Gottfried Wilhelm Leibniz Universität Hannover
Institut für Theoretische Informatik

Die Komplexität von Fillmat

Bachelorarbeit

Jan-Niklas Ballerstein
Matrikelnr. 2880070

Hannover, den 13. April 2022

Erstprüfer: PD Dr. rer. nat. habil. Arne Meier
Zweitprüfer: Prof. Dr. rer. nat. Heribert Vollmer
Betreuer: Timon Barlag, M. Sc

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 13. April 2022

_____

Jan-Niklas Ballerstein

Inhaltsverzeichnis

1. Einleitung

Logikrätsel, die im "Pen-&-Paper"-Stil zu lösen sind, erfreuen sich nicht erst seit der rasanten Verbreitung von Sudokus einer großen Beliebtheit. Unter "Pen-&-Paper", aus dem Englischen wörtlich mit "Stift und Papier" zu übersetzen, versteht man solche Rätsel, die dem Spieler einen Rahmen vorgeben, in dem er die Lösung (auf einem Papier) direkt (mit einem Stift) einträgt. Zu den wahrscheinlich bekanntesten Knobelspielen dieser Art gehören neben Sudoku auch Kakuro und Fillomino [12].

Ein weiteres Mitglied dieser Kategorie ist Fillmat. Das Ziel dieses Puzzles ist es, ein vorgegebenes Feld der Größe $m \times n$ unter bestimmten Regeln, auf die in der Arbeit im Detail eingegangen wird, mit Matten der Größen 1×1, 1×2, 1×3 und 1×4 zu füllen.

Spiele dieser Art werden nicht zuletzt deshalb gerne gespielt, weil ihre Bearbeitung eine geistige Herausforderung darstellt und zum Nachdenken anregt. Im rechnerischen Umfeld hingegen ist es bei der Betrachtung von Schwierigkeit besonders interessant zu untersuchen, ob Lösungen zu einem gegebenen Problem effizient überprüfbar sind. Ist dies der Fall, wird das Problem der Komplexitätsklasse NP (nichtdeterministisch polynomielle Zeit) zugeordnet. Innerhalb dieser Klasse gibt es weiterhin Probleme, die man als NP-schwer bezeichnet. Obwohl eine Lösung zu so einem Problem effizient verifiziert werden kann, kann das Problem selbst nicht effizient gelöst werden. Probleme, die sowohl in NP liegen als auch NP-schwer sind, nennt man NP-vollständig.

Insbesondere bei Puzzlespielen ist auch das "Another Solution Problem", kurz ASP, relevant. Das ASP eines gegebenen Problems P und einer dazugehörigen Lösung L ist es, eine weitere Lösung L' (ungleich L) für P zu finden. Da die Komplexität des ASP eines Problems durchaus von der Komplexität des ursprünglichen Problems abweichen kann [13], müssen diese beiden Komplexitäten unabhängig voneinander untersucht werden.

Wie viele ähnliche "Pen-&-Paper" Spiele erscheinen Fillmat Puzzle regelmäßig in dem japanischen Rätselmagazin Nikoli [7]. Nachdem einige der in diesem Magazin veröffentlichten Rätsel bereits erfolgreich auf NP-Vollständigkeit überprüft wurden, beweisen Akihiro Uejima und Hiroaki Suzuki, dass dies auch für eine leicht abgewandelte Version von Fillmat gilt. Zusätzlich beweisen sie die ASP-Vollständigkeit des Puzzlespiels [11].

In dieser Arbeit wird das Logikrätsel Fillmat komplexitätstheoretisch betrachtet. Dazu wird es für ein besseres Verständnis zunächst ausführlich beschrieben und mit Beispielen illustriert, bevor eine vollständige Instanz von Fillmat beispielhaft Schritt für Schritt anschaulich gelöst wird. Um einen besseren Überblick über das Thema der Komplexitätstheorie zu erhalten,

werden danach relevante Begriffe wie die Komplexitätsklassen P und NP sowie NP-Schwere und NP-Vollständigkeit definiert und erklärt, wie man die Mitgliedschaft verschiedener Probleme in diesen Klassen beweist. Zudem wird das Another Solution Problem erläutert und mithilfe eines Papers von Takayuki Yato und Takahiro Seta [13] formal definiert. Schließlich wird der Beweis von Uejima und Suzuki, dass ihre Version von Fillmat tatsächlich NP- und ASP-vollständig ist, nachvollzogen und grafisch veranschaulicht.

2. Fillmat

2.1. Hintergründe und Regeln

Bei Fillmat handelt es sich um ein Logikrätsel, das im Pen-&-Paper-Stil zu lösen ist. Die ersten Puzzlespiele dieser Art veröffentlichte das japanische Magazin Nikoli im April 1993 unter dem Namen Firumatto (japanisch フィルマット), wörtlich mit „(mit) Matten füllen“ zu übersetzen [7, 12] – daraus entstand der im europäischen Bereich gebräuchliche Name „Fillmat“. Dieser Name beschreibt das Rätsel insofern zutreffend, als es in der Tat das Ziel des Spiels ist, ein vorgegebenes Spielfeld der Größe $m \times n$ komplett mit „Matten“ der Größen 1×1, 1×2, 1×3 und 1×4 zu füllen. Die Bezeichnung $1 \times x$ gibt im Folgenden immer nur die Größe einer Matte an, nicht jedoch deren Rotation. Ist also zum Beispiel von einer 1×4 Matte die Rede, kann diese sowohl horizontal als auch vertikal verlaufen.

3		3			4
		1			
	4		4		
1		2		1	
		3		2	

3		3			4
		1			
	4		4		
1		2		1	
		3		2	

Abbildung 2.1.: Ein Fillmat Puzzle der Größe 6×5 mit dazugehöriger Lösung

In Abbildung 2.1 ist links ein Spielfeld, welches einige vorgegebene Zahlen enthält, zu sehen. Um das Rätsel zu lösen, muss dieses Spielfeld mit Matten genau so gefüllt werden, dass folgende Regeln [7] erfüllt werden:

1. Die Matten dürfen nur an den vorgegebenen Rasterlinien des Spielfeldes verlaufen.
2. Jede Matte muss entweder die Größe 1×1, 1×2, 1×3 oder 1×4 haben.
3. Eine Matte darf genau eine oder keine vorgegebene Zahl enthalten, nicht mehrere.

4. Enthält eine Matte eine vorgegebene Zahl x, muss diese Matte die Größe $1 \times x$ haben.

5. Matten der gleichen Größe dürfen sich weder vertikal noch horizontal berühren.

6. Ein Rasterpunkt darf nicht von vier verschiedenen Matten berührt werden.

Um die Regeln am besten verständlich zu machen, zeigt Abbildung 2.2 grafisch einige Variationen von Regelbrüchen:

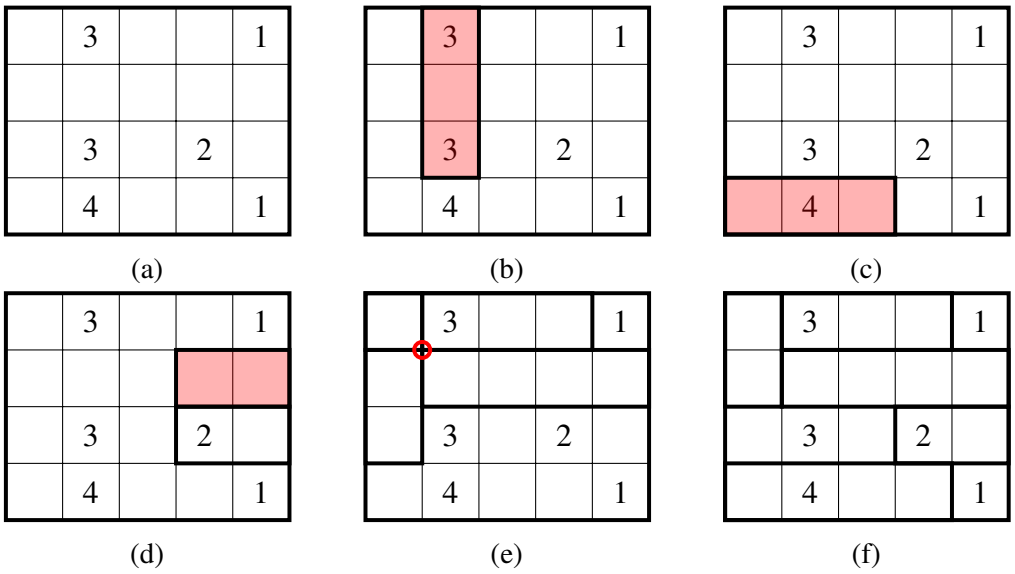


Abbildung 2.2.: Ein Fillmat Puzzle der Größe 5×4 mit Regelbrüchen und Lösung

Dabei sind folgende Regelbrüche abgebildet:

(a) zeigt ein vorgegebenes Fillmat Puzzle

(b) bricht Regel 3, da die Matte mehr als nur eine vorgegebene Zahl enthält

(c) bricht Regel 4, da die Matte eine 4 enthält, aber nicht die Größe 1×4 hat

(d) bricht Regel 5, da sich zwei Matten der gleichen Größe (1×2) berühren

(e) bricht Regel 6, da der gekennzeichnete Rasterpunkt von vier Matten berührt wird

(f) zeigt die korrekte Lösung des Puzzles

2.2. Lösungsweg eines Fillmat Puzzles

Ein gut konstruiertes Fillmat Puzzle hat, wie alle Logikrätsel, nur eine einzige gültige Lösung [1]. Der Lösungsweg ähnelt dem eines Sudokus; der Spieler sucht nach Matten, die nur eine mögliche gültige Platzierung haben und platziert diese. Dies wiederholt man, bis das gesamte Spielfeld gefüllt ist. Um diesen Vorgang besser zu verstehen, wird im Folgenden der Lösungsweg eines beispielhaften Puzzles nachvollzogen. Es wird folgendes Fillmat Puzzle der Größe 5×4 betrachtet:

	3			
1				
2		1		2
			3	

Für eine gute Veranschaulichung werden mögliche Platzierungen gelb, die tatsächlichen Platzierungen des jeweiligen Schrittes grün und Regelverstöße rot markiert. Des Weiteren werden die einzelnen Felder des Spielfeldes für eine klare Zuordnung mithilfe eines Koordinatensystems folgendermaßen bezeichnet:

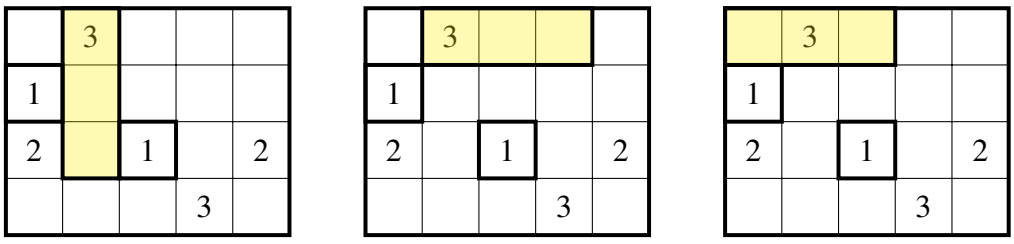
(1,4)	(2,4)	(3,4)	(4,4)	(5,4)
(1,3)	(2,3)	(3,3)	(4,3)	(5,3)
(1,2)	(2,2)	(3,2)	(4,2)	(5,2)
(1,1)	(2,1)	(3,1)	(4,1)	(5,1)

Betrachtet wird nun erneut das vorgegebene Puzzle; alle Felder, die eine 1 enthalten, müssen zwangsläufig 1×1 Matten sein. Diese können dementsprechend platziert werden.

