi_{new} \Downarrow V^- \cup (\phi_{cell.uniq} \cup \phi_{row.uniq} \cup \phi_{col.uniq} \cup \phi_{blk.uniq}) \Downarrow V^-$$

Lemma 0.4 A satisfiability equivalence relation holds between $\Phi_{pre.process}$ and Φ_{new} with respect to a satisfying assignment α , i.e. α satisfies $\Phi_{pre.process}$ iff α satisfies Φ_{new} .

Conclusion

Our proposed encoding not only reduces a large amount of redundant clauses but also this process of redundant clause elimination coincides with the logic of finding newly assigned values for the blank cells. Thus not only eliminating a larger set of clauses for the solver but also filling in the blank cells in a correct fashion.

References

- David Champan. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.
- Bertram Felgenhauer and Frazer Jarvis. Enumerating possible sudoku grids. 2005. <http://www.shef.ac.uk/~pm1afj/sudoku/>.
- Himanshu Jain Gihwon Kwon. Optimized cnf encoding for sudoku puzzles. 2006. 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006).
- McCarthy John and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Meltzer and Michie, Machine Intelligence*, pages 463–502, 1969.
- DeadMan’s Handle Ltd. Sudoku solver. 2005. <http://www.sudoku-solver.com/>.
- I. Lynce and J. Ouaknine. Sudoku as a sat problem. 2006. 9th International Symposium on Artificial Intelligence and Mathematics.
- Henry Kautz Bart Selman and Joerg Hoffmann. Satplan: Planning as satisfiability. 2006. 5th International Planning Competition.
- Mark Stefik. Planning with constraints. *Artificial Intelligence*, 16:111–140, 1981.
- Pete Wake. Sudoku solver by logic. 2005. <http://www.sudokusolver.co.uk>.
- Tjark Weber. A sat-based sudoku solver. 2005. 12th International Conference on Logic for Programming Artificial Intelligence and Reasoning(LPAR 2005).