Knowledge Presentation - Project 1: Sudoku Encodings

Pasquale Muscettola, Mathijs Mul

September 22, 2016

Hypothesis

Logically, the constraints governing the Sudoku game can be considered as a set of axioms, which allow for the inference from an initial assignment of given numbers to a completely filled out grid through a number of intermediate reasoning steps. Each configuration of numbers that follows from the given assignment in accordance with the rules of the game can thus be seen as a 'theorem' of the particular Sudoku. Continuing this logic is eventually supposed to culminate in a correct solution to the puzzle. Taking this logical approach on the traditional $n = 3$ Sudoku format, we will investigate different possible encodings of the game constraints. After identifying the minimal set of constraints that are necessary and sufficient in characterising the game, we will identify several extensions and variations. Using a SAT solver we will then study how the different and extended encodings influence the computational effort that is needed to solve puzzles that have been indicated to be of the same (human) difficulty rate on an online database.

Thus, what we will do is look at alternative Sudoku encodings that are logically implied by the minimal set of constraints, and that could thus be considered redundant from a purely syntactic point of view. In particular, we wish to look at techniques that human solvers use to address Sudoku puzzles, formalize such strategies and add them to the encoding to study how this influences the required computational effort. Logically, such strategies could also be considered superfluous, as they follow from the general axioms and thus provide no new information. An interest in the difference between the computational and the logical impact of adding such redundancies is the main motivation for this research.

We hypothesize that **adding redundant clauses to the encoding of the Sudoku game will speed up the computation that is carried out by a SAT engine to find a solution**. Even though such propositions are implied by the minimal set of constraints characterising the game, so that they are logically redundant, we expect that including them in the propositional representation of the game will lead to computational speed-up. Our main reason for believing this is that adding constraints decreases the search space for a SAT solver, and should thus speed up the computation.

Experimental setup

In order to test our hypothesis we used a large database of Sudokus from the website www.thonky.com, which are rated according to their difficulty as perceived by human solvers. We used the SAT solver zChaff to solve the puzzles, which comprises a deterministic algorithm that has achieved very well in SAT competitions and that provides clear statistical information with its output.

The notion of computational effort at stake in this project is identified as the number of decisions that have to be made by zChaff in order to complete a Sudoku puzzle. This is our metric of computational 'hardness'. We chose not to base our conclusions on runtime, mainly because solving a Sudoku can be done very quickly by zChaff, so that the required time is in a range that is too sensitive to irrelevant background processes in the computer to be an accurate indication of underlying computational effort.

Before running any tests, we had to determine the different kinds of encodings of Sudoku puzzles that we wished to consider. In this, we decided to focus on three cases: one where we only work with the minimal set of constraints that are necessary and sufficient for characterising

the game, a second one that uses a trivial extension of this minimal encoding, and a third one that incorporates the propositionalization of a human solving strategy.

First, the minimal propositional encoding of the Sudoku puzzle has to be determined. This only requires the basic constraints, which on their own are enough to define the game, namely:

- D_{cell}: **Definedness of cells**: all cells in the grid should contain at least one value from the set $\{i|1 \leq i \leq n^2\}$.
- U_{row}: **Uniqueness in rows**: in each row, all values from the set $\{i|1 \leq i \leq n^2\}$ should feature at most once.
- U_{column}: **Uniqueness in columns**: idem for columns.
- U_{block}: **Uniqueness in blocks**: idem for blocks.

In order to give an adequate representation in propositional logic, we define $(n^2)^3$ variables of the form (r, c, v), where r denotes row number, c denotes column number and v denotes number value of the cell whose location in the grid is specified by r and c. With these variables we can formulate the following propositional encoding of the above-mentioned constraints:

- D_{cell}: $\bigwedge_{r=1}^{n^2} \bigwedge_{c=1}^{n^2} \bigvee_{v=1}^{n^2} (r, c, v)$
- U_{row}: $\bigwedge_{r=1}^{n^2} \bigwedge_{v=1}^{n^2} \bigwedge_{c_i=n}^{n^2-1} \bigwedge_{c_j=c_i+1}^{n^2} \neg(r, c_i, v) \vee \neg(r, c_j, v)$
- U_{column}: $\bigwedge_{c=1}^{n^2} \bigwedge_{v=1}^{n^2} \bigwedge_{r_i=n}^{n^2-1} \bigwedge_{r_j=r_i+1}^{n^2} \neg(r_i, c, v) \vee \neg(r_j, c, v)$
- U_{block}: $\bigwedge_{r_{\text{block}}=1}^{n} \bigwedge_{c_{\text{block}}=1}^{n} \bigwedge_{v=1}^{n^2} \bigwedge_{r=1}^{n^2} \bigwedge_{c=r+1}^{n^2} \neg(r_{\text{block}}n + r \mod n, c_{\text{block}}n + r \mod n, v) \vee \neg(r_{\text{block}}n + c \mod n, c_{\text{block}}n + c \mod n, v)$

We let $E_{\text{minimal}} = \{D_{\text{cell}}, U_{\text{row}}, U_{\text{column}}, U_{\text{block}}\}$ denote the minimal encoding. From this minimal set of constraints, we can derive some other constraints, which will be added as redundancies. The most elementary redundancies to consider are the following:

- U_{cell}: **Uniqueness of cells**: all cells in the grid should contain at most one value from the set $\{i|1 \leq i \leq n^2\}$.
- D_{row}: **Definedness in rows**: in each row, all values from the set $\{i|1 \leq i \leq n^2\}$ should feature at least once.
- D_{column}: **Definedness in columns**: idem for columns.
- D_{block}: **Definedness in blocks**: idem for blocks.