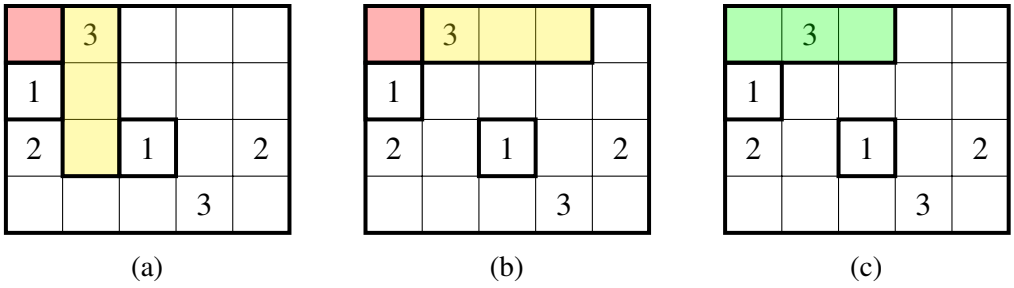
	3			
1				
2		1		2
			3	

Nun wird die vorgegebene 3 in Feld $(2, 4)$ betrachtet. Es gibt drei mögliche Platzierungen für die 1×3 Matte:

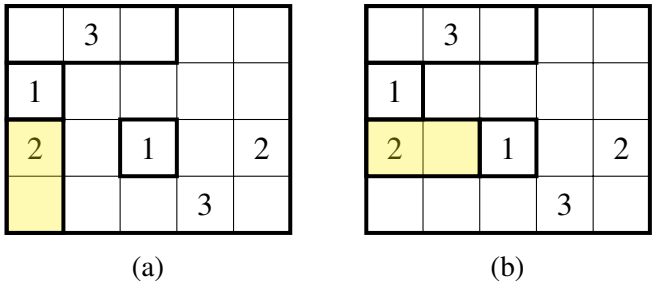
Bei (a) und (b) tritt jedoch ein Problem auf; in beiden Fällen wäre das Feld $(1, 4)$ abgetrennt, müsste also zu einer 1×1 Matte werden. Dies würde jedoch gegen die Regel 5 verstoßen, da sie die 1×1 Matte bei $(1, 3)$ berühren würde. Möglichkeit (c) muss also die richtige



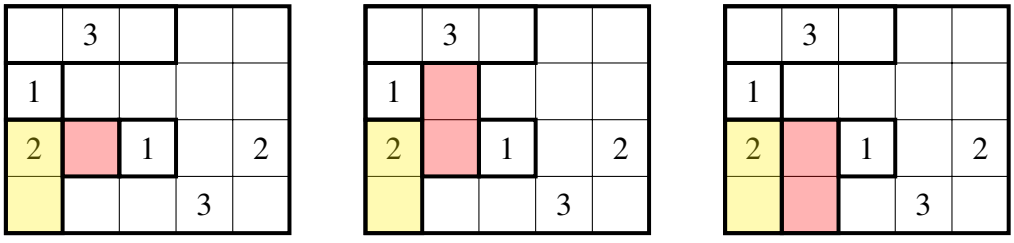
Platzierung sein.

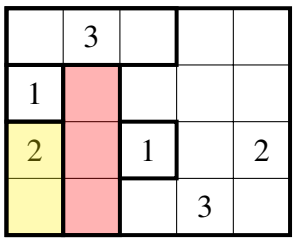


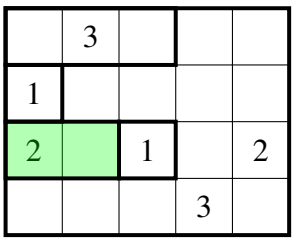
Als nächstens wird das Feld $(1, 2)$ betrachtet, welches aufgrund der vorgegebenen 2 eine 1×2 Matte enthalten muss. Dafür gibt es zwei Möglichkeiten:



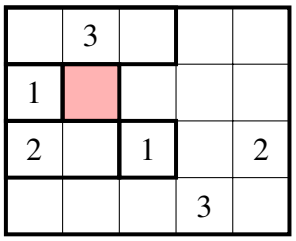
Ausgegangen davon, dass Möglichkeit (*a*) die richtige ist, wird nun Feld $(2, 2)$ betrachtet. Es kann keine 1×1 Matte enthalten, da es die 1×1 Matte bei $(3, 2)$ berühren würde. Auch eine 1×2 Matte kommt nicht infrage, da eine solche an die soeben platzierte 1×2 Matte angrenzen würde. Eine 1×3 Matte ist auch nicht möglich, da die einzige Platzierung einer 1×3 Matte an dieser Stelle die 1×3 Matte bei $(2, 4)$ berühren würde. Für eine 1×4 Matte ist an dieser Stelle kein Platz. Es gibt also bei Möglichkeit (*a*) keine gültige Mattenplatzierung für das Feld $(2, 2)$. Folglich muss Platzierung (*b*) die richtige sein.

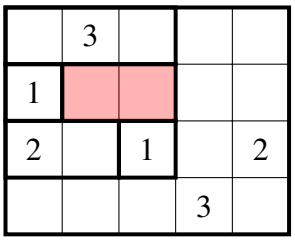


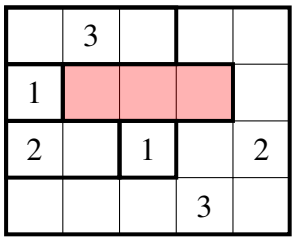


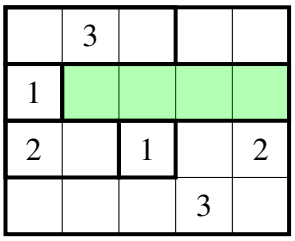


Nun wird das Feld $(2, 3)$ betrachtet. Es kann keine 1×1, 1×2 oder 1×3 Matte enthalten, ohne dass diese eine Matte der gleichen Größe berührt; es muss sich also eine 1×4 Matte handeln, und für eine solche gibt es nur eine mögliche Platzierung:

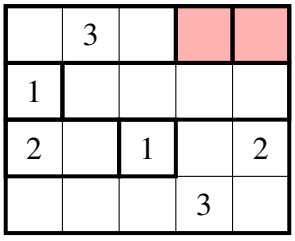


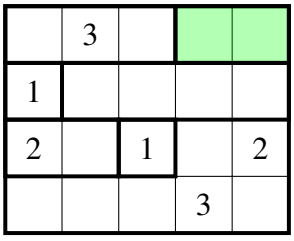




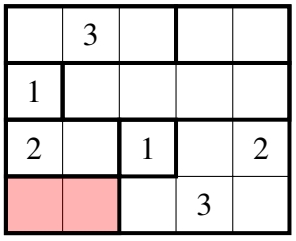


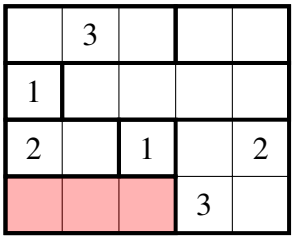
Die 1×2 Matte bei $(4, 4)$ und $(5, 4)$ ergibt sich, weil die beiden Felder keine anliegenden 1×1 Matten sein dürfen.

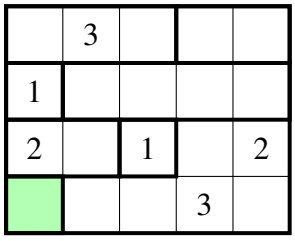




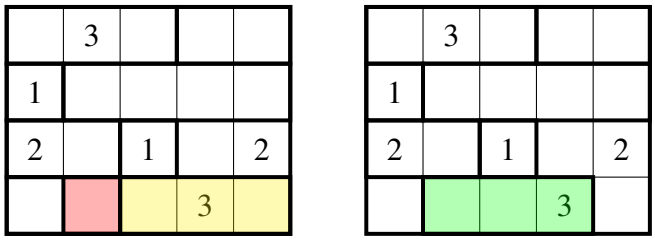
Als Nächstes wird Feld $(1, 1)$ betrachtet: Eine 1×2 Matte ist wegen der angrenzenden 1×2 Matte nicht erlaubt und eine 1×3 Matte würde die vorgegebene 3 bei Feld $(1, 4)$ unmöglich machen. Für eine 1×4 Matte ist der Platz nicht ausreichend, also muss die Matte bei Feld $(1, 1)$ die Größe 1×1 haben.



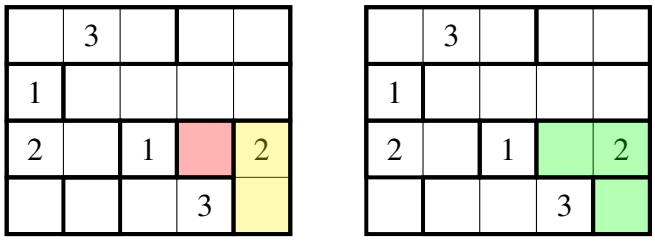




Es gibt nun zwei Möglichkeiten, eine 1×3 Matte über die vorgegebene 3 in Feld $(4, 1)$ zu platzieren; platziert man sie aber auch über Feld $(5, 1)$, müsste Feld $(2, 1)$ eine 1×1 Matte werden, und diese würde die soeben platzierte 1×1 Matte berühren. Daher muss die 1×3-Matte stattdessen Feld $(2, 1)$ mit einschließen.



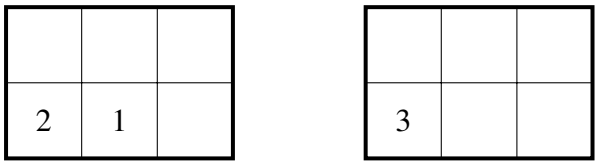
Ähnlich zum vorherigen Schritt muss die 1×2 Matte über Feld $(5, 2)$ so platziert werden, dass die entstehende 1×1 Matte nicht die 1×1 Matte bei Feld $(3, 2)$ berührt.



Da nun alle Felder mit Matten gefüllt sind, ist das Rätsel gelöst.

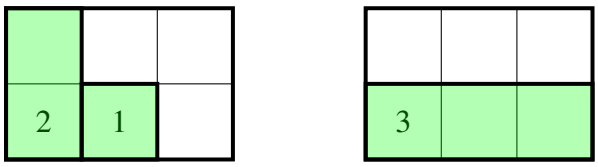
2.3. Beispiele für unlösbare Puzzle

Fillmat Puzzles, die in Rätselheften gefunden werden können, werden von „Rätselmachern“ genau so konstruiert, dass es nur eine einzige Lösung gibt und diese auch zu finden ist [1]. Im Gegensatz können natürlich auch Felder erschaffen werden, die entweder keine oder mehrere gültige Lösungen haben. Dazu werden folgende Spielfelder betrachtet:



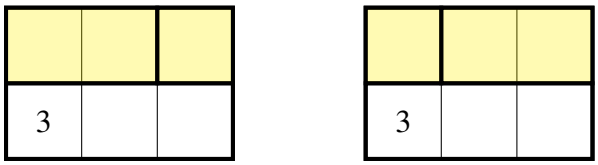
(a) (b)

Es lässt sich schnell erkennen, dass sowohl bei Spielfeld (*a*) als auch bei Spielfeld (*b*) die ersten Schritte eindeutig sind:



(a) (b)

Nun ergibt sich allerdings ein Problem. Bei (a) kann bei Feld $(2, 2)$ keine Matte platzieret werden, ohne eine Regel zu brechen. Es gibt demnach keine gültige Lösung. Bei (b) hingegen ist offensichtlich, dass das Rätsel durch die Platzierung einer 1×1 und einer 1×2 Matte gültig gelöst werden kann. Es gibt allerdings zwei Möglichkeiten, diese beiden Matten zu platzieren:



Keine der beiden Möglichkeiten können ausgeschlossen werden; das Rätsel hat mehrere gültige Lösungen.

Ob Probleme wie das Rätsel Fillmat (effizient) lösbar oder überprüfbar sind, wird in der Komplexitätstheorie untersucht, auf dessen Grundlagen im Folgenden eingegangen wird.

2.4. Puzzle zur Selbstbearbeitung

Nachfolgend sind zwei zusätzliche Instanzen von Fillmat abgebildet, die mithilfe der vorherigen Beispiele und oben beschriebenen Regeln gelöst werden können. Abbildung 2.16 zeigt ein Puzzle moderater Schwierigkeit, wohingegen das in Abbildung 2.17 dargestellte Rätsel einen höheren Schwierigkeitsgrad hat. Die Lösungen zu den Spielfeldern befinden sich im Appendix.

