

# NP-Probleme in Logikspielen

**Emilia Wittmers, Markus Güntert**

betreut von: Prof. Christoph Kreitz

Vorlesungsergänzendes Projekt Theoretische Informatik II  
Sommersemester 2009

Datum der Abgabe: 19. Juli 2009



## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Mastermind</b>	<b>2</b>
2.1	Die Spielregeln . . . . .	2
2.2	Grundlegende Begriffe . . . . .	3
2.3	Die mathematische Definition . . . . .	3
2.4	Die abgewandelte Version „Static Mastermind“ . . . . .	4
2.5	Das Mastermind-Erfüllbarkeits-Problem . . . . .	4
2.5.1	Die Behauptung . . . . .	5
2.5.2	Der Beweis . . . . .	5
2.5.3	Das Problem der eindeutigen Lösung . . . . .	7
<b>3</b>	<b>Sudoku</b>	<b>8</b>
3.1	Die Spielregeln . . . . .	8
3.2	Eigenschaften von Sudoku-Puzzles . . . . .	8
3.3	Berechenbarkeit . . . . .	9
	<b>Quellenverzeichnis</b>	<b>13</b>

## Abbildungsverzeichnis

1	Das klassische Mastermind-Spiel . . . . .	3
2	minimales Sudoku-Puzzle . . . . .	11
3	nicht eindeutig lösbar . . . . .	11



# 1 Einleitung

NP-Probleme sind Probleme, die mithilfe derzeit bekannter deterministischer Algorithmen nicht in polynomieller Zeit gelöst werden können.

Ein Problem wird NP-vollständig genannt, wenn es NP-hart ist (d.h. wenn es mindestens genauso schwer lösbar wie alle NP-Probleme ist) und selbst in der Klasse der NP-Probleme liegt.

NP-vollständige Probleme tauchen in verschiedensten Bereichen auf. Viele bekannte von ihnen, wie beispielsweise das Cliquen-Problem, das Knotenüberdeckungsproblem, das Färbbarkeitsproblem und das Problem des Hamiltonschen Kreises, sind Probleme der Graphentheorie. Aber auch im Bereich der Spiele lassen sich allerhand Probleme finden, die keine polynomielle Lösung besitzen.

Spiele mit NP-vollständigen Problemen sind unter anderem Schiffeversenken, Tetris, Minesweeper und SameGame, sowie die beiden Denkspiele Mastermind und Sudoku, die im Folgenden dieser Arbeit als Beispiele für NP-Probleme in Logikspielen vorgestellt werden.

## 2 Mastermind

Mastermind ist ein Logikspiel, das von dem Israeli Mordechai Meirovitz 1970 entwickelt worden ist. Es kann von zwei Personen als Brettspiel gespielt werden oder von einer Person, die gegen einen Computer antritt.

Im Web sind hunderte Varianten des Spiels zu finden. Allein auf der Seite 'onlinespiele-sammlung'<sup>1</sup> werden 163 verschiedene Online-Mastermind-Spiele aufgelistet. Eine der vielen Abwandlungen des originalen Mastermind-Spiels ist 'Static Mastermind'. Dieses Spiel dient als Grundlage für das Mastermind-Efüllbarkeits-Problem, das von Jeff Stuckman und Guo-Qiang-Zhang im Jahr 2006 als NP-vollständig bewiesen worden ist. [Stuckman and Zhang, 2006]

### 2.1 Die Spielregeln

In dem Spiel Mastermind geht es darum eine verdeckte Farbkombination bestimmter Länge mit einer minimalen Anzahl an Versuchen zu erraten.

Der erste Spieler (spielt man allein, wäre dies in dem Fall der Computer) legt den Farbcode fest, der für den zweiten Spieler nicht sichtbar sein darf. Der zweite Spieler versucht den Farbcode zu ermitteln, indem er eine Farbkombination rät und diese anschließend von dem ersten Spieler bewerten lässt.

Eine richtige Farbe an der richtigen Position wird mit einem schwarzen Stift bewertet, eine richtige Farbe an der falschen Position wird mit einem weißen Stift bewertet. Die schwarzen und weißen Stifte verraten die Anzahl der richtig oder halbrichtig geratenen Farben. Wie die einzelnen Farben jeweils tatsächlich bewertet worden sind, kann der zweite Spieler jedoch nur durch Vermutungen herausfinden.

