Kapitel C

Kombinatorische Spieltheorie und der Satz von Sprague–Grundy

Teach me how to lose a winning match.
Shakespeare, *Romeo and Juliet* 3.2.12

You got to lose to know how to win.
Aerosmith, *Dream On* (1973)

Playing with Chevalley one never won a game. If one had a winning position, Chevalley would sweep the pieces off and start a new game. If one had a losing position, Chevalley insisted that the game be played out.
Richard Bellman (1920–1984), *Eye of the Hurricane* (1984)

Inhalt dieses Kapitels C

1. Neutrale kombinatorische Spiele
   - Einzeiliges Nim und dynamische Programmierung
   - Neutrale Spiele und Rückwärtsinduktion
   - Das Spiel Nim und Boutons effiziente Lösung
   - Summen von Spielen und Sprague–Grundy–Satz
2. Anwendungsbeispiele und weitere Aufgaben
   - Lasker–Nim, Grundys Spiel, Fliesentetris
   - Schleifenspiele: Poker-Nim und Northcotts Spiel
   - Das Spiel Chomp! nach David Gale
   - Wie viel Rationalität benötigen wir?
3. Mengen und Logik als Spiele
   - Schach ist determiniert: Zermelo
   - Mengen als Spiele: von Neumann
   - Quantorenspiele: Eva vs Adam
   - Isomorphiespiele: Samson vs Delila

Motivation und Überblick

Dynamische Spiele nutzen Positionen / Zustände und Züge / Aktionen:

- Es gibt verschiedene (meist nur endlich viele) Positionen, oft auch eine bestimmte Startposition zu Beginn des Spiels.
- Die Regeln des Spiels legen für jede Position eindeutig fest, welche Züge möglich sind; diese führen zu neuen Positionen.
- Der Zustandsübergang kann deterministisch sein oder randomisiert; der „blinde Zufall“ agiert dann formal als spezieller weiterer Spieler.
- Jede Spieler:in entscheidet sich frei unter den ihr möglichen Zügen und erhält Auszahlungen, sofort während des Spiels oder am Ende.

Ausgehend von dieser allgemeinen Beschreibung haben wir im vorigen Kapitel B Graphen als tragende Grundstruktur erklärt. Zusammen mit weiteren Daten zur Auszahlung konnten wir die Gewinnfunktion über die Bellman–Gleichung definieren und bestimmen. Dies ist zunächst nur für einen Spieler formuliert, doch erlaubt schon eine erstaunliche Vielfalt an Optimierungsfragen. Wir wollen dies auf mehrere Spieler ausdehnen.

Motivation und Überblick

Viele Spiele erfreuen sich zudem **vollständiger Information**:

- Jede Spieler:in kennt das gesamte Spiel: alle möglichen Zustände, Aktionen und Auszahlungen. Diese sind *common knowledge*.
- In jedem aktiven Zustand zieht genau eine der Spieler:innen. Es gibt keine Zufallszüge und keine gleichzeitigen Züge.
- Der ziehende Spieler kennt den Startzustand und alle bisherigen Züge und somit den aktuellen Zustand des Spiels (*total recall*). Insbesondere:
- Es gibt keine verdeckten Züge oder Vergesslichkeit der Spieler; diese raffinierten Erweiterungen diskutieren wir erst später.

Wir wollen vorerst diese einschränkenden Annahmen machen, denn sie erleichtern sowohl die Beschreibung des Spiels als auch seine Analyse. Später werden wir auch diese Beschränkungen nach und nach aufheben.

Viele Spiele werden erst reizvoll durch unvollständige Information und ein gerüttelt Maß Zufall. Sie kennen Beispiele aus den vorigen Kapiteln. Es scheint ratsam, zunächst mit einfachen Bei-Spielen zu beginnen.

