

SuDoku: un problema NP-completo

Valverde Rebaza Jorge Carlos

Escuela de Informática

Universidad Nacional de Trujillo

Perú

Resumen

En el presente trabajo se muestra una manera de realizar la demostración de que el popular juego de tipo puzzle llamado **SuDoku** es un problema de tipo *NP-completo*, cuya demostración se realizará a través de pruebas formales. Para cumplir con el objetivo se realiza una codificación de una instancia de Sudoku de manera tal que pueda ser expresada en una Fórmula Normal Conjuntiva y por tanto sea equivalente a un problema *SAT*, a partir de esto será posible la demostración de que $\text{SuDoku} \in NP$ y que $\text{SuDoku} \in NP\text{-Hard}$, las cuales son las condiciones suficientes y necesarias para que quede demostrado de que $\text{SuDoku} \in NP\text{-completo}$.

1. Introducción

El SuDoku se crea a partir de los trabajos del matemático suizo Leonhard Euler (1707-1783) cuando se encontraba trabajando en la demostración de un conjunto de teoremas acerca del cálculo de probabilidades. Sin embargo, el Sudoku visto como un juego, tiene su origen en Indianapolis en 1979. Para entonces no se llamaba Sudoku sino simplemente *Number Place* (el lugar de los números), siendo publicado en la revista *Math Puzzles and Logic Problems* de la empresa especializada en puzzles Dell.

Posteriormente Nikoli, una empresa japonesa especializada en pasatiempos para prensa, lo exportó a Japón publicándolo en el periódico *Monthly Nikolist* en abril de 1984 bajo el título "*Suji wa dokushin ni kagiru*", que se puede traducir como "*los números deben estar solos*", y que posteriormente se abreviaría a Sudoku (su=número, doku=solo) [7].

En el ámbito de las matemáticas es visto como un *problema de satisfacción de restricciones*¹. Los juegos de puzzle como el SuDoku son muy populares debido a que se caracterizan por la simplicidad de sus reglas, simplicidad que necesariamente no significa no complejo, pues por el contrario se necesita de un razonamiento metódico para llegar a una solución. Cada instancia de puzzle tiene una solución única. En el presente documento nos enfocaremos en la demostración formal de que SuDoku es un problema del tipo *NP-completo*.

El análisis de la complejidad del SuDoku ya ha sido revisada por muchos autores, entre ellos [4] quien postula que el SuDoku es un problema puzzle de tipo *ASP-completo* (Another Solution Problem) a partir del cual, mediante inferencias formales, demuestra que un SuDoku de $N \times N$ casillas, es también *NP-completo*. El autor de [3] presenta el problema del SuDoku como un Problema de Satisfacción Restringida

¹Éste tipo de problemas vienen definidos por un conjunto de variables con un conjunto de dominios asociados. Además existen una serie de restricciones. La solución es encontrar una asignación de las variables a un valor de su dominio equivalente de tal forma que se satisfagan las restricciones.

- Constraint Satisfaction Problem (*CSP*), donde se comparan los diferentes regimenes de propagación para resolver el SuDoku y además sugiere nuevas técnicas para resolver el problema sin la necesidad de realizar búsquedas.

Sin embargo, para demostrar que SuDoku es un problema *NP-completo* bastará con codificar una instancia de SuDoku de manera tal que pueda ser expresada en una Fórmula Normal Conjuntiva, lo que significará que SuDoku es equivalente a *SAT*. En el presente documento presentaremos un método formal para lograr esto.

El documento queda estructurado de la siguiente manera: en la sección 2 presentamos el problema del SuDoku definiendo las reglas que caracterizan al juego; en la sección 3 se realiza la codificación de una instancia de SuDoku en una representación booleana que nos permita continuar con la demostración deseada de manera formal; en el capítulo 4 realizamos la demostración formal en sí de que SuDoku es un problema *NP-completo* siguiendo los pasos mostrados en [5]; en la sección 5 presentamos las conclusiones y finalmente las referencias.

2. El problema del SuDoku

Según [1] se tiene que:

Definición 2.1. *Un SuDoku es representado por un tablero de 9×9 casillas, a su vez, éste tablero esta formado por 9 subtableros de 3×3 casillas cada uno (llamados regiones). Inicialmente se llenan algunas casillas del tablero con números del 1 al 9 mientras que otras casillas se dejan en blanco.*

Definición 2.2. *Un SuDoku es resuelto mediante la asignación de números del 1 al 9 en las casillas en blanco de cada fila, cada columna y de cada región de 3×3 .*

En la figura 1 se muestra un ejemplo de como se inicia un tablero de SuDoku. El nivel de complejidad de los puzzles depende de la cantidad de casillas llenas provistas al inicio del juego. Generalmente se llenan 17 casillas al iniciar el juego, sin embargo podrían llenarse más casillas. Una de las características de éste juego es que los números del 1 al 9 solamente son usados según nos convenga, pues las relaciones aritméticas existentes entre ellas son totalmente irrelevantes.