Anhand der Informationen, die durch die schwarzen und weißen Stifte gegeben werden, probiert er eine weitere Farbkombination aus, die er anschließend wieder von dem ersten Spieler bewerten lässt.

Der Wechsel zwischen Rateversuch des zweiten Spielers und Bewertung durch den ersten Spieler findet solange statt, bis der zweite Spieler die Lösung erfolgreich bestimmt hat.

---

<sup>1</sup>[http://www.onlinespiele-sammlung.de/mastermind/list-of-mastermind-games.php?records\\_to\\_show=-1&list=1](http://www.onlinespiele-sammlung.de/mastermind/list-of-mastermind-games.php?records_to_show=-1&list=1)

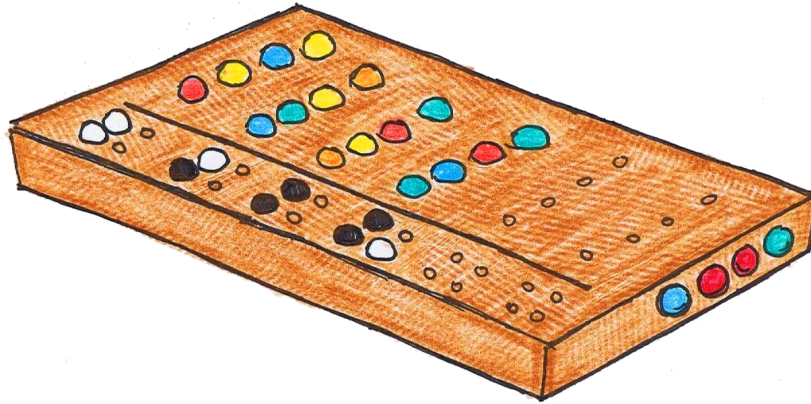


Abbildung 1: Das klassische Mastermind-Spiel

## 2.2 Grundlegende Begriffe

**Lösung:** die zu erratene Farbkombination

**Vorschlag:** eine geratene Farbkombination

**Schwarz:** eine geratene Farbe befindet sich in der Lösung an der selben Position

**Weiß:** eine geratene Farbe befindet sich in der Lösung an einer anderen Position

**Bewertung:** die Anzahl an schwarzen und weißen Stiften, die für einen Vorschlag vergeben werden

## 2.3 Die mathematische Definition

Das Mastermind-Spiel kann mathematisch wie folgt beschrieben werden:

- $k$  gibt die Anzahl aller möglichen Farben an
- $N_k := \{1, 2, \dots, k\}$  ist die Menge aller  $k$  Farben
- $l$  ist die Länge der Lösung und der Vorschläge
- $N_l := \{1, 2, \dots, l\}$  ist die Menge aller  $l$  möglichen Positionen, die eine Farbe in den Vorschlägen bzw. der Lösung einnehmen kann
- $N_l^k$  ist die Menge aller möglichen Vorschläge der Länge  $l$  mit den Farben aus  $N_k$ , wobei die Lösung ebenfalls Element von  $N_l^k$  ist
- das Paar  $p(x, y) := (b, w - b)$  gibt die Bewertung eines Vorschlags an, es gilt:

- $x, y \in N_l^k$  ( $x$  ist der Vorschlag und  $y$  die Lösung)
- $b := \#\{i | i \in N_l, x_i = y_i\}$  ( $b$  ist die Anzahl der schwarzen Stifte, d.h. die Anzahl aller Positionen, für die gilt, dass die Farbe an der Position  $i$  im Vorschlag gleich der Farbe an der Position  $i$  in der Lösung ist)
- $w := \sum_{j \in N_k} \min(\#\{i \in N_l | x_i = j\}, \#\{i \in N_l | y_i = j\})$  ( $w$  ist die Anzahl aller erratenen Farben, die ebenfalls in der Lösung vorkommen, wobei jedoch mehrfach verwendete Farben im Vorschlag höchstens so oft gezählt werden, wie oft sie in der Lösung auch tatsächlich auftauchen; da bei der Aufsummierung aller richtig geratenen Farben auch die mitgezählt werden, die an der richtigen Position stehen, muss von  $w$   $b$  abgezogen werden, um die Anzahl der zu vergebenen weißen Stifte zu erhalten)

