Sudoku Puzzle Theory

Spencer T. Parkin

December 8, 2013

1 Introduction

While finding solutions to Sudoku puzzles gives many solvers a sense of accomplishment, for some such as myself, there is no real accomplishment in solving any one Sudoku puzzle, no matter how hard it may be. Instead, such a sense can only be attained by producing a general theory of the puzzle in which a method for solving any Sudoku puzzle naturally unfolds. Having such a method, and being able to implement it as a computer algorithm, one can put all Sudoku puzzles to rest.

Using the ideas of Peter Norvig in his essay [], this paper is an attempt to develop such a theory, and to apply it in the problem of solving the general Sudoku puzzle.

In the discussion to follow, we assume a knowledge of the general Sudoku puzzle, and so procede with no introduction to it.

2 Definitions and Lemmas

A given instance of the Sudoku puzzle, in this paper, will be represented by a valid set of decisions. A decision d is a decision to put a specific number into a specific location of a Sudoku square. A valid set of decisions D has the property that for any two decisions $d_0, d_1 \in D$, the square associated with d_0 is not that of d_1, and that if all decisions in D were applied to an empty Sudoku square, it would produce a valid, partially or fully completed Sudoku square. Unless otherwise stated, all decisions sets in this paper are valid.

A valid set of decisions D_0 is said to be completable, if there exists a valid set of decisions D_1 such that $D_0 \cup D_1$ is a valid set of decisions that, if

applied to the empty Sudoku square, would product a fully complete Sudoku square. To be short, we say that the set of decisions $D_0 \cup D_1$ is complete. If for a given completable set of decisions D_0, there exists one and only one set of decisions D_1 such that $D_0 \cup D_1$ is complete, then we say that D_0 is uniquely completable.

Given any valid set of decisions D, we let $f(D)$ denote the set of all decisions d such that $D \cup \{d\}$ is a valid set of decisions. Notice that if $f(D)$ is the empty set, then D is either complete or not completable. Note also, however, that non-completability of D does not imply the emptiness of $f(D)$.

For any given valid set of decisions D, if $d \in f(D)$ is a decision such that $D \cup \{d\}$ is not completable, we call d a bad decision in the context of D, and a good decision in the context of D otherwise.

Lemma 2.1. *Let D_0 and D_1 each be a completable set of decisions with $D_0 \subseteq D_1$. Then, if $d \in f(D_0)$ is a bad decision in the context of D_0 and $d \in f(D_1)$, d is a bad decision in the context of D_1.*

Proof. Since d is a bad decision in the context of D_0, we know that there does not exist a set of decisions D' such that $D_0 \cup \{d\} \cup D'$ is complete. Keeping this in mind, suppose, to the contrary of the lemma, that there exists a set of decisions D such that $D_1 \cup \{d\} \cup D$ is complete. But if we let $D' = (D_1 - D_0) \cup D$, we then see that $D_0 \cup \{d\} \cup D'$ is complete, which contradicts the earlier establishment of the non-existance of such a set D'. □

Lemma 2.2. *Let D_0 and D_1 each be a uniquely completable set of decisions with $D_0 \subseteq D_1$. Then, if $d \in f(D_1)$ is a bad decision in the context of D_1, d is a bad decision in the context of D_0.*

Proof. To begin, if $D_0 \subseteq D_1$, then $f(D_1) \subseteq f(D_0)$ so that if $d \in f(D_1)$, then $d \in f(D_0)$. Then, since d is a bad decision in the context of D_1, we know that there does not exist a set of decisions D' such that $D_1 \cup \{d\} \cup D'$ is complete. Keeping this in mind, suppose, to the contrary of the lemma, that there does exist a set of decisions D such that $D_0 \cup \{d\} \cup D$ is complete. But since D_0 is uniquely complete, we must have $D_1 \subseteq D_0 \cup D$. It follows that if we let $D' = D - D_1$, then $D_1 \cup \{d\} \cup D'$ is complete, which contradicts the earlier establishment of the non-existance of such a set D'. □

Notice that the key to proving Lemma 2.2 was the uniqueness of the completability of D_0.

Lemma 2.3. *Let D_0 be a uniquely completable set of decisions, and let each of D_1 and D_2 be a completable set of decisions with $D_0 \subseteq D_1$ and $D_0 \subseteq D_2$. Then, if $d \in f(D_1)$ and $d \in f(D_2)$ and d is a bad decision in the context of D_1, then d is a bad decision in the context of D_2.*

Proof. Clearly, each of D_1 and D_2 are uniquely completable, because D_0 is uniquely completable. Then, by Lemma 2.2, $d \in f(D_0)$ is a bad decision in the context of D_0. Lastly, by Lemma 2.1, we see that $d \in f(D_2)$ is a bad decision in the context of D_2. □

3 Application

Armed with Lemma 2.3, we proceed now to give a method of solving any Sudoku puzzle.

A Sudoku puzzle may be thought of as an initial condition. The solver is then trying to find a completion of the square that obeys the constraint of this initial condition. In the language of this paper, such a condition is a completable set of decisions D_0, and the solver is simply trying to find the set D_1 such that $D_0 \cup D_1$ is complete. The algorithm we describe here is a systematic search for this set.

We begin with the naive approach to solving the general Sudoku puzzle. In the psuedo-code to follow, D_1 should be initially given as an empty set.

```
function SOLVE(D_0, D_1)
    if D is complete then
        return success
    end if
    for all d ∈ f(D_0) do
        D ← D_0 ∪ {d}
        if success = SOLVE(D, D_1) then
            D_1 ← D_1 ∪ {d}
            return success
        end if
    end for
    return failure
end function
```

While this algorithm may be correct, it is not practical due to the combinatorial explosion found in trying to enumerate all possible search paths that

this algorithm would potentially need to go through before finally finding a solution to the puzzle. What we'll find, however, is that Lemma 2.3 offers a major optimization to the code above, as realized by Peter Norvig in his essay []. Applying this lemma, the revised code becomes as follows, with the set B being initially given as an empty set, and as not necessarily being a valid set of decisions.

```
function SOLVE(D_0, D_1, B)
    if D is complete then
        return success
    end if
    for all d ∈ f(D_0) do
        if d ∉ B then
            D ← D_0 ∪ {d}
            if success = SOLVE(D, D_1) then
                D_1 ← D_1 ∪ {d}
                return success
            else
                B ← B ∪ {d}
            end if
        end if
    end for
    return failure
end function
```

Explain why this works here and how it optimizes the problem...