Motivation und Überblick

Für **kombinatorische Spiele** vereinfachen wir noch weiter:

1. Es gibt genau zwei Spieler (Links / Rechts, Alice / Bob, etc).
2. Sie spielen um Gewinn 1 oder Verlust 0, keine weiteren Werte.
3. Jede:r der beiden kann beginnen. Gezogen wird abwechselnd.
4. Das Spiel endet, wenn der Ziehende keine Zugmöglichkeit hat.
5. Normale Konvention: Wer nicht mehr ziehen kann, verliert.
6. Die Spielregeln garantieren, dass jeder Spielverlauf endet.
7. Beide Spieler haben vollständige Information, wie oben erklärt.
8. Insbesondere gibt es hier keine Zufallszüge.

Wir betrachten speziell **neutrale Spiele** [*impartial games*]:

9. In jeder Position x haben beide Spieler dieselben Zugoptionen.

Das Teilspiel ab x ist daher vollkommen symmetrisch in beiden Spielern. Allein die Unterscheidung in ziehend und wartend bricht die Symmetrie. Diese noch vagen Ideen formalisieren wir durch einen Spielgraphen!

Motivation und Überblick

Modellierung: Prüfen Sie diese Eigenschaften für Ihre Lieblingsspiele!

Spiel / Eigenschaft	1	2	3	4	5	6	7	8	9
Nim und Varianten	✓	✓	✓	✓	✓	✓	✓	✓	✓
Tic-Tac-Toe	✓	✘	✓	✓	✘	✓	✓	✓	✘
Schach	✓	✘	✓	✘	✘	✓	✓	✓	✘
Backgammon	✓	✘	✓	✘	✘	✓	✓	✘	✘
Schiffe versenken	✓	✓	✓	✘	✘	✓	✘	✓	✘
Poker	?✓	✘	✓	✘	✘	✓	✘	✘	✘
Monopoly	?✓	?✓	✓	✘	✘	✘	✘	✘	✘
Schere-Stein-Papier	✓	✘	✘	✘	✘	✓	✘	✓	✓

Auch Fußball, Handball, Basketball, etc. sind Spiele. Vereinfacht sind die Mannschaften hier die beiden Spieler. Man spricht zwar von Positionen und Spielzügen, doch ist es schwierig, sie mathematisch zu präzisieren. Dennoch lohnt es; diese Bemühungen sind in letzter Zeit sehr erfogreich.

Motivation und Überblick

Die wichtigste Frage zu einem kombinatorischen Spiel lautet:
Welcher der beiden Spieler hat eine **Gewinnstrategie**?

Das heißt ausführlich: Welcher Spieler kann gewinnen,
wenn er nur optimal spielt, *egal* wie der Gegner vorgeht?

Dabei darf er sich nicht auf Fehler des Gegners verlassen,
sondern muss auf *alle* Gegenstrategien vorbereitet sein.

Eine Position heißt **Gewinnposition**, wenn der ziehende Spieler gewinnt (bei optimalem Spiel). Andernfalls heißt sie **Verlustposition**.

Ein Zug heißt **Gewinnzug**, wenn der ziehende Spieler durch ihn gewinnt (bei weiterhin optimalem Spiel). Andernfalls heißt er **Verlustzug**.

Die **kombinatorische Spieltheorie** [*combinatorial game theory*, CGT] ist ein umfangreiches und aktives Gebiet der Mathematik und Informatik. Ihre Geschichte beginnt 1901 mit der Lösung des Spiels Nim (Satz C1E):

📖 Charles L. Bouton: *Nim, a game with a complete mathematical theory.* Annals of Mathematics 3 (1901) 35–39. doi.org/10.2307/1967631

Motivation und Überblick

Nim ist der Prototyp für alle Spiele, die in eine Summe unabhängiger Teilspiele zerfallen. Der Satz C1K von Sprague–Grundy überführt dazu jedes neutrale kombinatorische Spiel in ein äquivalentes Nim-Spiel! Damit lassen sich viele Spiele analysieren und effizient lösen.