## 2.4 Die abgewandelte Version „Static Mastermind“

Bei dem Spiel 'Static Mastermind' wird eine bestimmte Anzahl an Vorschlägen zusammen mit ihren Bewertungen bereits zu Beginn des Spiels vorgegeben. Der ratende Spieler hat einen Versuch, um mit Hilfe dieser Angaben die Lösung zu bestimmen.

In dem Spiel von Don Greenwell<sup>2</sup> werden 6 Vorschläge mit jeweils 4 Farben (gewählt aus einer Menge von 6 Farben) dem Spieler gezeigt, so dass es möglich ist, anhand dieser Vorschläge und ihren Bewertungen mit nur einem Versuch die Lösung zu erraten.

## 2.5 Das Mastermind-Erfüllbarkeits-Problem

Das Mastermind-Erfüllbarkeits-Problem, kurz MSP (Mastermind Satisfiability Problem), ist ein Entscheidungsproblem und basiert auf dem Spiel 'Static Mastermind' (siehe Kapitel 2.4):

*Gegeben sei eine Menge an Vorschlägen und die dazugehörigen Bewertungen. Existiert für diese Konstellation wenigstens eine gültige Lösung?*

---

<sup>2</sup><http://math2.eku.edu/greenwell/MMSTATIC/index.html>

### 2.5.1 Die Behauptung

Das MSP mit folgenden Ein- und Ausgaben ist bezüglich der Länge  $l$  (mit  $k > 1$ ) NP-vollständig.

**Input:** Die Eingabe für das Problem ist eine Menge an Vorschlägen  $G \subseteq N_l^k$  und die dazugehörigen Bewertungen  $(b_g, b_g - w_g)$  für jeden Vorschlag  $g \in G$ .

**Output:** Die Ausgabe des Problems ist  $JA$ , wenn eine Lösung  $s \in N_l^k$  existiert, so dass gilt  $\forall g \in G. p(g, s) = (b_g, b_g - w_g)$ . Die Ausgabe ist  $NEIN$ , wenn ein solches  $s$  nicht existiert.

### 2.5.2 Der Beweis

**MSP liegt in der Klasse NP** Eine Orakel-Turingmaschine generiert nichtdeterministisch Lösungsvorschläge für das Mastermind-Erfüllbarkeits-Problem. Die Gültigkeit einer Lösung für eine Instanz des MSP kann in polynomieller Zeit ausgewertet werden: Die Erfüllbarkeit eines Lösungsvorschlags  $s$  wird überprüft, indem für jeden gegebenen Vorschlag  $g$  getestet wird, ob  $p(g, s) = (b_g, b_g - w_g)$ . Das Testen eines Lösungsvorschlags  $s$  benötigt  $(l + k * (2 * l + 1) + 1) * \#G$  Schritte. Diese Zahl ergibt sich aus der Anzahl aller Vorschläge  $g$  multipliziert mit der Summe der Anzahl der Schritte, die für die Berechnung der schwarzen Stifte nötig ist ( $l$ ), und der Anzahl der Schritte, die für die Berechnung der weißen Stifte gebraucht wird ( $k * (2 * l + 1) + 1$ ).

**VC ist polynomiell reduzierbar auf MSP** Um zu beweisen, dass das Mastermind-Erfüllbarkeits-Problem NP-hart ist, wird gezeigt, dass das NP-vollständige Knotenüberdeckungsproblem VC (Vertex-Cover Problem) polynomiell reduzierbar auf MSP ist.

*Das Knotenüberdeckungsproblem* VC gehört zu den 21 klassischen Problemen, die von Richard Karp 1972 als NP-vollständig bewiesen worden sind.