	1				
2		2		1	
				2	
2		3			
				1	4

Abbildung 2.16.

2					
					2
			3	4	1
		1			

Abbildung 2.17.

3. Grundlagen der Komplexitätstheorie

3.1. Komplexitätsklassen

Wenn man von der Komplexität eines Problems spricht, fällt es oft schwer, diese mit der eines anderen Problems zu vergleichen und zu entscheiden, welches Problem „schwieriger“ ist. In der Komplexitätstheorie wurden für diesen Zweck Komplexitätsklassen eingeführt, um ähnlich komplexe Probleme zu gruppieren. Bei der Definition dieser Klassen wird sich an den Ausführungen von Arne Meier und Heribert Vollmer [6] orientiert. Die wahrscheinlich wichtigste dieser Klassen ist die Klasse P, die genau jene Probleme beinhaltet, für deren Lösung es einen effizienten Algorithmus gibt. Effizient bedeutet in diesem Zusammenhang, dass der Algorithmus eine polynomielle Laufzeit hat.

Definition 3.1. *Die Laufzeit $t_A(n)$, $n \in \mathbb{N}$ eines Algorithmus A ist* polynomiell*, wenn es ein $x \in \mathbb{N}$ gibt, sodass $t_A(n)$ durch $O(n^x)$ beschränkt wird.*

Allgemein lässt sich sagen:

Definition 3.2. *Die* Komplexitätsklasse P *ist die Klasse der effizient lösbaren Probleme.*

Für viele Probleme wurde hingegen noch kein effizienter Lösungsalgorithmus gefunden. Hat man allerdings eine vorgegebene Lösung zu einem Problem, kann man diese Lösung auf ihre Korrektheit prüfen. Die Klasse NP beinhaltet dabei all jene Probleme, für deren Überprüfung es einen Algorithmus mit polynomieller Laufzeit gibt. Allgemein schreiben Meier und Vollmer:

Definition 3.3. *Die Komplexitätsklasse NP ist die Klasse der effizient überprüfbaren Probleme.*

P ist eine Teilmenge von NP; Alle Probleme, die effizient lösbar sind, sind auch effizient überprüfbar. Ob auch das Gegenteil wahr ist, das heißt, ob alle effizient überprüfbaren Probleme auch effizient lösbar sind, und somit P = NP gilt, bleibt eines der wichtigsten ungelösten Probleme der theoretischen Informatik [9].

3.2. Reduzierbarkeit und NP-Vollständigkeit

Innerhalb der Klasse NP gibt es weiterhin Probleme, die besonders schwierig sind; mindestens so schwer, wie jedes andere Problem in NP. Um die Schwierigkeit von diesen Problemen zu vergleichen, werden Reduktionen benutzt.

Definition 3.4. *Seien $A \subseteq \Sigma^*, B \subseteq \Delta^*$ zwei Sprachen. A heißt auf B in* Polynomialzeit m-reduzierbar*, falls es eine Funktion $f : \Sigma^* \to \Delta^*$ gibt, so dass gilt: $x \in A$ gdw. $f(x) \in B$ für alle $x \in \Sigma^*$ und f ist in Polynomialzeit berechenbar. Dabei ist f eine (Polynomialzeit-)Reduktion von A auf B. Wir schreiben $A \leq_m^P B$.*

Mithilfe von Reduktionen kann ein Problem A also durch ein anderes Problem B beschrieben werden. Weiterhin folgt intuitiv, dass Problem B nicht schwieriger zu lösen sein kann als Problem A. Diejenigen Probleme, die mindestens so schwierig wie jedes andere Problem in NP zu lösen sind, nennt man NP-schwer. Ist ein solches Problem zusätzlich selbst effizient überprüfbar, nennt man es NP-vollständig.

Definition 3.5. *Ein Problem A ist* NP-schwer*, falls für alle $B \in NP$ gilt: $B \leq_m^P A$.*

Definition 3.6. *Ein Problem A ist* NP-vollständig*, falls A NP-schwer ist und $A \in NP$.*

Um für ein beliebiges Problem A NP-Vollständigkeit zu beweisen, muss also sowohl gezeigt werden, dass A in NP liegt, als auch, dass es sich bei A um ein NP-schweres Problem handelt. Während polynomielle Überprüfbarkeit oft trivial ist, muss für die NP-Schwere ein geeignetes als NP-schwer bekanntes Problem B gefunden werden, das genug Ähnlichkeiten mit A aufweist, um eine Reduktion von A auf B vorzunehmen.

3.3. Circuit-SAT

Für viele Probleme in der theoretischen Informatik wurde die NP-Vollständigkeit bereits bewiesen. Eines dieser Probleme ist das Erfüllbarkeitsproblem für Schaltkreise, kurz Circuit-SAT aus dem Englischen für Schaltkreis und Erfüllbarkeit (Satisfiability) [8]. Circuit-SAT ist eine Erweiterung des bekannten Erfüllbarkeitsproblems der Aussagenlogik (SAT), welches das erste Problem war, für das NP-Vollständigkeit nachgewiesen wurde [2].

Ähnlich zu dem Problem SAT, welches sich mit der Frage beschäftigt, ob eine aussagenlogische Formel erfüllbar ist, geht es bei Circuit-SAT um die Erfüllbarkeit eines booleschen Schaltkreises, der aus beliebig vielen Und-, Oder- und Nicht-Gattern sowie Eingabe-Gattern und einem Ausgabe-Gatter besteht. Dabei sind alle Werte binär; sie können nur die Wahrheitswerte *wahr* oder *falsch* annehmen, im Folgenden auch mit 1 und 0 bezeichnet.

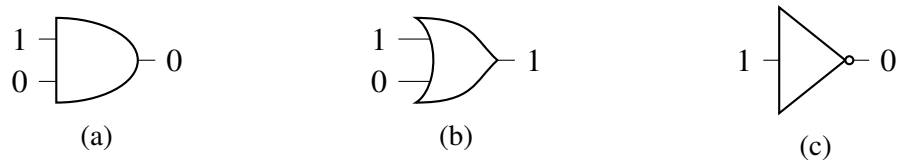


Abbildung 3.1.: Logikgatter eines booleschen Schaltkreises mit Ein- und Ausgabe

(a) ein Und-Gatter mit Eingaben *wahr* und *falsch* und Ausgabe *falsch*

(b) ein Oder-Gatter mit Eingaben *wahr* und *falsch* und Ausgabe *wahr*

(c) ein Nicht-Gatter mit Eingabe *wahr* und negierter Ausgabe *falsch*

Mithilfe dieser einfachen Gatter lassen sich komplette Schaltkreise darstellen. Abbildung 3.2 zeigt den booleschen Schaltkreis für die aussagenlogische Formel $f = (x \wedge \bar{y}) \vee (\bar{z} \wedge y)$.

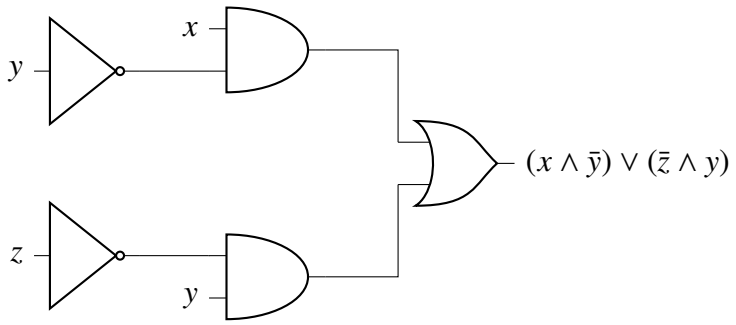


Abbildung 3.2.: Ein boolescher Schaltkreis

Damit die Formel f erfüllt wird, müssen die Eingaben x, y und z entsprechend gewählt werden. In diesem Beispiel sind gültige Eingabewerte unter anderem $x = 1$, $y = 1$ und $z = 0$.

4. Another Solution Problem

4.1. Einleitung

Bei einigen Problemen ist es nicht nur interessant, eine einzige Lösung zu finden, sondern auch zu überprüfen, ob es neben einer bekannten Lösung L weitere Lösungen $L' \neq L$ zu diesem Problem gibt. Insbesondere bei der Erschaffung von Puzzlespielen wie Sudoku oder Fillmat ist es wichtig, dass gerade dies nicht der Fall ist [1]; bei einem gut konstruierten Rätsel soll die Lösung einzigartig sein. Die Betrachtung dieses Problems haben Nobuhisa Ueda und Tadaaki Nagao 1996 unter dem Namen „Another Solution Problem (ASP)“ bekannt gemacht [10], bevor Takayuki Yato und Takahiro Seta die Theorie von ASP 2003 formal beschreiben [13].

Neben dem offensichtlichen Nutzen in Puzzlespielen lässt sich das Prinzip von ASP auch in weiteren Gebieten anwenden. Yato und Seta nennen als Beispiel etwa die Restauration historischer Materialien wie Fossilien, bei der es wichtig ist, dass das restaurierte Ergebnis tatsächlich ohne Zweifel einzig auf das ursprüngliche Material zurückzuführen ist.

4.2. Formale Definition von Funktionsproblemen

Yato und Seta liefern eine vollständige formale Definition des Another Solution Problems [13]. Dazu benutzen sie folgende Definitionen von Funktionsproblemen:

Definition 4.1. *Sei* Π *ein Tripel* (D, S, σ)*, wobei gilt:*

- D *ist die Menge der Instanzen eines Problems*
- S *ist die Menge aller Lösungen zu allen Instanzen*
- σ *ist eine Abbildung von* D *zu* 2^S*. Für eine Instanz* $x \in D$ *nennt man* $\sigma(x)$ *die* Lösungsmenge *zu* x *und ein Element aus* $\sigma(x)$ *eine* Lösung *von* x*.*

Dann nennt man ein Problem, das eine Lösung zu einer Instanz $x \in D$ *(oder einfach* Π *selbst) findet, ein* Funktionsproblem*.*

Mit dieser Formalisierung wird die Klasse FNP folgendermaßen beschrieben:

Definition 4.2. *FNP ist eine aus Funktionsproblemen* $\Pi = (D, S, \sigma)$ *bestehende Klasse. Für alle Probleme* $\in$ *FNP gilt:*

- *Es existiert ein Polynom p, sodass für alle* $x \in D$ *und* $s \in \sigma(x)$ *gilt:* $|s| \leq p(|x|)$.
- *Die Aussage* $y \in \sigma(x)$ *kann für alle* $x \in D$ *und* $y \in S$ *in polynomieller Zeit entschieden werden.*

Definition 4.3. *Sei* $\Pi = (D, S, \sigma)$ *ein Funktionsproblem und sei* $Y = \{x \in D \mid \sigma(x) \neq \emptyset\}$. *Dann nennt man* $\Pi_d = (D, Y)$. *(Das Paar besteht also aus der Menge aller Instanzen und der Menge aller ja-Instanzen) das* durch Π erzeugte Entscheidungsproblem *oder einfach das Entscheidungsproblem von* Π.

Aus der Charakteristik von NP folgt weiterhin:

Satz 4.4. *Für jedes Entscheidungsproblem* $\hat{\Pi} \in$ *NP existiert ein Funktionsproblem* $\Pi \in$ *FNP, sodass* $\Pi_d = \hat{\Pi}$. *(Dabei impliziert* $\Pi \in$ *FNP natürlich* $\Pi_d \in$ *NP.)*

4.3. Formale Definition von ASP

Mithilfe der im vorherigen Kapitel beschriebenen Funktionsprobleme definieren Yato und Seta ASPs wie folgt [13]:

Definition 4.5. *Sei* $\Pi = (D, S, \sigma)$ *ein Funktionsproblem. Für* Π *und ein* $n \in \mathbb{N}$ *wird das Funktionsproblem* $\Pi_{[n]} = (D_{[n]}, S, \sigma_{[n]})$*, das wie folgt konstruiert wird, das* n-Another Solution Problem *von* Π *(oder kurz* n-ASP*) genannt.*

$D_{[n]} = \{(x, S_x) \mid S_x \subseteq \sigma(x), |S_x| = n\}$,
$\sigma_{[n]}(x, S_x) = \sigma(x) - S_x$

Das Entscheidungsproblem eines n-Another Solution Problems wird *n-Another Solution Entscheidungsproblem* genannt.