📖 Roland P. Sprague: *Über mathematische Kampfspiele.* Tôhoku Math. 41 (1935) 438–444. www.jstage.jst.go.jp/article/tmj1911/41/0/41_0_438/_pdf

Roland P. Sprague: *Über zwei Abarten von Nim.* Tôhoku Math. 43 (1937) 451–454. www.jstage.jst.go.jp/article/tmj1911/43/0/43_0_351/_pdf

Patrick M. Grundy: *Mathematics and Games.* Eureka 2 (1939) 6–8

Die monumentalen Klassiker WW und ONAG und das Lehrbuch CGT:

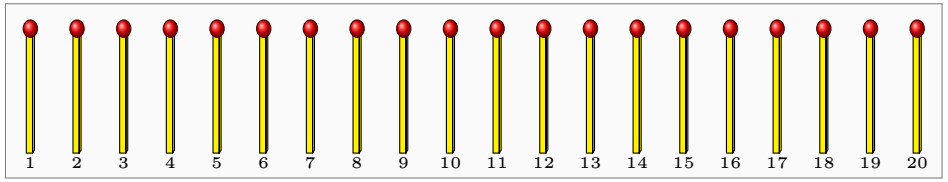
📖 Elwyn R. Berlekamp, John H. Conway, Richard K. Guy:
Winning Ways for Your Mathematical Plays. A K Peters 2001-2004
Gewinnen: Strategien für mathematische Spiele. Vieweg 1985–1986

John H. Conway: *On Numbers and Games.* A K Peters 2000

Aaron N. Siegel: *Combinatorial Game Theory.* AMS 2013

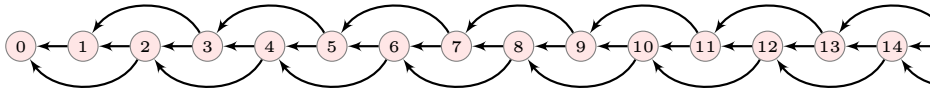
Nimm eins oder zwei!

Auf dem Tisch liegen anfangs $x \in \mathbb{N}$ Streichhölzer, Münzen, Steine, o.ä.



Beide Spieler ziehen abwechselnd, jeder entfernt ein oder zwei Hölzer.
Normalspiel / Misèrespiel: Wer nicht mehr ziehen kann, verliert / gewinnt.

Aufgabe: Übersetzen Sie die Spielregeln in einen Spielgraphen. **Lösung:**



☺ Abstraktion ist die Kunst, Wichtiges von Unwichtigem zu trennen.
So können wir jedes Spiel knapp und übersichtlich als Graph codieren!
Er beantwortet die grundlegenden Fragen zum Spiel und seinen Regeln:
Welche Spielstände x gibt es? Welche Aktionen a sind in x möglich?

Nimm eins oder zwei!

Bevor Sie weiterlesen sollten Sie dieses Spiel einige Male durchspielen, am besten realistisch: gegeneinander! In der Vorlesung spielen Sie gegen mich an der Tafel… und langsam entwickeln Sie Ihre eigene Vermutung. Folgen Sie Ihrer natürlichen Neugier: Es macht Freude! So geht Lernen.

Beobachten Sie Ihren Lernprozess von *Whaaa?* über *Aha!* zu *Alles klar!* Anfangs werden Sie vermutlich wenig Struktur erkennen. Mit Erfahrung ahnen Sie gewisse Regelmäßigkeiten. Diese können Sie in den folgenden Aufgaben ausarbeiten und schließlich die allgemeine Regel formulieren.

Am Ende steht ein mathematischer Satz als Extrakt Ihrer Erfahrungen. Diesen können Sie induktiv beweisen und zukünftig getrost anwenden! Diesen mathematischen Reifeprozess *erkennen – beweisen – anwenden* möchte ich mit Ihnen hier zelebrieren, an mehreren Bei-Spielen.

Abstraktion vereinfacht: Der Spielgraph fasst alles wunderbar zusammen. Jede Ecke (Position) ist eine natürliche Zahl $x \in X := \mathbb{N} = \{0, 1, 2, 3, \dots\}$. Jede Kante (Zug) $a = (x, s) : x \to y$ erfüllt $y = x - s$ mit $s \in S = \{1, 2\}$. Dies ist ein einzeiliges Subtraktionsspiel mit Zugoptionen $S = \{1, 2\}$.