Propositional encoding is as follows:

- U_{cell}: $\bigwedge_{r=1}^{n^2} \bigwedge_{c=1}^{n^2} \bigwedge_{v_i=n}^{n^2-1} \bigwedge_{v_j=v_i+1}^{n^2} \neg(r, c, v_i) \vee \neg(r, c, v_j)$
- D_{row}: $\bigwedge_{r=1}^{n^2} \bigwedge_{v=1}^{n^2} \bigvee_{c=1}^{n^2} (r, c, v)$
- D_{column}: $\bigwedge_{c=1}^{n^2} \bigwedge_{v=1}^{n^2} \bigvee_{r=1}^{n^2} (r, c, v)$
- D_{block}: $\bigwedge_{r_{\text{block}}=1}^{n} \bigwedge_{c_{\text{block}}=1}^{n} \bigwedge_{v=1}^{n^2} \bigvee_{r=1}^{n} \bigvee_{c=1}^{n} (r_{\text{block}}n + r, c_{\text{block}}n + c, v)$

We let $E_{\text{extended}} = E_{\text{minimal}} \cup \{U_{\text{cell}}, D_{\text{row}}, D_{\text{column}}, D_{\text{block}}\}$ denote the extended encoding comprising the minimal constraints in addition to the redundancies listed above.

In addition to the extension thus defined, we are interested in human Sudoku solving techniques, i.e. so-called 'pencil and paper' algorithms that are usually applied intuitively and informally. Such strategies could also be seen as logical consequences of E_{minimal}. Strictly speaking,

they add no new information or constraints, but by efficient filtering of the infinitely many implications of the basic Sudoku axioms, they do make solving the puzzles easier. We want to encode such a strategy, add it to the propositional Sudoku representation and study how this impacts the computational effort needed by a SAT enging to find solutions. According to our hypothesis, extending the encoding with such information should make computation easier, not only for a human solver, but also for a SAT solver.

In particular, we decided to look at the so-called 'Naked Twins' strategy, as it is a basic and well-known human Sudoku solving method. The strategy makes use of an exclusion rule that can be described and formalised as follows:

- R_{NT}: **Naked Twins**: if two cells in a unit are such that together they must contain two specific numbers, but the order is unknown, then no other cells in this unit should contain either of of these numbers. If we consider the row units, then the propositional formalisation is as follows:
  $\bigwedge_{r=1}^{n^2} \bigwedge_{c_1=1}^{n^2-1} \bigwedge_{c_2=c_1+1}^{n^2} \bigwedge_{v_1=1}^{n^2-1} \bigwedge_{v_2=v_1+1}^{n^2} \bigwedge_{c_3 \neq c_1,c_2}$
  $\Big(\bigvee_{v_3 \neq v_1,v_2} ((r,c_1,v_3) \vee (r,c_2,v_3)) \vee \neg(r,c_3,v_1)\Big) \wedge \Big(\bigvee_{v_3 \neq v_1,v_2} ((r,c_1,v_3) \vee (r,c_2,v_3)) \vee \neg(r,c_3,v_2)\Big)$

  Also did row completion, out of memory.
  for each: show computational statistics

Experimental results

describe your experimental findings, discuss if they are reliable etc.

Interpretation

what do your experimental results say about your hypothesis, do they confirm or falsify your hypothesis, do they show any further interesting lessons

Conclusion

summary, future work: summarise your work and conclusions, and suggest new tasks or questions that follow from your work.

Results

Conclusion

future work: - consider larger sudokus/variations - encode more strategies - compare different sat solvers

NOTES

basic rules: axioms, techniques as logical consequences, logically redundant but may ease computation look at possible variations in basic axioms of encodings (human techniques, jellyfish, xwings, naked singles) look at different difficulty degrees possibly consider improper sudokus too?

Goal: investigate different CNF representations of the Sudoku game constraints and study how they influence the computational hardness of SAT solving games (measured in flips)

There is a minimal CNF representation of the Sudoku game including the following clauses: - existence of cell values - uniqueness in rows - uniqueness in columns - uniqueness in blocks - representation of given numbers

However, in manually solving a Sudoku puzzle, humans do not exclusively rely on these bare rules, but use them to infer 'theorems' and strategies that make solving easier. As a seemingly

trivial example: cell values must not only exist, but also be unique, and existence constraints also apply to rows/columns/blocks.

Specifically: * consider minor extensions of minimal representation * consider particular methods employed by Sudoku players, represent them in CNF, add them to the Sudoku representation and investigate impact on computational effort.

(2) Method, metric and dataset

We are considering computational effort, and will be using the required number of flips as a measure. (Required running time should follow form these data in a device-specific way.)

(3) Dataset

or http://www.menneske.no/sudoku/ and extract from html, so better use the other one.

(4) Solver

We will use the zChaff SAT solver, because it appears to be the best one that is currently available.

Important reference: http://www.cs.cmu.edu/ hjain/papers/sudoku-as-SAT.pdf