VC: Gibt es in einem gegebenen Graphen  $(V, E)$  der Größe  $n$  eine Teilmenge  $V' \subseteq V$  mit höchstens  $m$  Elementen ( $m \leq \#V$  sei ebenfalls gegeben), so dass aus jeder Kante aus  $E$  mindestens eine Ecke in  $V'$  liegt?

$VC = \{((V, E), m) | \exists V' \subseteq V. \#V' \leq m \wedge V' \text{ Knotenüberdeckung von } (V, E)\}$

Die Transformation der Eingaben für VC in Eingaben für MSP, die nötig ist, um VC auf MSP polynomiell reduzieren zu können, wird im Folgenden beschrieben.

Für den Graphen  $(V, E)$  soll die entsprechende MSP-Instanz die Eigenschaften haben  $k = \#V + \#E + 2$  und  $l = 3 + 2\#V + \#E$ .

Jeder Knoten aus  $V$  und jede Kante aus  $E$  wird als eine andere Farbe kodiert, zusätzlich gibt es die zwei Kontrollfarben  $Y$  und  $N$ . Wie die Zusammensetzung der Länge der Farbkombination (= die Anzahl der Positionen) zustande kommt, soll im Verlauf des Beweises deutlich werden (für eine detaillierte Beschreibung wird an dieser Stelle auf das Paper von Stuckmann und Zhang [Stuckman and Zhang, 2006] verwiesen).

$K = \{v_1, v_2, \dots, v_{\#V}, e_1, e_2, \dots, e_{\#E}, Y, N\}$  sei die Menge der Farben.

Die Menge der gegebenen Vorschläge soll wie folgt definiert werden:

1. Der erste Vorschlag ist  $(N, N, \dots, N)$  mit der Bewertung  $(0, 0)$ . Dieser soll verhindern, dass die Kontrollfarbe  $N$  in der Lösung auftaucht.
2. Der zweite Vorschlag lautet  $(Y, Y, Y, N, \dots, N)$  mit der Bewertung  $(3, 0)$ , um zu erzwingen, dass die ersten drei Positionen mit der Kontrollfarbe  $Y$  belegt werden.
3. Die nächsten  $\#E$  Vorschläge werden so konstruiert, dass für jede  $i$ -te Kante  $(a, b)$  ein Vorschlag  $(e_i, a, b, N, \dots, N)$  existiert mit einer Bewertung  $(0, 2)$ .
4. Der letzte Vorschlag ist  $(Y, Y, Y, v_1, v_2, \dots, v_{\#V}, N, \dots, N)$  mit der Bewertung  $(3, m)$ .

Es soll nun gezeigt werden, dass VC  $((V, E), m)$  genau dann eine Lösung besitzt, wenn für die oben beschriebene MSP-Instanz eine Lösung existiert.

„ $\Rightarrow$ “ Angenommen es gibt für die oben beschriebene MSP-Instanz eine Lösung, dann ist die Farbe  $N$  wegen Bedingung (1) nicht in der Lösung enthalten. Aus dieser Tatsache und Bedingung (4) folgt, dass genau  $m$  Knoten  $w_1, w_2, \dots, w_m$  aus  $V$  (kodiert als Farben) Teil der Lösung sein müssen. Mithilfe von Bedingung (3) kann gezeigt werden, dass die Teilmenge  $V' = \{w_1, w_2, \dots, w_m\}$  eine Knotenüberdeckung von  $(V, E)$  ist, d.h. von jeder Kante aus  $E$  liegt ein Knoten in  $V'$ . Nach Bedingung (3) muss nämlich für jede  $i$ -te Kante  $e_i$  ein Knoten aus dem Paar  $(a, b)$  in  $V'$  liegen, damit die Bewertung  $(0, 2)$  erfüllt sein kann. Hieraus folgt, dass wenn es eine Lösung für die MSP-Instanz gibt,  $V' = \{w_1, w_2, \dots, w_m\}$  eine Knotenüberdeckung von  $(V, E)$  der Größe  $m$  ist.