Als nächstes wird die ASP-Polynomialzeitreduktion definiert. Yato und Seta weisen darauf hin, dass es sich dabei um eine von Ueda und Nagao eingeführte „anzahlerhaltende“ Reduktion handelt [10], die Polynomialzeit-getreue Umwandlungen von Lösungen ermöglicht, da die Anzahl der Lösungen gleich bleibt.

Definition 4.6. *Seien* $\Pi_1 = (D_1, S_1, \sigma_1)$ *und* $\Pi_2 = (D_2, S_2, \sigma_2)$ *Funktionsprobleme. Wir nennen das Paar* $\varphi = (\varphi_D, \varphi_S)$ *eine* ASP-Polynomialzeitreduktion *von* Π_1 *auf* Π_2*, wenn gilt:*

- φ_D *ist eine in Polynomialzeit berechenbare Abbildung von* D_1 *auf* D_2.
- *Für alle* $x \in D_1$ *ist* φ_S *eine in Polynomialzeit berechenbare Bijektion von* $\sigma_1(x)$ *auf* $\sigma_2(\varphi_D(x))$.

Wenn es eine ASP-Polynomialzeitreduktion von Π_1 *auf* Π_2 *gibt, nennt man* Π_1 ASP- polynomialzeitreduzierbar *auf* Π_2 *(auch* $\Pi_1 \preccurlyeq_{ASP} \Pi_2$*).*

Die Relation $\preccurlyeq_{ASP}$ ist, genau wie andere Reduktionen, transitiv. Außerdem gilt, dass wenn $\Pi_1 \preccurlyeq_{ASP} \Pi_2$, dann ist Π_{1d} auf Π_{2d} (als Entscheidungsproblem) polynomialzeit-m-reduzierbar.

Satz 4.7. *Seien* Π_1 *und* Π_2 *Funktionsprobleme. Wenn gilt:* $\Pi_1 \preccurlyeq_{ASP} \Pi_2$*, dann gilt auch* $\Pi_{1[n]} \preccurlyeq_{ASP} \Pi_{2[n]}$ *für alle* $n \in \mathbb{N}$.

Beweis. Folgt direkt aus der Definition. □

Satz 4.8. *Für alle Funktionsprobleme* Π *und* $m, n \in \mathbb{N}$ *gilt:*
$(\Pi_{[m]})_{[n]} \preccurlyeq_{ASP} \Pi_{[m+n]}$.

Beweis. Sei $\Pi = (D, S, \sigma)$. Eine Instanz $\tilde{x}$ aus $(\Pi_{[m]})_{[n]}$ hat die Form

$((x, \{s_1, ..., s_m\}), \{t_1, ..., t_n\})$
$(x \in D; s_1, ...s_m, t_1, ..., t_n \in \sigma(x))$.

Dafür wird $\varphi_D(\tilde{x})$ als $((x, \{s_1, ..., s_m\}), \{t_1, ..., t_n\})$ festgelegt. Dann ist die Lösungsmenge von $\tilde{x}$ gleich der Lösungsmenge von $\varphi_D(\tilde{x})$, und somit kann φ_S als Identitätsfunktion festgelegt werden. □

Indem sie die beiden vorangegangen Sätze kombinieren, kommen Yato und Seta auf folgendes wichtige Ergebnis:

Satz 4.9. *Sei* Π *ein Funktionsproblem. Wenn gilt:* $\Pi \preccurlyeq_{ASP} \Pi_{[1]}$*, dann gilt für alle nicht negativen Ganzzahlen n:* $\Pi_{[n]} \preccurlyeq_{ASP} \Pi_{[n+1]}$*. (Deswegen* $\Pi \preccurlyeq_{ASP} \Pi_{[n]}$*).*

Beweis. Aus $\Pi \preccurlyeq_{ASP} \Pi_{[1]}$ und Satz 4.7 folgt $\Pi_{[n]} \preccurlyeq_{ASP} (\Pi_1)_{[n]}$, und aus Satz 4.8 folgt $(\Pi_{[1]})_{[n]} \preccurlyeq_{ASP} \Pi_{[n+1]}$. Also gilt Satz 4.9 wegen der Transitivität von $\preccurlyeq_{ASP}$. □

4.4. ASP-Vollständigkeit

Schließlich definieren Yato und Seta ASP-Vollständigkeit als Vollständigkeit in Bezug auf ASP-Reduktionen folgendermaßen [13]:

Definition 4.10. *Ein Funktionsproblem* Π *ist genau dann* ASP-vollständig*, wenn* $\Pi \in$ *FNP und* $\Pi' \preccurlyeq_{ASP} \Pi$ *für alle* $\Pi' \in$ *FNP.*

Satz 4.11. *Seien* Π *und* Π' *Funktionsprobleme. Wenn* Π *ASP-vollständig ist,* $\Pi' \in$ *FNP und* $\Pi \preccurlyeq_{ASP} \Pi'$*, dann ist* Π' *ASP-vollständig.*

Des Weiteren beweisen Yato und Seta ASP-Vollständigkeit für die NP-Probleme SAT, 3SAT und 1-in-3 SAT sowie für die Puzzlespiele Slither Link, Cross Sum und Number Place [13]. Zuvor haben Ueda und Nagao ASP-Vollständigkeit bereits für das Puzzlespiel Nonogram bewiesen [10].

4.5. Hamiltonkreisproblem

Bei der Betrachtung der Komplexität eines Problems P ist anzumerken, dass das ASP von P durchaus eine andere Komplexität haben kann als P selbst. Ein einfaches Beispiel, das von Yato und Seta erwähnt wird, ist das Hamiltonkreisproblem. Dabei wird der Frage nachgegangen, ob ein Graph einen Pfad erhält, der jeden Knoten des Graphen genau einmal besucht und schließlich wieder bei dem Startknoten endet und somit einen Kreis bildet. Die NP-Vollständigkeit des Problems wurde 1976 von Garey, Johnson und Tarjan bewiesen [3]. Zur Veranschaulichung zeigt die folgende Abbildung einen beispielhaften Graphen mit einem eingezeichneten Hamiltonkreis sowie Beispiele, die keinen Hamiltonkreis bilden.

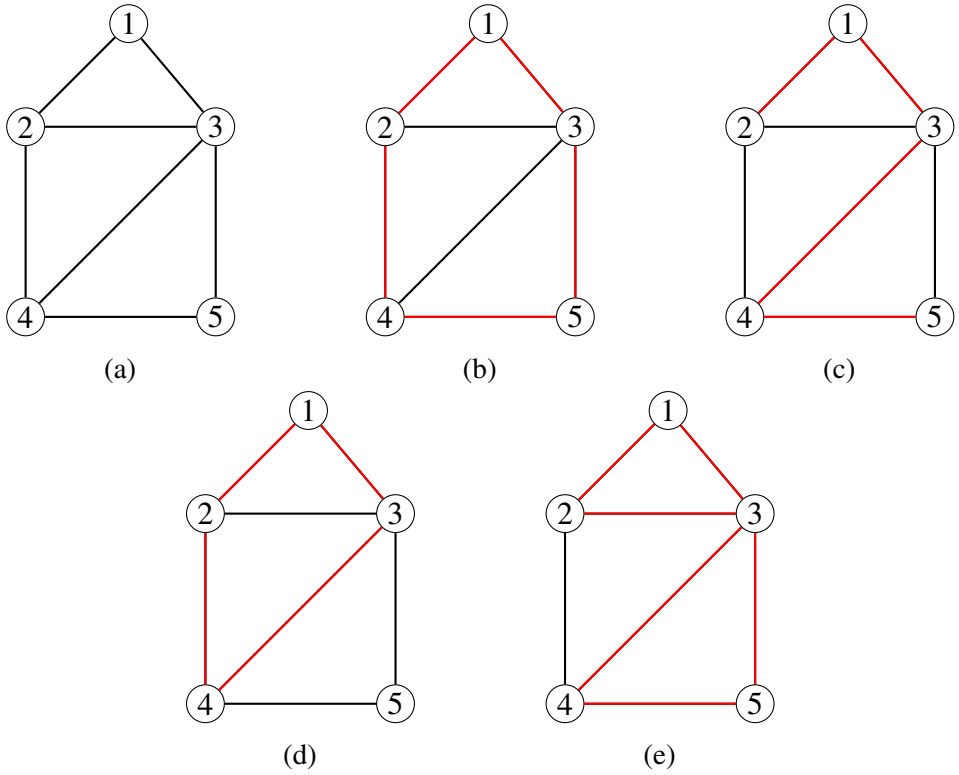


Abbildung 4.1.: Graph mit Hamiltonkreis

(a) zeigt einen vorgegebenen Graphen

(b) zeigt einen Hamiltonkreis im Graphen

(c) besucht zwar alle Knoten des Graphen, endet aber nicht am Startknoten

(d) bildet zwar einen Kreis, besucht aber nicht alle Knoten

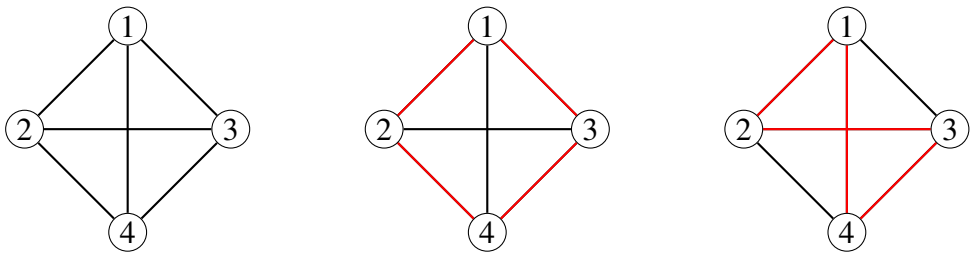
(e) besucht Knoten 3 doppelt

Eine spezielle Form von Graphen sind in der Graphentheorie die kubischen Graphen, dessen Knoten alle den Grad drei besitzen; das heißt, alle Knoten haben genau drei angrenzende

Kanten. Garey, Johnson und Tarjan beweisen, dass das Hamiltonkreisproblem auch auf kubische Graphen beschränkt NP-vollständig bleibt [3]. Weiterhin gilt der Satz von Smith [4]:

Satz 4.12. *Die Anzahl der Hamiltonkreise, die eine gegebene Kante enthalten, ist in kubischen Graphen gerade.*

Daraus ergibt sich, dass ein kubischer Graph, wenn er einen Hamiltonkreis enthält, auch einen weiteren enthalten muss. Dies lässt sich auch grafisch anhand des kleinsten kubischen Graphen sehr leicht erkennen:



(a) (b) (c)

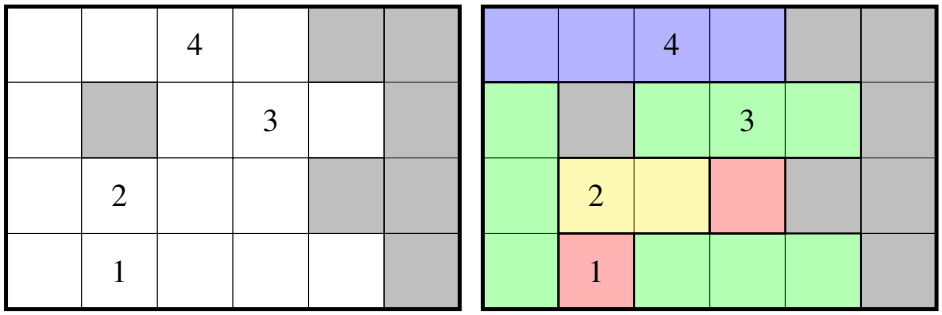
Abbildung 4.2.: Kubischer Graph mit Hamiltonkreisen

Obwohl das Hamiltonkreisproblem für kubische Graphen also NP-vollständig ist, kann es für einzelne Probleminstanzen wie für die in Abbildung 4.2a gezeigte Instanz keine einzigartige, sondern mehrere gültige Lösungen geben. Es ist also nicht ASP-vollständig.