Propiedad 2.3. *Una instancia de SuDoku tiene una solución única.*

Propiedad 2.4. *Una instancia de SuDoku será resuelta siguiendo un único razonamiento.*

Sabiendo que tan sólo existe una única solución para cada puzzle, entonces se utilizarán todos los números del 1 al 9 para ser asignados en las casillas en blanco del tablero. La segunda propiedad indica que en cualquier etapa de solución del SuDoku siempre deberá existir por lo menos una casilla del tablero en blanco en la que se pueda asignar un determinado número. Por lo tanto, el razonamiento consiste en la utilización de reglas de inferencia de manera tal que todas las casillas del tablero sean llenadas.

3. Codificación de SuDoku

Para poder realizar un proceso de demostración adecuado, primero nos encargaremos de codificar un SuDoku de una manera formal que nos permita una fácil representación del tablero. Para realizar esta

	1	8				7		
			3			2		
	7							
				7	1			
6							4	
3								
4			5					3
	2			8				
							6	

Figura 1: Un tablero de SuDoku.

codificación, según [1] y [2] haremos uso de la implementación *Isabelle/Hol*² la cual nos permitirá una representación sencilla e interactiva de las reglas del SuDoku.

Definiendo formalmente mediante *Isabelle/Hol* tenemos que los dígitos que forman parte del tablero serán modelados por un tipo de dato de 9 elementos: $1, 2, 3, \dots, 9$. Entonces decimos que 9 casillas $x_1, x_2, x_3, \dots, x_9$, son válidas (aceptadas en el tablero del SuDoku) si y solo si contienen todos los elementos $1, 2, 3, \dots, 9$. De ésta manera tenemos:

Definición 3.1. $valido(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) \equiv \bigwedge_{d=1}^9 \bigvee_{i=1}^9 x_i = d$

Ahora basandonos en la Definición 3.1 podemos codificar cualquier tablero de SuDoku validando cada fila, cada columna y cada región.

Definición 3.2. $sudoku(\{x_{i,j}\}_{i,j \in \{1,2,3,\dots,9\}}) \equiv \bigwedge_{i=1}^9 valido(x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}, x_{i6}, x_{i7}, x_{i8}, x_{i9}) \wedge \bigwedge_{j=1}^9 valido(x_{1j}, x_{2j}, x_{3j}, x_{4j}, x_{5j}, x_{6j}, x_{7j}, x_{8j}, x_{9j}) \wedge \bigwedge_{i,j \in \{1,4,7\}} valido(x_{ij}, x_{i(j+1)}, x_{i(j+2)}, x_{(i+1)j}, x_{(i+1)(j+1)}, x_{(i+1)(j+2)}, x_{(i+2)j}, x_{(i+2)(j+1)}, x_{(i+2)(j+2)})$

Nótese que podemos codificar un SuDoku con 9 variables booleanas para cada casilla de un tablero de 9×9 , lo que significa que tendremos $9^3 = 729$ variables en total. Cada variable booleana $p_{i,j}^d$ (donde $1 \leq i, j, d \leq 9$) representa el valor de verdad de la ecuación $x_{i,j} = d$ (ver figura 2).

$$\bigvee_{d=1}^9 p_{i,j}^d \quad (1)$$

La siguiente clausula nos asegura que cada casilla $x_{i,j}$ denota solamente uno de los nueve dígitos posibles y que estos no se repiten en una misma fila, columna o región.

²Es un conocido demostrador de teoremas que, mediante la definición de fórmulas matemáticas en un lenguaje formal permite la verificación y prueba de un sistema. Fue desarrollado en la Universidad de Cambridge (Larry Paulson) y en la Universidad Técnica de Múnich (Tobias Nipkow).

$$\bigwedge_{1 \leq d < d' \leq 9} \neg p_{i,j}^d \vee \neg p_{i,j}^{d'} \quad (2)$$

Donde d' indica el complemento de d . Por tanto, la *Definición 3.1* es equivalente a la siguiente caracterización de validación, la cual afirma que las 9 casillas $x_1, x_2, x_3, \dots, x_9$ contienen valores diferentes.