„ $\Leftarrow$ “ Angenommen  $V' = \{w_1, w_2, \dots, w_m\}$  ist eine Knotenüberdeckung von  $(V, E)$  der Größe  $m$ , dann kann die entsprechende Lösung der MSP-Instanz als das Tupel  $s = (Y, Y, Y; Y, \dots, Y; w_1, w_2, \dots, w_n, Y, \dots, Y; e_{i_1}, e_{i_2}, \dots, e_{i_t}, Y, \dots, Y)$  konstruiert werden. Die Kante  $e_{ij} = (a, b)$  taucht in der Lösung genau dann auf, wenn die Menge  $\{a, b\}$  nicht Teilmenge von  $V'$  ist, d.h.  $e_{ij}$  enthält nur genau einen Knoten aus  $V'$ . Das Tupel  $s$  wird durch Semikola in verschiedene Bereiche unterteilt (für eine detaillierte Beschreibung der Unterteilung wird an dieser Stelle erneut auf das Paper von Stuckmann und Zhang [Stuckman and Zhang, 2006] verwiesen). Die Korrektheit der Bewertungen für die gegebenen Vorschläge, die unter den Punkten (1) bis (4) definiert worden sind, bezüglich der Lösung  $s$  kann leicht gezeigt werden. Hieraus folgt, dass wenn es eine Knotenüberdeckung  $V' = \{w_1, w_2, \dots, w_m\}$  von  $(V, E)$  der Größe  $m$  gibt, existiert eine Lösung  $s$  für die MSP-Instanz.

Da die oben beschriebene Transformation der Eingaben für VC in Eingaben für MSP in polynomieller Zeit durchführbar ist, ist MSP NP-hart.

q.e.d.

### 2.5.3 Das Problem der eindeutigen Lösung

Das Problem, ob eine gefundene Lösung eindeutig ist, d.h. ob es für eine gegebene Anzahl an Vorschlägen nur genau eine Lösung gibt, wenn es eine Lösung gibt, ist nicht schwieriger zu lösen als MSP selbst. Gibt es nämlich einen Algorithmus, der eine Lösung  $s$  für eine MSP-Instanz bestimmen kann, so lässt sich auch in polynomieller Zeit überprüfen, ob diese Lösung die einzige ist. (Siehe hierzu unter „Uniqueness of Solution“ in [Stuckman and Zhang, 2006])

## 3 Sudoku

Sudoku ist ein japanisches Puzzlespiel, das seit 2005 international populär ist, es gilt als der *Rubik's Cube* des 21. Jahrhunderts [Pendlebury, 2005]. Sudoku-Puzzles werden traditionell mit Papier und Stift gelöst, mittlerweile erfreuen sich aber auch technische Umsetzungen (z.B. auf Computern oder Handys) immer größerer Beliebtheit.

### 3.1 Die Spielregeln

Die Spielregeln sind einfach zu verstehen. Das Schlussfolgern, welches zum Ziel des Spiels führt, macht seinen Reiz aus. Ein klassisches Sudoku-Puzzle besteht aus einem Gitterfeld mit  $3 \times 3$  *Blöcken*, welche wiederum in  $3 \times 3$  *Felder* unterteilt sind (allgemein:  $n \times n$  Blöcke mit je  $n \times n$  Feldern). Es besteht demnach aus insgesamt 81 Feldern, die in 9 Blöcke, 9 Spalten und 9 Zeilen aufgeteilt sind.

Ziel des Spiels ist es, die Felder mit den *Zahlen 1 bis 9* (1 bis  $n^2$ ) so auszufüllen, dass jede Zahl *genau einmal in jedem Block, in jeder Spalte und in jeder Zeile* vorkommt. Zu Beginn des Spiels ist ein Teil der Felder bereits ausgefüllt, durch Schlussfolgern soll der Spieler nach und nach die leeren Felder ausfüllen. Der Schwierigkeitsgrad variiert, je nachdem wie viele Felder bereits vorausgefüllt sind und wie schwierig sich die notwendigen Schlussfolgerungen gestalten.

### 3.2 Eigenschaften von Sudoku-Puzzles

Es folgt eine Reihe von Definitionen, anhand derer Sudoku-Puzzles klassifiziert werden können:

- Ein Puzzle ist *partiell*, wenn nicht alle Felder ausgefüllt sind; sonst ist es *vollständig*.
- Ein Puzzle heißt *lösbar*, wenn es partiell ist und eine Belegung der freien Felder existiert, so dass alle in 3.1 beschriebenen Regeln erfüllt sind.
- Ein Puzzle heißt *eindeutig lösbar*, wenn es partiell ist und es genau eine Lösung gibt, die die in 3.1 beschriebenen Spielregeln einhält.