5. Komplexität von Fillmat

5.1. Einleitung und Definitionen

Nachdem die Komplexität verschiedener Puzzlespiele bereits untersucht wurde, beweisen Akihiro Uejima und Hiroaki Suzuki die NP- sowie ASP-Vollständigkeit für Fillmat [11]. Sie wandeln dafür die ursprünglichen vom Rätselheft Nikoli publizierten Regeln des Puzzlespiels [7] leicht ab und ergänzen es um „Löcher“ im Spielfeld, die nicht mit Matten gefüllt werden müssen. Ein Beispiel für ein Spielfeld mit Löchern zeigt Abbildung 5.1a. Löcher werden dabei im Folgenden mit grauen Feldern gekennzeichnet. Des Weiteren wird von der vorherigen Darstellung abgewichen, nach der sichere Mattenplatzierungen grün, mögliche Platzierungen gelb und Widersprüche rot markiert wurden; stattdessen wird die Darstellung von Uejima und Suzuki übernommen, die 1×1 Matten rot, 1×2 Matten gelb, 1×3 Matten grün und 1×4 Matten blau kennzeichnen.



(a) (b)

Abbildung 5.1.: Ein Fillmat Puzzle mit Löchern und dazugehöriger Lösung

Uejima und Suzuki definieren Fillmat als Entscheidungsproblem folgendermaßen:

Definition 5.1. *Eine Instanz von Fillmat ist ein rechteckiges Feld* P *der Größe* $m \times n$ *mit* $m, n \in \mathbb{N}$*, in welchem einige Felder die Zahlen {1,2,3,4} enthalten und welches Löcher enthalten darf. Das Entscheidungsproblem ist die Frage, ob es eine Anordnung von Matten in* F *gibt, die die Regeln von Fillmat einhält.*

5.2. Reduktion

Wie bei vielen Entscheidungsproblemen lässt sich eine potentielle Lösung einer Instanz von Fillmat sehr schnell in Polynomialzeit überprüfen, wodurch die NP-Mitgliedschaft trivial wird. Um hingegen die NP-Schwere zu beweisen, betrachten Akihiro und Hiroaki das im Kapitel 3.3 thematisierte und als NP-vollständig bewiesene Circuit-Sat [8] und reduzieren von diesem auf ihre in Definition 5.1 beschriebene Version von Fillmat. Sie weisen zusätzlich darauf hin, dass Circuit-Sat außerdem ASP-vollständig ist, da es sich bei Cooks Reduktion, mit der dieser die NP-Vollständigkeit von SAT zeigt [2], um eine ASP-Polynomialzeitreduktion (siehe Kapitel 4.3) handelt. Das Another Solution Problem von Fillmat definieren Uejima und Suzuki folgendermaßen:

Definition 5.2. *Gegeben sei eine Instanz* P *von Fillmat und eine dazugehörige Lösung* s. *Finde eine Lösung* s' *(ungleich* s*) zu* P.

5.2.1. Gattertypen

In der Reduktion, die Uejima und Suzuki benutzen, weisen sie die Lösung jeder Instanz von Fillmat einer dazu korrespondierenden Lösung von Circuit-Sat zu, indem sie Module konstruieren, die die logischen Gatter eines booleschen Schaltkreises simulieren. Da es sich dabei um eine eins-zu-eins-Korrespondenz handelt, impliziere dies, das Fillmat, wie Circuit-Sat, ASP-vollständig sei.

Bei den Gattertypen beschränken sich Uejima und Suzuki auf die Basisgatter UND und NICHT sowie auf Hilfsmodule wie Verbindungsmodule und Splitter. Da Fillmat Puzzle nur zweidimensional sind, müssen die korrespondierenden Schaltkreise planar bleiben; das heißt, Leitungen dürfen sich nicht überschneiden. Um dem vorzubeugen, wird auf einen von William F. McColl konstruierten Schaltkreis verwiesen, der eine planare Lösung aus sich überschneidenden Leitungen produziert [5]. XOR Gatter, wie sie zum Beispiel in Abbildung 5.2 zu sehen sind, können dann entsprechend mit UND Gattern und NICHT Gattern planar umgesetzt werden.

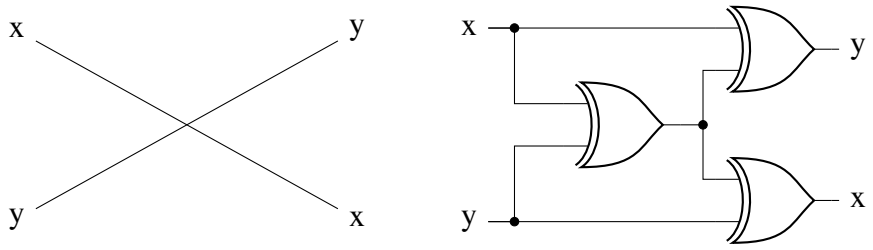


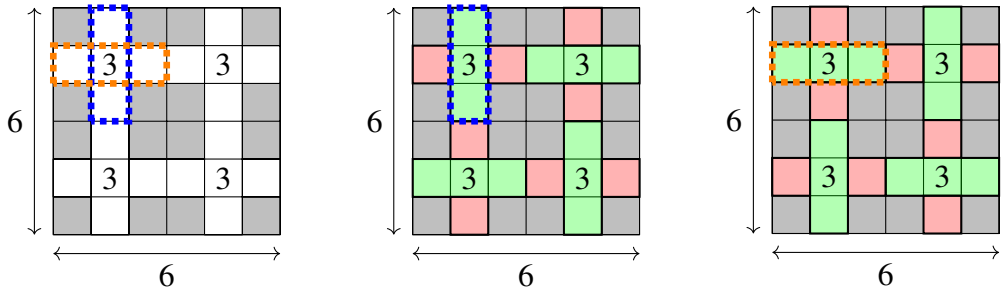
Abbildung 5.2.: Planare Lösung zweier sich überschneidender Leitungen

Beim Entwurf ihrer Module verwenden Uejima und Suzuki Fillmat-Felder der Größe 6×6 als Grundeinheiten und konstruieren die Module für Logikgatter entsprechend so, dass deren

Größe in der Regel ein Vielfaches der Größe einer Grundeinheit entspricht. Die linke Seite eines jeden Moduls empfängt dabei einen booleschen Wert als Eingabe vom links anliegenden Modul und gibt die Ausgabe an der rechten Seite an das nächste Modul weiter.

5.2.2. Eingabegatter und Leitungen

Als Eingabe dient eine einfache, wie in Abbildung 5.3a dargestellte Grundeinheit. Betrachtet man die gekennzeichnete 3, so stellt man fest, dass es für die 1×3 Matte genau zwei gültige Platzierungsmöglichkeiten gibt. Nach Platzierung dieser Matte ergeben sich die anderen Matten im Feld nach den Regeln von Fillmat eindeutig. Jede der beiden entstehenden Möglichkeiten für das Eingabemodul wird nun einem logischen Wahrheitswert zugewiesen. Uejima und Suzuki definieren dabei die in 5.3b abgebildete Anordnung als *wahr* und die in 5.3c abgebildete Anordnung als *falsch*.



(a) (b) Eingabe = *wahr* (c) Eingabe = *falsch*

Abbildung 5.3.: Eingabemodul

Als Verbindung zwischen einzelnen Gattern dienen bei booleschen Schaltkreisen die Leitungen. Diese Funktion wird durch Verbindungsmodule umgesetzt, die wiederum aus Vielfachen von Grundeinheiten bestehen. Sie erhalten den Wahrheitswert des ersten Moduls und geben diesen bis an das anschließende Modul weiter. Ein Beispiel für ein Verbindungsmodul ist in Abbildung 5.4 zu sehen.

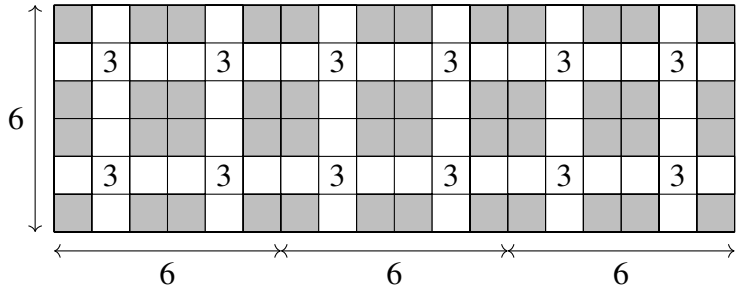


Abbildung 5.4.: Verbindungsmodul

5.2.3. Splitter und Ausgabegatter

Da es in booleschen Schaltkreisen oft zu Verzweigungen kommt, muss eine solche Funktion auch mit den Modulen, die einen solchen simulieren, möglich sein. Uejima und Suzuki nutzen dazu die symmetrischen Eigenschaften ihrer Grundeinheit, um Leitungen nicht nur horizontal, sondern auch vertikal verlaufen zu lassen. An ein einzelnes Verbindungsmodul können außerdem mehrere Module angrenzen, die den eingehenden Wahrheitswert erhalten und nach oben, nach unten und nach rechts weiter verlaufen können. Durch ein Splittermodul, wie in Abbildung 5.5 zu sehen, lassen sich Verzweigungen einfach umsetzen.

Um am Ende des Schaltkreises die Ausgabe zu lesen, braucht man schließlich ein Ausgabemodul. Ein solches zeigt Abbildung 5.6a. Es besteht lediglich aus zwei an das Ende eines Basismoduls angefügten Feldern, an denen man sehen kann, ob die Ausgabe *wahr* oder *falsch* ist; oder in Begriffen des Fillmat Puzzles, ob dieses eine gültige Lösung hat oder nicht. Wie in 2.2 auf Seite 8 erwähnt, können 1×1 Matten bereits zu Beginn des Lösungsvorganges auf alle Einsen im Feld platziert werden. Das lässt bei einer *wahren* Eingabe eine gültige Lösung zu, indem eine 1×2 Matte die letzten leeren Felder bedeckt (Abbildung 5.6b). Bei Eingabe eines *falschen* Eingabewertes kann das Puzzle hingegen nicht gültig gelöst werden, da zwei Matten gleicher Größe nicht aneinander angrenzen dürfen (Abbildung 5.6c).

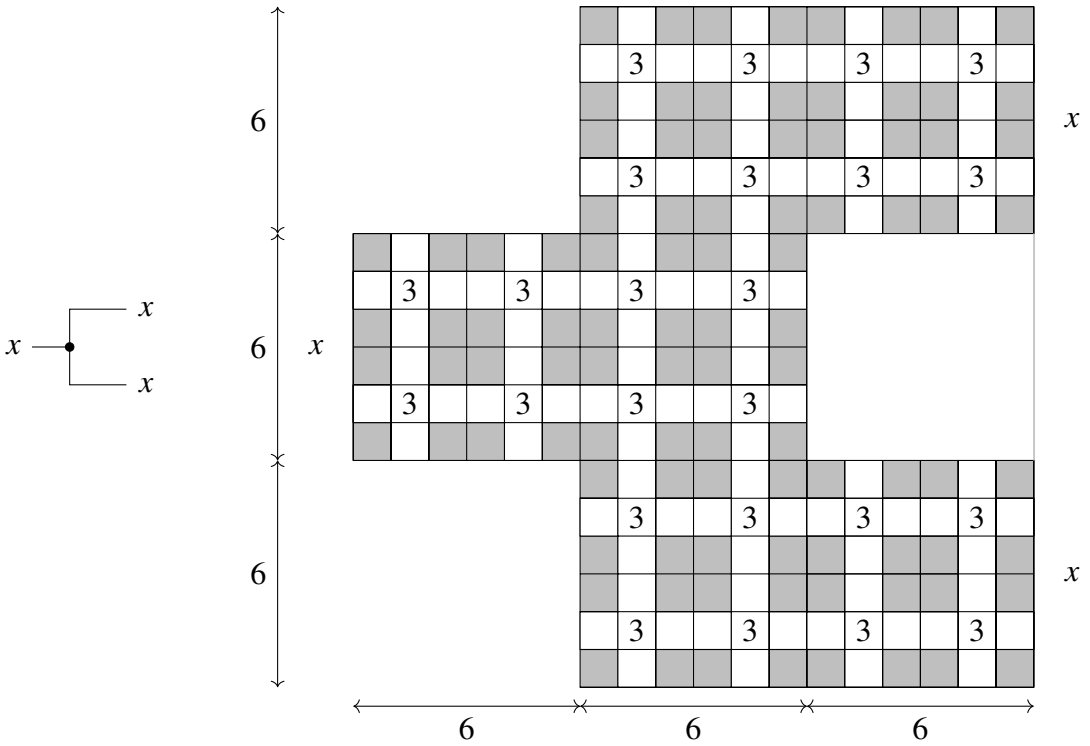
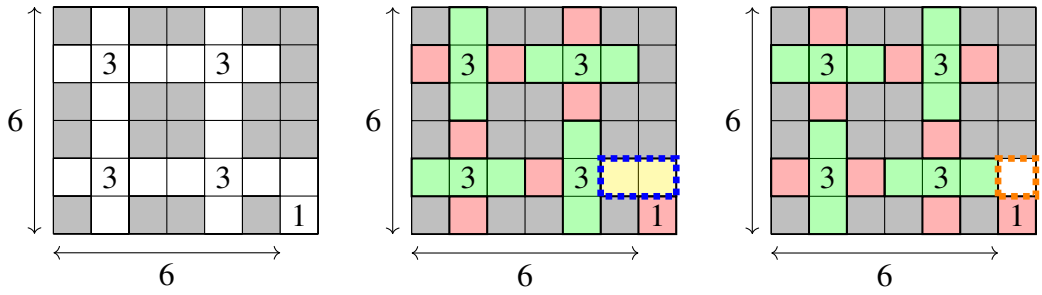


