CS 31: Problem Set 4

Will McCall, Mahir Singh, Claire Meli, Hank Patil

August, 2021

1 Problem specification: Proving that solve will always find a solution (if one exists)

SOLVING SUDOKU
Input: A 2D 9x9 integer array $arr[0:8][0:8]$, with every element either being 0 or a valid sudoku number in its location.
Output: A completed sudoku board $arr[0:8][0:8]$ reconstituted from the input.

1.1 Definition:

In addition, for any integer $1 \leq k \leq 9$ and any two index integers i,j: $0 \leq i, j \leq 8$ we define a function $F(k, i, j)$ such that:

$$F(k,i,j) = \begin{cases} 1 & \text{if } k \text{ is an entry meeting the constraints of a sudoku board at index i,j} \\ 0 & \text{otherwise} \end{cases}$$

We are therefore interested in F(k,8,8) (and will worry about how to recover the solution later)

1.2 Base Cases:

$F(k, 0, 0) = 1$. Any integer k can be placed in the first index.

1.3 The Recurrence Relation:

We must establish what to do if i,j are not 0. To avoid explicitly writing the relation between i and j, we reconstitute F as the function F(k, n) where n is an integer from [0:80] where each integer corresponds to a unique square in the board. Notice this does not lose any information as $n \mod 9 = i$ and furthermore $j = \lfloor \frac{n}{9} \rfloor$. In the case of $n > 0$, our ability to solve the problem depends on two things: 1) that the current input does not violate any conditions of the Sudoku puzzle. 2) that the prior input did not violate any conditions of the Sudoku puzzle. In other words, we look for a way to break the validity question into a verification at step n and a recursive verification at step n-1. Incorporating this knowledge with the base case yields:

$$F(k,n) = \begin{cases} 1 & \text{if n} \leftarrow 0 \\ 1 & \text{if isValid(k,n) } \forall k_1 \in 1..9 | k_1 \neq k | F(k_1, n-1) \\ 0 & \text{otherwise} \end{cases}$$

1.4 Proof:

Ignoring the trivial base cases of F(k,0) for $n \leftarrow 0$ and T(1) as explained Section 1.2, we must show 2 things each requiring two explanations:

1) $F(k,n) \leq 1$ if isValid(k,n) $\forall k_1 \in 1..9 | k_1 \neq k | F(k1, n-1)$:

Subcase 1: There is at least one valid way to populate all squares prior to n with a valid input value. In this case, F(k,n) will only ever at most be one depending on if a valid item exists at the insertion site n. This by definition means that $F(k,n) \leq 1$ if isValid(k,n) $\forall k_1 \in 1..9 | k_1 \neq k | F(k1, n-1)$: because F(k,n) will only ever be 1 or 0, hence the identity.

Subcase 2:
There are no valid ways to populate all squares prior to n with a valid input value. In this case, the recursive call on $F(k_1, n-1)$ would fail, thus showing in this case the function will return 0. Since $0 \leq 1$ we have thus proven the identity again.

2) $F(k,n) \geq 1$ if isValid(k,n) $\forall k_1 \in 1..9 | k_1 \neq k | F(k1, n-1)$:

Subcase 1: if there is at least one valid way to populate all squares prior to n with a valid input value, the recursive call will return true, which in combination with the valid insertion at square n, demonstrates the identity.

Subcase 2: if there is at least one valid way to populate all squares prior to n with a valid input value, then the function will automatically return 0, the result of the recursive call, hence, the overall function will return 0, preserving the identity.

We have proven both sides of our inquality, and can thus conclude that:
$F(k,n) = 1$ if isValid(k,n) $\forall k_1 \in 1..9 | k_1 \neq k | F(k1, n-1)$:

1.5 Running Time and Space:

Space: A Squence of 9x9 tables in the stack, which will contain at most 81 frames consecutively. Therefore space is O(1), though in practice will be non-negligibly large.

Time: A recursive definition which is theoretically running in $O(n^m)$ where n represents the number of possible values for a given square and m is the number of squares in the table. calls to isValid() only require checking 27 items, and thus runtime is $O(1)$ Thus it can be said that the recursive solver structure dominates the run time.