- Ein Puzzle heißt *deterministisch lösbar*, wenn es partiell ist und es beim Lösen immer mindestens ein Feld gibt, das allein durch Schlussfolgern eindeutig ausgefüllt werden kann. Sofern es einen Zeitpunkt gibt, an dem kein Feld definitiv bestimmt werden kann (und der Spieler einen möglichen Schritt „probieren“ muss), ist es eben genau nicht deterministisch lösbar.
- Ein *minimales* Puzzle ist eindeutig lösbar und dies nicht mehr, sobald eine der vorgegebenen Zahlen entfernt wird.

Beschreiben  $S_l$ ,  $S_{el}$  und  $S_{dl}$  die Mengen der lösbaren, eindeutig lösbaren und deterministisch lösbaren Puzzles, so gilt  $S_{dl} \subset S_{el} \subset S_l$ .

### 3.3 Berechenbarkeit

Das generalisierte Sudoku-Problem beschreibt die Menge

$$SUDOKU = \{ \langle S \rangle \mid S \text{ ist partielles Sudoku-Puzzle} \wedge S \text{ ist lösbar} \}$$

und ist **NP-vollständig**. Der entsprechende Beweis wird in [Yato, 2003] geführt und sei im Folgenden skizziert: Es wird gezeigt, dass das *Another Solution Problem* für  $SUDOKU$ <sup>3</sup> NP-vollständig ist, was impliziert, dass  $SUDOKU$  selbst NP-vollständig ist.<sup>4</sup> Das bekannte *3SAT*-Problem wird zunächst auf *1-in-3 SAT* reduziert, für den Beweis sei auf [Schaefer, 1978] verwiesen. *1-in-3 SAT* schränkt *3SAT* insofern ein, dass jede Klausel nur genau ein wahres Literal (und damit zwei falsche) enthalten darf. Analog zu *1-in-3 SAT* kann die NP-Vollständigkeit von *1-in-4 SAT* nachgewiesen werden.

Anschließend wird das Problem *LATINSQUARE* betrachtet, welches als Vereinfachung des Sudoku-Problems angesehen werden kann. Eine Instanz von *LATINSQUARE* besteht aus  $n \times n$  Feldern, die mit  $n$  verschiedenen Symbolen zu belegen sind, sodass jedes Symbol in jeder Zeile und in jeder Spalte jeweils genau einmal auftritt.  $SUDOKU$  ist ein Spezialfall von *LATINSQUARE* mit  $n = 9$  sowie der Zusatzbedingung, dass in der Aufteilung in neun  $3 \times 3$ -Quadrate in jedem dieser

<sup>3</sup>ASP für  $SUDOKU \equiv$  gibt es für partielles Sudoku mit gegebener Lösung mehr als eine Lösung, entsprechend obiger Definition:

$ASP-SUDOKU = \{ \langle S \rangle \mid S \text{ ist partielles Sudoku-Puzzle} \wedge S \text{ ist eindeutig lösbar} \}$

<sup>4</sup>ASP-Vollständigkeit impliziert NP-Vollständigkeit siehe [Yato, 2003]

Quadrate alle Symbole jeweils genau einmal auftreten müssen.

In [Colbourn et al., 1984] wird *1-in-4 SAT* auf *ASP-LATINSQUARE* reduziert und dadurch die NP-Vollständigkeit von *LATINSQUARE* bewiesen. [Yato, 2003] beweist abschließend, dass *LATINSQUARE* in polynomieller Zeit auf *SUDOKU ASP*-reduziert werden kann. Zusammengefasst also:

$$3SAT \leq_p 1\text{-in-3 SAT} \leq_p 1\text{-in-4 SAT} \leq_p \text{ASP-LATINSQUARE} \leq_p \text{ASP-SUDOKU} \\ \Rightarrow \text{SUDOKU} \in NP \quad \square$$