Abbildung 5.5.: Splitter



(a) (b) Ausgabe = *wahr* (c) Ausgabe = *falsch*

Abbildung 5.6.: Ausgabemodul

5.2.4. NICHT Gatter

Nachdem sie die notwendigen Grundlagenmodule etabliert haben, konstruieren Uejima und Suzuki als nächstes das in Abbildung 5.7 gezeigt NICHT Gatter. Die Anordnung der 1×3 und 1×1 Matten ergibt sich parallel zu der der Verbindungsmodulen aus der vom vorherigen Modul übernommenen Eingabe (Abbildung 5.8). Für die folgenden 1×2 Matten gibt es dann wiederum nur eine gültige Platzierung (Abbildung 5.9), da eine von dieser Platzierung abweichende Anordnung gegen die Regel, dass Matten gleicher Größe sich nicht berühren dürfen, verstoßen würde. Die 1×3 Matten zum Ende des Moduls gehen wieder in den symmetrischen Aufbau des Basismoduls über (Abbildung 5.10), um den Schaltkreis einfach mithilfe eines anschließenden Verbindungsmoduls fortführen zu können. Da es für jede Eingabe immer nur jeweils eine gültige Anordnung gibt, simuliert das NICHT Modul ein NICHT Gatter in einem booleschen Schaltkreis eins zu eins.

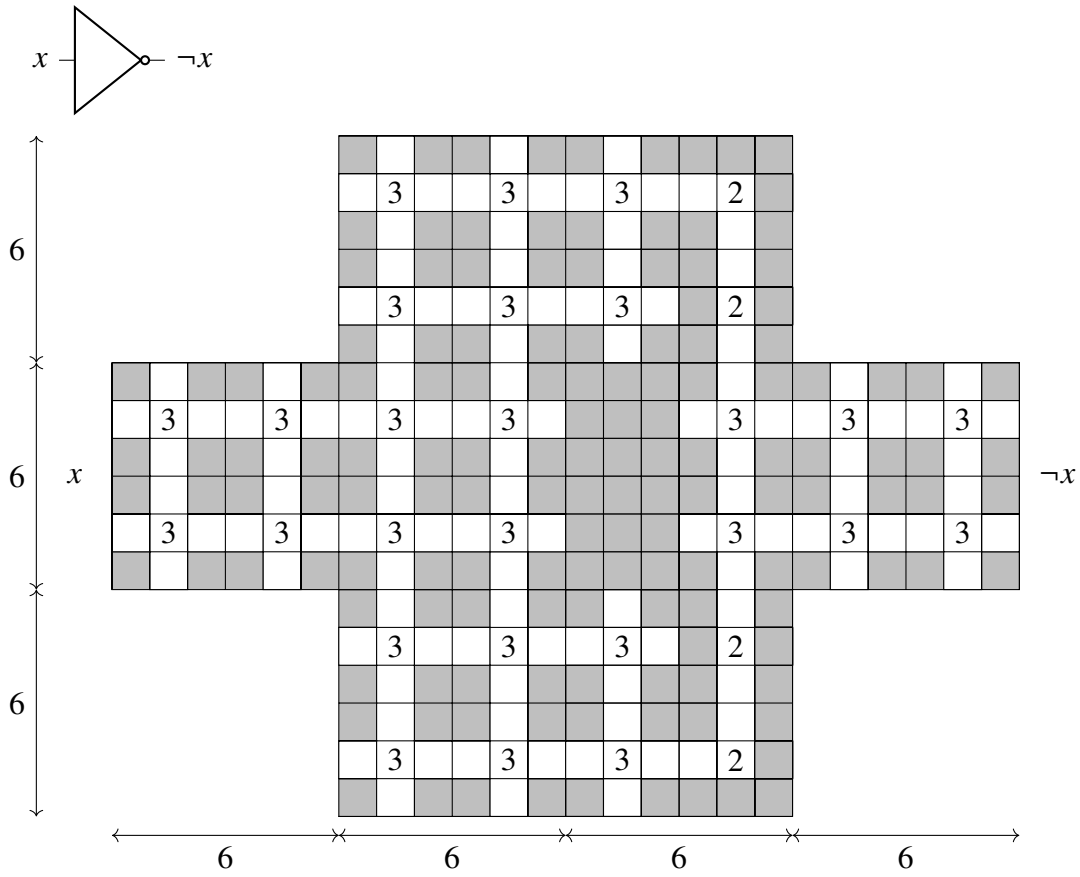
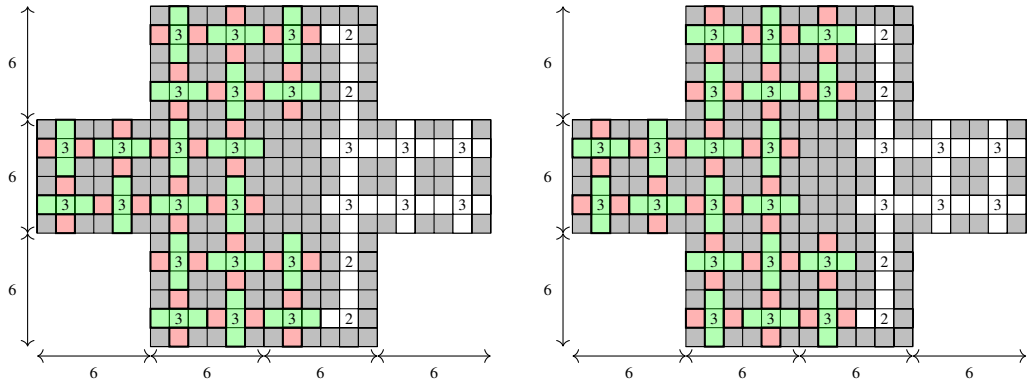


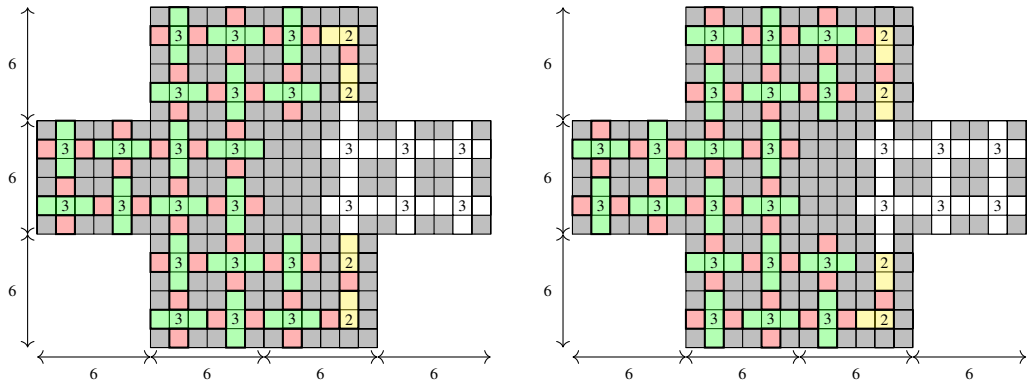
Abbildung 5.7.: NICHT Modul



(a) Eingabe = *wahr*

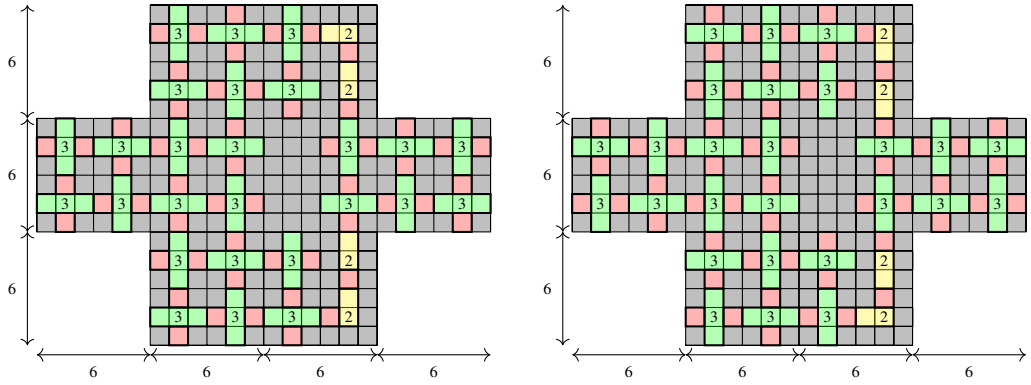
(b) Eingabe = *falsch*

Abbildung 5.8.



(a) Eingabe = *wahr* (b) Eingabe = *falsch*

Abbildung 5.9.



(a) Eingabe = *wahr* (b) Eingabe = *falsch*

Abbildung 5.10.

5.2.5. UND Gatter

Das letzte Gatter aus einem booleschen Schaltkreis, das durch ein Modul simuliert werden muss, ist das UND Gatter ($y = x_1 \wedge x_2$). Die Umsetzung von Uejima und Suzuki wird in Abbildung 5.11 dargestellt. Abhängig von x_1 und x_2 gibt es vier verschiedene Möglichkeiten für die Kombinationen aus Eingabewerten; dementsprechend gilt dies auch für die möglichen Anordnungen in den in der Abbildung markierten Feldern, die einem Verbindungsmodul ähneln. Aufgrund der Symmetrie dieser Module ist es offensichtlich, dass die Ausgabe y den Wert *falsch* annimmt, wenn mindestens einer der Eingabewerte x_1 oder x_2 den Wert *falsch* hat, da eine horizontale 1×3 Matte dadurch auf Feld B beziehungsweise auf Feld C platziert werden würde (siehe Abbildungen 5.15, 5.19 und 5.20) und diese Anordnung bis zur Ausgabe symmetrisch weiter gegeben werden würde.

Die Abbildung 5.12 zeigt den Fall (x_1, x_2) = (*wahr*, *wahr*). Die ersten zu platzierenden 1×3 Matten ergeben sich aus der Symmetrie der Eingabe. Zu beachten sind die 1×3 Matten auf den Feldern B und C, die in diesem Fall beide vertikal platziert sind. Die nächsten Matten

kann man ausgehend von dem blau markierten Feld eindeutig platzieren. Es ergibt sich die Anordnung, wie sie in Abbildung 5.13 zu sehen ist.