Lema 3.3. $\text{valido}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) \Leftrightarrow \bigwedge_{1 \leq i < j \leq 9} x_i \neq x_j$
 $\Leftrightarrow \bigwedge_{1 \leq i < j \leq 9} \bigwedge_{d=1}^9 x_i \neq d \vee x_j \neq d$

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}	x_{28}	x_{29}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	x_{37}	x_{38}	x_{39}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	x_{47}	x_{48}	x_{49}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}	x_{57}	x_{58}	x_{59}
x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{66}	x_{67}	x_{68}	x_{69}
x_{71}	x_{72}	x_{73}	x_{74}	x_{75}	x_{76}	x_{77}	x_{78}	x_{79}
x_{81}	x_{82}	x_{83}	x_{84}	x_{85}	x_{86}	x_{87}	x_{88}	x_{89}
x_{91}	x_{92}	x_{93}	x_{94}	x_{95}	x_{96}	x_{97}	x_{98}	x_{99}

Figura 2: Representación de un tablero de SuDoku mediante ecuaciones de tipo $x_{i,j} = d$.

4. Demostración de que SuDoku es NP-completo

4.1. SuDoku $\in NP$

Para demostrar que SuDoku es un problema NP-completo debemos empezar por demostrar que $SuDoku \in NP$, para esto debe de encontrar un algoritmo que verifique en tiempo polinomial si un determinado tablero de Sudoku es válido. A continuación se presenta el algoritmo requerido.

Como se puede notar, el *Algoritmo 1* se vale de la función $sudoku(x_{i,j})$ presentada en la *Definición 3.2* para validar un tablero de SuDoku T en un tiempo polinomial. Por lo tanto, dada la existencia del *Algoritmo 1* queda demostrado que $SuDoku \in NP$.

4.2. SuDoku $\in NP\text{-Hard}$

Esto significa que debemos demostrar que existe una fórmula *SAT* que sea equivalente a cualquier instancia de SuDoku (cualquier tablero T).

Algoritmo 1 Verifica SuDoku

Entrada: Tablero de SuDoku T .**Salida:** verificación del Tablero T , valor booleano 1 si se satisface, 0 en caso contrario.

1. Representar cada elemento de T de la forma: $x_{i,j} = d$; donde i representa el número de fila, j el número de columna, y d el valor contenido en dicha casilla ($i,j,d \in \{1, 2, 3, \dots, 9\}$).

para $i = 1$ hasta 9 hacer

para $j = 1$ hasta 9 hacer

2. $p_{i,j}^d \leftarrow sudoku(x_{i,j})$

si $p_{i,j}^d = 0$ entonces

3. devolver 0

fin si

fin para

fin para

3. devolver 1

Un problema *SAT* es representado mediante el uso de variables proposicionales $x_1, x_2, x_3, \dots, x_n$, sobre las cuales puede asignarse un valor de verdad: 0 (falso) o 1 (verdadero). Un literal l es una variable x_i o su complemento $\neg x_i$, es decir, un literal positivo o negativo respectivamente. Una cláusula w es una disyunción de literales y una fórmula φ llamada *Fórmula Normal Conjuntiva (CNF)*, es una conjunción de cláusulas.

Cuando a un literal l_j de una cláusula w_a se le asigna el valor de verdad 1 satisfaciendo la cláusula, entonces ésta será llamada *cláusula satisfecha*. Si a dicho literal se le asigna el valor de verdad 0, entonces la cláusula no se satisficará. Una cláusula con un solo literal es llamada *cláusula unitaria* y el valor de verdad de dicho literal debe ser 1 para hacer que la cláusula sea satisfecha. La derivación en una cláusula vacía indica que la cláusula no se satisficará. Todo esto significa que, una fórmula será satisfecha si todas sus cláusulas son satisfechas. Por lo tanto, *el problema SAT consiste en decidir si la asignación de los valores de verdad de las variables hace que la fórmula sea satisfecha*.

Ahora, generaremos una fórmula booleana φ que represente las instancias de un tablero T de SuDoku, es decir:

$$SuDoku = \{\langle \varphi \rangle / \varphi \text{ es una fórmula booleana satisfacible}\} \quad (3)$$

Nuevamente nos basaremos en la *Definición 3.2* para generar φ , la cual estará formada por la conjunción de todas las cláusulas (CNF) obtenidas a partir de $sudoku(x_{i,j})$, es decir:

$$\varphi = sudoku(x_{1,1}) \wedge sudoku(x_{1,2}) \wedge \dots \wedge sudoku(x_{9,8}) \wedge sudoku(x_{9,9}) \quad (4)$$