**Umgang mit der NP-Vollständigkeit.** Ist von einem gegebenen Sudoku-Puzzle bekannt, dass dieses *deterministisch lösbar* ist, so kann es in polynomieller Zeit gelöst werden. Ein entsprechendes Verfahren ist in [Ist et al., 2006] erläutert: Ein Sudoku-Puzzle wird dazu als *SAT*-Problem repräsentiert. Um das Sudoku-Puzzle in KNF darzustellen, werden insgesamt  $9 \cdot 9 \cdot 9 = 729$  Variablen benötigt. Variable  $s_{xyz}$  besitze dabei genau dann den Wert 1, wenn das Feld der Spalte  $x$  in Reihe  $y$  den Wert  $z$  besitzt. Auf diese Weise werden für jedes Feld neun Variablen definiert, die mögliche Belegungen des Feldes mit den Zahlen 1 bis 9 beschreiben. Außerdem sind die in 3.1 aufgeführten Regeln in der Formel zu repräsentieren.<sup>5</sup> Ziel ist es, anhand der initial ausgefüllten Felder sämtliche Variablen  $s_{xyz}$  nach und nach mit den Werten 0 und 1 zu belegen; hierfür werden spezielle Inferenz-Algorithmen beschrieben, die in polynomieller Zeit ausgeführt werden können.

Ein alternativer Ansatz für das generalisierte Sudoku-Problem führt dieses auf *GC* (Graph Coloring) zurück. Ein klassisches Sudoku-Puzzle wird dabei durch 81 Knoten repräsentiert, welche jeweils genau einem Feld des Sudoku entsprechen. Zwei Knoten sind genau dann verbunden, wenn die Felder des Sudoku in der gleichen Spalte, in der gleichen Zeile oder in der gleichen Box liegen. Das Lösen des Sudoku-Puzzles entspricht dem Färben des Graphen mit 9 Farben, so dass keine zwei verbundenen Knoten die gleiche Farbe besitzen. Methoden zum Umgang mit diesem Problem sind neben *Backtracking* und *Brute Force* vor allem *Contraction* und *Greedy Coloring*. Bei klassischen Sudoku-Puzzles mit 81 Feldern arbeiten diese Algorithmen effizient, das Problem wird aber bei größeren Sudoku nicht mehr handhabbar.

**Erstellen von Sudoku-Puzzles.** Beim Erstellen von Sudoku-Puzzles gibt es prinzipiell zwei Vorgehensweisen: *Bottom-up* und *Top-down*. *Bottom-up* bezeichnet das Vorfüllen eines leeren Sudoku bis ein eindeutig lösbares Sudoku-Puzzle entsteht,

---

<sup>5</sup>Ein entsprechendes Verfahren wird ebenfalls in [Ist et al., 2006] erläutert.

Top-down wählt den umgekehrten Weg. Aus einem vollständigen Sudoku werden dabei nach und nach Zahlen entfernt, sodass ständig eindeutige Lösbarkeit garantiert werden kann. In [Felgenhauer and Dresden, 2005] wird gezeigt, wie die Menge aller vollständigen, den laut 3.1 definierten Spielregeln genügenden Sudoku effizient konstruiert werden kann (die Kardinalität dieser Menge ist ungefähr  $6,6 \cdot 10^{21}$ ). Durch Nutzung von Zufallsfunktionen und Backtracking können so eindeutig lösbare klassische Sudoku-Puzzles erstellt werden, bei größeren Sudoku wäre diese Vorgehensweise aber nicht mehr praktikabel.

[Lewis and Rhyd, 2007] beschreibt einen Algorithmus, der zusätzlich Meta-Heuristiken verwendet, um Sudoku-Puzzles zu erstellen. Ein weiterer Ansatz, der in [Leone et al., 2008] erläutert wird, ist das *transformieren* partieller Sudoku-Puzzles zu neuen Aufgaben. Wie viele eindeutig lösbare Sudoku-Puzzles es insgesamt gibt, ist unbekannt.