Dies wiederum erzwingt die Platzierung der 1×4 Matte auf Feld A und damit auch die der restlichen Matten, wodurch das gesamte Feld eindeutig ausgefüllt werden kann (Abbildung 5.14). Die Anordnung des sich auf der ganz rechten Seite des Moduls befindlichen Basismoduls entspricht damit genau der Anordnung, die den Wert *wahr* an das nächste Modul weitergeben würde. Es folgt also, dass die Ausgabe des UND Moduls dem Wert *wahr* entspricht, wenn beide Eingaben den Wert *wahr* haben.

Wenn die Eingabe des Moduls $(x_1, x_2) = (\mathit{falsch}, \mathit{falsch})$ ist, ist die Anordnung der Matten innerhalb der basismodulähnlichen Abschnitte wie in Abbildung 5.15 gezeigt festgelegt. Die zu platzierenden 1×3 Matten der beiden markierten Feld haben aufgrund der potentiell angrenzenden 1×3 Felder nur jeweils eine gültige Platzierungsmöglichkeit, woraus sich die Anordnung der anderen Matten wie in Abbildung 5.16 ergibt.

Wenn die 1×3 Matte bei dem markierten Feld vertikal verlaufen würde, gäbe es für das links angrenzende Feld keine gültige Mattenplatzierungsmöglichkeit; es könnte weder eine 1×1 noch eine 1×2 Matte platziert werden, ohne dass sich zwei Matten gleicher Größe berühren würden. Die 1×3 Matte muss dementsprechend horizontal verlaufen. Nach der Platzierung dieser und der logisch folgenden Matten entsteht die in Abbildung 5.17 gezeigte Anordnung.

Die 1×4 Matte bei Feld A kann nun nicht mehr vertikal verlaufen, da es sonst keine gültige Mattenplatzierungen für die rechts an A angrenzenden Felder geben würde. Es ergibt sich die in 5.18 dargestellte einzigartige Anordnung. Die Ausgabe des UND Moduls ist bei den Eingabewerten $(x_1, x_2) = (\mathit{falsch}, \mathit{falsch})$ also als $y = \mathit{falsch}$ eindeutig festgelegt.

Die beiden verbleibenden Kombinationen von Eingabewerten lassen sich parallel zu den oben detaillierter beschriebenen Lösungsvorgängen herleiten. Auch hier ergeben sich die symmetrischen Abschnitte oben und unten im UND Modul logisch durch die Eingabemodule. Der Mittelteil des Moduls folgt eindeutig durch das Herleiten der einzigen gültigen Lösung. Der Fall $(x_1, x_2) = (\mathit{wahr}, \mathit{falsch})$ ist in Abbildung 5.19, der Fall $(x_1, x_2) = (\mathit{falsch}, \mathit{wahr})$ in Abbildung 5.20 dargestellt.

Damit zeigen Uejima und Suzuki, dass das von ihnen konstruierte UND Modul bei allen möglichen Eingaben die gleichen Ausgabewerte hat wie ein UND Gatter in einem booleschen Schaltkreis bei entsprechenden Eingabewerten. Außerdem gilt, dass die jeweilige Anordnung von Matten für jeden Fall von Eingabewerten eindeutig ist und es keine andere Möglichkeit gibt, die Matten mit gleichem Ergebnis anders zu platzieren. Daraus folgt, dass das UND Modul ein UND Gatter eins zu eins simuliert.

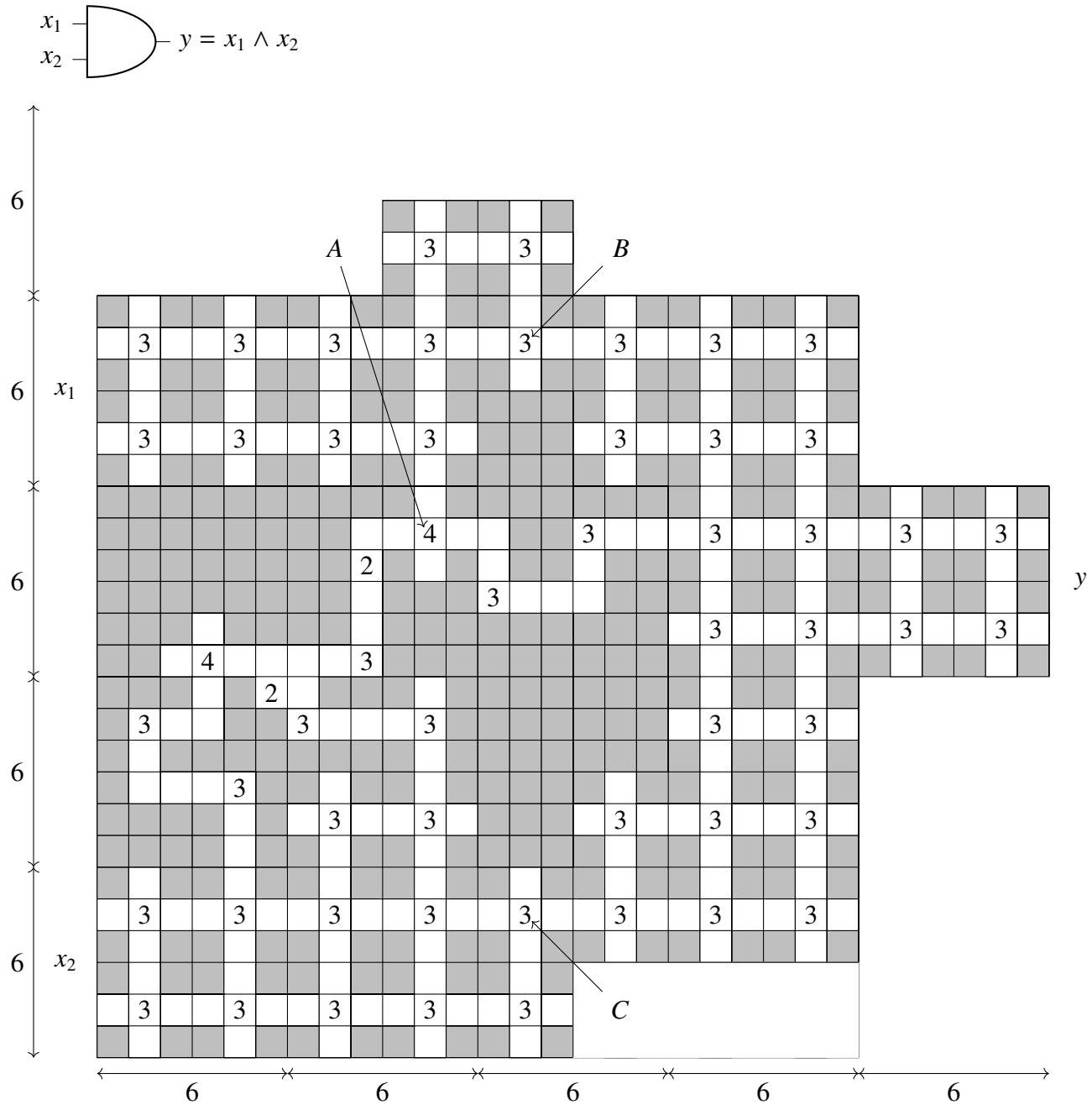


Abbildung 5.11.: UND Modul

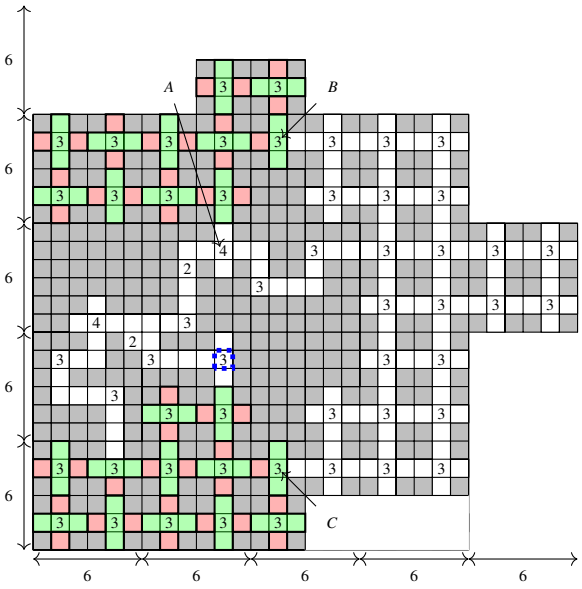


Abbildung 5.12.

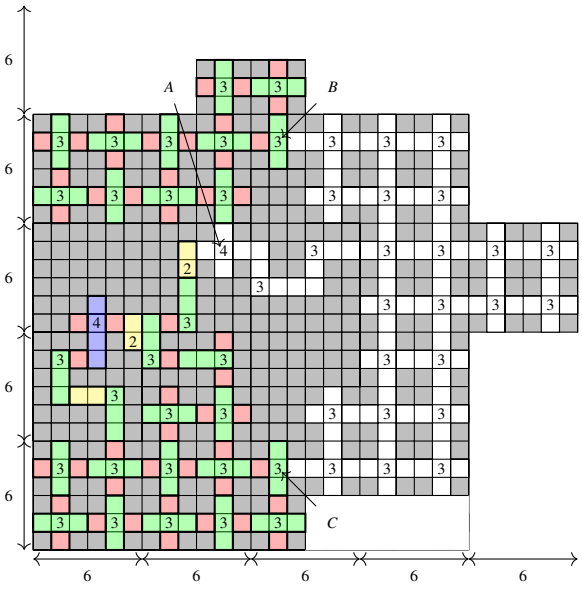


Abbildung 5.13.

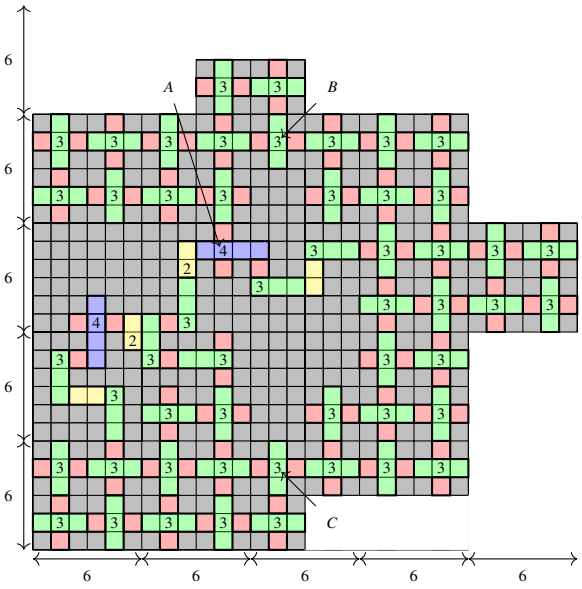


Abbildung 5.14.: Lösung eines UND Moduls mit Eingabe $(x_1, x_2) = (wahr, wahr)$

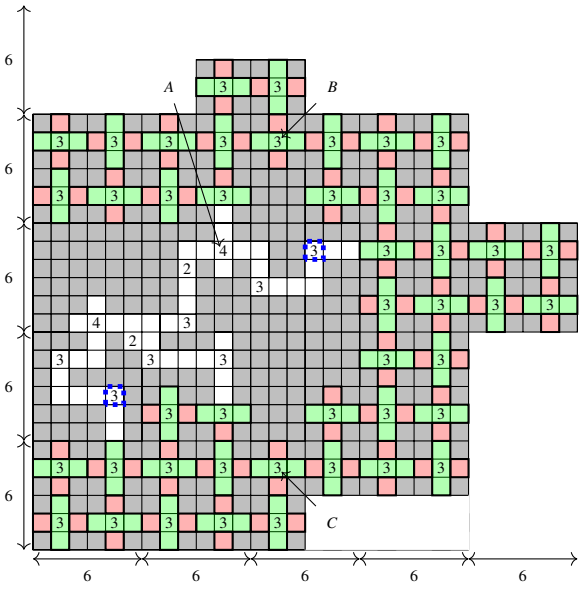


Abbildung 5.15.