Según la ecuación (1), la ecuación (4) equivale a:

$$\varphi = p_{1,1}^d \wedge p_{1,2}^d \wedge \dots \wedge p_{9,8}^d \wedge p_{9,9}^d \quad (5)$$

Sin embargo, se debe cumplir que en cada fila, columna y región, ningún elemento debe repetirse. Ésta condición es asegurada con la clausula mostrada en la ecuación (2), por tanto, la ecuación (4) realmente sería equivalente a:

$$\varphi = \left(\neg p_{1,1}^d \vee \neg p_{1,1}^{d'} \right) \wedge \left(\neg p_{1,2}^d \vee \neg p_{1,2}^{d'} \right) \wedge \dots \wedge \left(\neg p_{9,8}^d \vee \neg p_{9,8}^{d'} \right) \wedge \left(\neg p_{9,9}^d \vee \neg p_{9,9}^{d'} \right) \quad (6)$$

Entonces tenemos que, con la ecuación (6) llegamos a obtener una fórmula booleana φ satisfacible, es decir que $SAT \leq_P$ SuDoku (SAT es reducible en tiempo polinómico en SuDoku), por lo tanto SuDoku $\in NP$ -Hard.

Por otro lado, como previamente el tablero T ha sido verificado por el *Algoritmo 1* se tendrá siempre que cada clausula de φ tendrá el valor de verdad 1 (de lo contrario $SuDoku \notin NP$). Por lo tanto, mediante la reducción de la ecuación (6) se tendrá siempre que φ será una conjunción de 1's, es decir:

$$\varphi = 1 \wedge 1 \dots \wedge 1 \wedge 1 = 1 \quad (7)$$

Por consiguiente $\varphi \in SAT$ y siendo φ la representación booleana de una instancia de SuDoku y habiendose demostrado que $Sudoku \in NP$ y que $Sudoku \in NP$ -Hard, por lo tanto **queda demostrado que SuDoku $\in NP$ -completo.**

	2							
			6					3
	7	4		8				
					3			2
	8			4				1
6			5					
				1		7	8	
5					9			
								4

1	2	6	4	3	7	9	5	8
8	9	5	6	2	1	4	7	3
3	7	4	9	8	5	1	2	6
4	5	7	1	9	3	8	6	2
9	8	3	2	4	6	5	1	7
6	1	2	5	7	8	3	9	4
2	6	9	3	1	4	7	8	5
5	4	8	7	6	9	2	3	1
7	3	1	8	5	2	6	4	9

Figura 3: Un tablero de SuDoku y su solución.

5. Conclusiones

1. El SuDoku es un juego de tipo puzzle muy popular en el mundo, dónde, para cada instancia de SuDoku se tiene una única solución. En el ámbito de las matemáticas es visto como un *problema de satisfacción de restricciones*.
2. Previamente, diversos autores ya han demostrado que SuDoku es un problema NP-completo, entre ellos, [4] postula que SuDoku es un problema de tipo *ASP-completo* (Another Solution Problem) a partir del cual demuestra que SuDoku es también *NP-completo*.

3. En el presente documento se ha logrado demostrar formalmente que, a partir de la adecuada codificación de una instancia de SuDoku, éste se puede convertir a una fórmula booleana φ satisfacible, de tipo *SAT*, por lo tanto SuDoku \in *NP-completo*.
4. La demostración se ha basado en un tablero de SuDoku de tamaño 9×9 pero se podría generalizar para un tablero de $n \times n$.

Referencias

- [1] Inês Lynce and Joël Ouaknine, *Sudoku as a SAT problem*, Oxford University Computing Laboratory, UK.
- [2] Tjark Weber, *A SAT-based Sudoku Solver*, Institut für Informatik, Technische Universität München Boltzmannstr. 3, D-85748 Garching b. München, Germany.
- [3] H. Simonis, *Sudoku as a constraint problem*, In CP Workshop on Modeling and Reformulating Constraint Satisfaction Problems, pages 13-27, October 2005.
- [4] Takayuki YATO, *Complexity and Completeness of Finding another Solution and its Application to Puzzles*, Graduate School of Science The University of Tokyo, 2003.
- [5] T. H. Cormen, C.E. Leiserson, R. L. Rivest, *Introduction to Algorithms*, Mc.Graw-Hill Book Company, pages 916-946, 2000.
- [6] T. Nipkow, L.C. Paulson, M.Wenzel, *A Proof Assistant for Higher-Order Logic*, February 2002.
- [7] Todosudoku.com, <http://www.todosudoku.com/comojugar.html>.
- [8] Sudoku and SAT, <http://sed.free.fr/complex/sudoku.html>.