**Minimale Sudoku-Puzzles.** Bisher ist nicht bekannt, wie viele Felder mindestens vorausgefüllt sein müssen, damit ein Sudoku-Puzzle eindeutig lösbar sein kann. In [Davis, 2008] ist ein partielles klassisches Puzzle aufgeführt, bei dem nur 17 der insgesamt 81 Felder vorausgefüllt sind und welches dennoch eindeutig lösbar ist. Es wird angenommen, dass bei einem klassischen Puzzle mindestens 17 vorausgefüllte Felder notwendig sind, dieser Sachverhalt konnte allerdings noch nicht bewiesen werden. Interessanterweise kann dagegen ein partielles klassisches Sudoku-Puzzle mit 77 ausgefüllten Feldern nicht eindeutig lösbar sein.

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

Abbildung 2: minimales Sudoku-Puzzle

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

Abbildung 3: nicht eindeutig lösbar



## Quellenverzeichnis

- [Colbourn et al., 1984] Colbourn, C. J., Colbourn, M. J., and Stinson, D. R. (1984). The computational complexity of recognizing critical sets. In *Graph theory, Singapore 1983*, number 1073 in Lecture Notes in Math., page 248–253. Springer.
- [Davis, 2008] Davis, T. (2008). The mathematics of sudoku. [www.geometer.org/mathcircles/sudoku.pdf](http://www.geometer.org/mathcircles/sudoku.pdf), accessed 2009-07-13.
- [Felgenhauer and Dresden, 2005] Felgenhauer, B. and Dresden, T. (2005). Enumerating possible sudoku grids.
- [Ist et al., 2006] Ist, I. L., Lynce, I., and Ouaknine, J. (2006). Sudoku as a sat problem. In *Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics, AIMATH 2006, Fort Lauderdale*. Springer.
- [Kreitz, 2009] Kreitz, C. (2009). Das P-NP Problem. <http://www.cs.uni-potsdam.de/ti/lehre/downloads/TI-II/slides-6.2.pdf>, accessed 2009-07-18.
- [Leone et al., 2008] Leone, A., Mills, D., and Vaswani, P. (2008). Sudoku: Bagging a difficulty metric and building up puzzles. <http://www.math.washington.edu/~morrow/mcm/team2280.pdf>, accessed 2009-07-13.
- [Lewis and Rhyd, 2007] Lewis and Rhyd (2007). Metaheuristics can solve sudoku puzzles. *Journal of Heuristics*, 13(4):387–401.
- [Müller-Lütken, 2008] Müller-Lütken, J. (2008). Liste von 163 Mastermind-Spielen online. [http://www.onlinespiele-sammlung.de/mastermind/list-of-mastermind-games.php?records\\_to\\_show=-1&list=1](http://www.onlinespiele-sammlung.de/mastermind/list-of-mastermind-games.php?records_to_show=-1&list=1), accessed 2009-07-13.
- [Pendlebury, 2005] Pendlebury, R. (2005). Can you sudoku? *The Mail on Sunday*.
- [Schaefer, 1978] Schaefer, T. J. (1978). The complexity of satisfiability problems. In *STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226, New York, NY, USA. ACM.
- [Stuckman and Zhang, 2006] Stuckman, J. and Zhang, G.-Q. (2006). Mastermind is NP-Complete. Master's thesis, Department of Electrical Engineering and Computer Science, Case Western Reserve University.

- [Wikipedia-Community, 2009a] Wikipedia-Community (2009a). Knotenüberdeckungsproblem. <http://de.wikipedia.org/wiki/Knotenüberdeckungsproblem>, accessed 2009-07-18.
- [Wikipedia-Community, 2009b] Wikipedia-Community (2009b). List of NP-complete problems. [http://en.wikipedia.org/wiki/List\\_of\\_NP-complete\\_problems](http://en.wikipedia.org/wiki/List_of_NP-complete_problems), accessed 2009-07-18.
- [Wikipedia-Community, 2009c] Wikipedia-Community (2009c). Mastermind. <http://de.wikipedia.org/wiki/Mastermind>, accessed 2009-07-13.
- [Yato, 2003] Yato, T. (2003). Complexity and completeness of finding another solution and its application to puzzles. Master's thesis, The University of Tokyo.