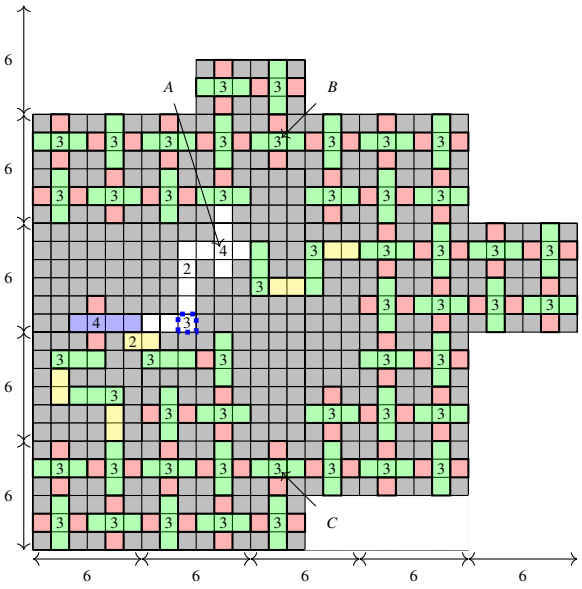


Abbildung 5.16.

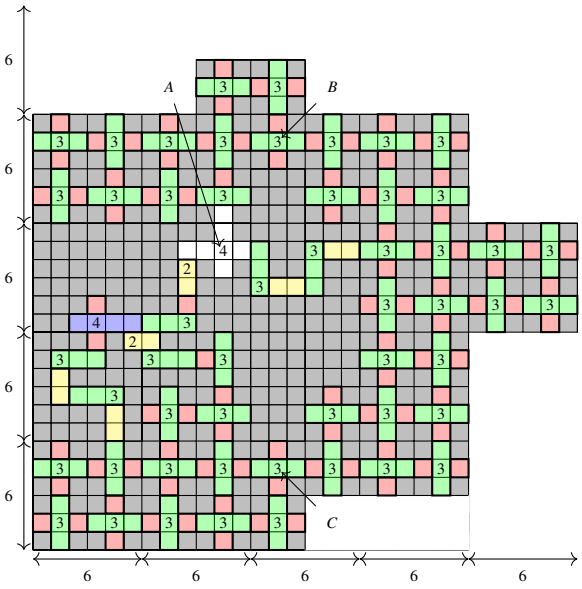


Abbildung 5.17.

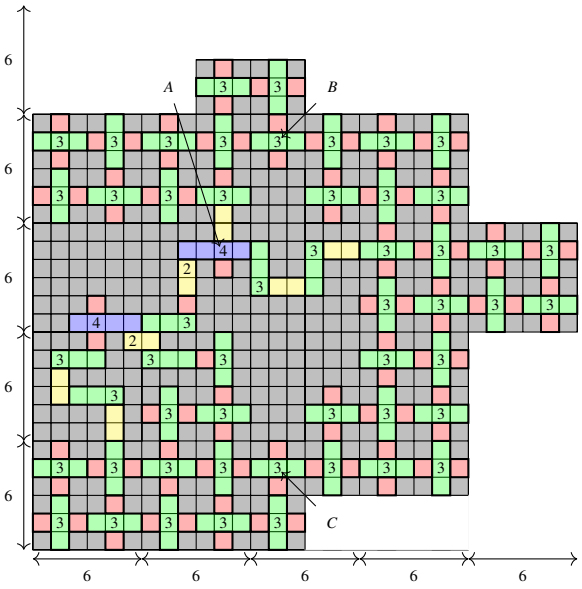


Abbildung 5.18.: Lösung eines UND Moduls mit Eingabe $(x_1, x_2) = (falsch, falsch)$

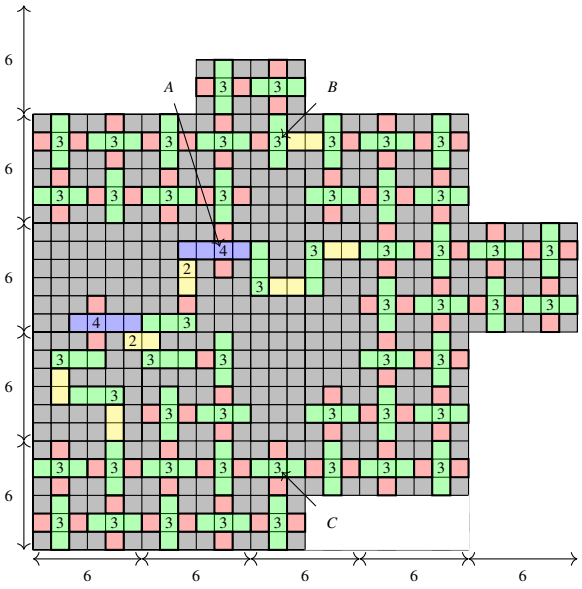


Abbildung 5.19.: Lösung eines UND Moduls mit Eingabe $(x_1, x_2) = (wahr, falsch)$

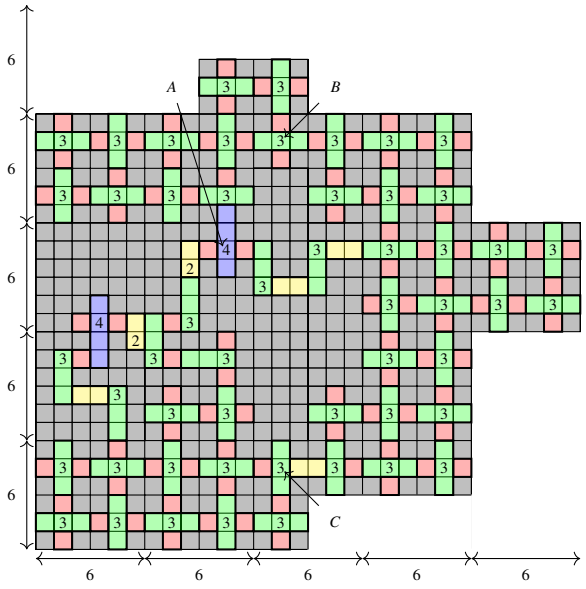


Abbildung 5.20.: Lösung eines UND Moduls mit Eingabe (x_1, x_2) = (*falsch*, *wahr*)

5.3. Beweis zur Korrektheit der Reduktion

Im vorherigen Abschnitt sind alle Gatter, die für einen booleschen Schaltkreis benötigt werden, durch konstruierte Module aus Fillmat Feldern simuliert. Mithilfe dieser Module kann nun ein beliebiger Schaltkreis in Polynomialzeit nachgebaut werden; die Bereiche zwischen den einzelnen Modulen werden dabei mit Löchern gefüllt.

Das erstellte Fillmat Puzzle hat dabei genau dann eine gültige Lösung, wenn es für den ursprüngliche Schaltkreis eine Kombination von Eingabewerten x_1 ... x_n gibt, die den Ausgabewert *y wahr* werden lassen, da die Instanz von Fillmat den ursprünglichen Schaltkreis korrekt simuliert. Daraus folgt:

Satz 5.3. *Das von Uejima und Suzuki definierte Fillmat Entscheidungsproblem (siehe Definition 5.1) ist* NP-vollständig.

Zusätzlich ist die Anordnung der Matten innerhalb der Instanz von Fillmat eindeutig bestimmt, sobald die Kombination von Eingabewerten festgelegt wurde. Diese Anordnung stimmt dabei eins zu eins mit dem Verhalten des ursprünglichen Schaltkreises überein. Folglich gilt:

Satz 5.4. *Die von Uejima und Suzuki definierte ASP-Version von Fillmat (siehe Definition 5.2) ist* ASP-vollständig.

6. Fazit

Das Logikrätsel Fillmat wurde ausführlich beschrieben und die Herangehensweise an eine Lösung anschaulich dargestellt. Bei der Betrachtung einer vollständig ausgefüllten und gelösten Instanz des Puzzlespiels ist es offensichtlich, dass die Korrektheit einer solchen Lösung in polynomieller Zeit überprüfbar ist und Fillmat somit Mitglied der Komplexitätsklasse NP ist. Um weiterhin NP-Schwere zu beweisen, reduzieren Akihiro Uejima und Hiroaki Suzuki das Erfüllbarkeitsproblem für Schaltkreise Circuit-SAT, das in der Arbeit ebenfalls vorgestellt wurde, auf eine für diesen Zweck angepasste Version von Fillmat. Diese Reduktion wurde nachvollzogen und umfangreich mit Grafiken veranschaulicht, wodurch bewiesen wurde, dass das Logikrätsel zu den NP-vollständigen Problemen gehört.

Weiterhin wurde das Konzept des Another Solution Problems erläutert und bewiesen, dass die Version von Fillmat, die Uejima und Suzuki beschreiben, zusätzlich ASP-vollständig ist, da es mit dem Verhalten des in der Reduktion benutzen Circuit-SAT, welches ebenfalls Mitglied der ASP-vollständigen Probleme ist, eins zu eins korrespondiert.

Bei der Einführung von Löchern als Flächen innerhalb eines Fillmat Puzzles, die nicht ausgefüllt werden müssen beziehungsweise können, handelt es sich um eine interessante Maßnahme zur Erschaffung einer Variante des Logikrätsels, die alternative Lösungsvorgänge und eine große Variation von vorgegebenen Feldanordnungen möglich machen könnte. Da es in der ursprünglichen Version von Fillmat, wie sie im Puzzlemagazin Nikoli [7] erschienen ist, keine Löcher gibt, ist die Komplexität dieser ursprünglichen Version jedoch weiterhin ungelöst.

Da sie Matten der Größe 1×4 in ihrer Reduktion ausschließlich in ihrem UND Modul verwenden, weisen Uejima und Suzuki außerdem darauf hin, dass es möglich sei, die Komplexität einer Version von Fillmat zu untersuchen, die gänzlich auf 1×4 Matten verzichtet. Tatsächlich wäre es dahingegen aber ebenso interessant, das Logikrätsel mit Matten der Größe 1×5 zu erweitern und zu untersuchen, welche Auswirkungen die Einführung größerer Matten auf die Komplexität oder die generelle Schwierigkeit des Puzzlespiels hätte.

A. Lösungen der Puzzles in Kapitel 2.4

	1				
2		2		1	
				2	
2		3			
				1	4

Abbildung A.1.: Lösung des Rätsels in Abb. 2.16

2					
					2
			3	4	1
		1			

Abbildung A.2.: Lösung des Rätsels in Abb. 2.17

Literaturverzeichnis

[1] Cameron Browne. Uniqueness in logic puzzles. In *Game & Puzzle Design*, pages 35–37, 2015.

[2] Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971.

[3] M. R. Garey, David S. Johnson, and Robert Endre Tarjan. The planar hamiltonian circuit problem is NP-complete. *SIAM J. Comput.*, 5(4):704–714, 1976.

[4] Adam Krawczyk. The complexity of finding a second hamiltonian cycle in cubic graphs. *J. Comput. Syst. Sci.*, 58(3):641–647, 1999.

[5] William F. McColl. Planar crossovers. *IEEE Trans. Computers*, 30(3):223–225, 1981.

[6] Arne Meier and Heribert Vollmer. *Komplexität von Algorithmen*, volume 4 of *Mathematik für Anwendungen*. Lehmanns Media, 2015.

[7] Ltd. NIKOLI Co. Internetauftritt des Rätselhefts Nikoli, Regeln von Fillmat. https://www.nikoli.co.jp/en/puzzles_jp/fillmat/. Zuletzt aufgerufen am 13.04.2022, 10:15 Uhr.

[8] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[9] Michael Sipser. The history and status of the P versus NP question. In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 603–618. ACM, 1992.

[10] Nobuhisa Ueda and Tadaaki Nagao. NP-completeness results for nonogram via parsimonious reductions. Technical report, 1996.

[11] Akihiro Uejima and Hiroaki Suzuki. Fillmat is NP-complete and ASP-complete. *J. Inf. Process.*, 23(3):310–316, 2015.

[12] Angela und Otto Janko. Internetauftritt mit Beschreibungen, Regeln und Beispielen vieler verschiedener Rätsel einschließlich Fillmat, Kakuro und Fillomino. https://www.janko.at/Raetsel/Firumatto/index.htm, https://www.janko.at/Raetsel/Kakuro/index.htm, https://www.janko.at/Raetsel/Fillomino/index.htm. Zuletzt aufgerufen am 13.04.2022, 10:15 Uhr.

[13] Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 86-A(5):1052–1060, 2003.