Nimm eins oder zwei! – Wer gewinnt und wie?

Aufgabe: Berechnen Sie für jede Position $x \in X$ Gewinn 1 oder Verlust 0.
Misèrespiel, $\mu : X \to \{0, 1\}$: Wer nicht mehr ziehen kann, gewinnt.
Normalspiel, $\nu : X \to \{0, 1\}$: Wer nicht mehr ziehen kann, verliert.

$$\nu(x) = \begin{cases} 0 & \text{falls } x \in \partial X, \text{ sonst} \\ 1 - \min\{\, \nu(y) \mid x \to y \,\}. \end{cases}$$

☺ Wohldefiniert dank noetherscher Rekursion auf artinschen Graphen!
Noch informativer als ν erweist sich die **Sprague–Grundy–Funktion**:

$$\begin{aligned} &\gamma : X \to \mathbb{N} : \gamma(x) = \operatorname{mex}\{\, \gamma(y) \mid x \to y \,\}, \\ &\operatorname{mex} : \{\, S \subsetneq \mathbb{N} \,\} \to \mathbb{N} : S \mapsto \min(\mathbb{N} \smallsetminus S). \end{aligned}$$

☺ Zu jeder echten Teilmenge $S \subsetneq \mathbb{N}$ definieren wir $\operatorname{mex} S := \min(\mathbb{N} \smallsetminus S)$ als das Minimum des Komplements [engl. *minimal excludant*, kurz *mex*], also die kleinste natürliche Zahl, die nicht in der Menge S enthalten ist.

Beispiele: $\operatorname{mex}\{0, 1, 3, 5\} = 2$, $\operatorname{mex}\{0\} = 1$, $\operatorname{mex}\{1, 2, 3\} = 0$, $\operatorname{mex}\{\,\} = 0$.

Nimm eins oder zwei! – Wer gewinnt und wie?

Die Sprague–Grundy–Funktion γ erweist sich als Hauptakteurin dieser Theorie, ihre Bedeutung erschließt sich allerdings erst nach Satz C1K.

Vorrangig interessiert uns zunächst die Gewinnfunktion $\nu : X \to \{0, 1\}$. Sie sagt zu jeder Spielposition $x \in X$, ob wir in einer Verlustposition mit $\nu(x) = 0$ sind oder aber in einer Gewinnposition mit $\nu(x) = 1$. Genauer:
(0) Aus $x \in X$ mit $\nu(x) = 0$ führt jeder Zug $x \to y$ zu $\nu(y) = 1$.
(1) Aus $x \in X$ mit $\nu(x) = 1$ führt ein Zug $x \to y$ zu $\nu(y) = 0$.

Wir werden bald sehen, dass die Sprague–Grundy–Funktion $\gamma : X \to \mathbb{N}$ noch nützlicher ist als ν. Satz C1D erklärt den genauen Zusammenhang:

$$\nu(x) = \gamma(x) \wedge 1 = \begin{cases} 0 & \text{falls } \gamma(x) = 0 \text{ (Verlust)}, \\ 1 & \text{falls } \gamma(x) \geq 1 \text{ (Gewinn)}. \end{cases}$$

Anschaulich zeigt $\gamma(x) \geq 1$ verschiedene Stufen von Gewinnpositionen.

Das Misèrespiel nenne ich hier zum Kontrast und als Variante zum Üben. Seine Theorie ist ganz anders, wesentlich mühsamer und weniger schön. In der Vorlesung gehe ich daher nicht genauer auf $\mu : X \to \{0, 1\}$ ein, doch hier in den Erläuterungen biete ich es Ihnen zur Ergänzung.

Nimm eins oder zwei! – naiv implementiert

Lösung: (0) Wir berechnen $\mu, \nu : X \to \{0,1\}$ rekursiv gemäß Definition:

```
def mu(x):
    if x == 0: return 1   # Wer nicht mehr ziehen kann, gewinnt.
    return 1 - min( mu(y) for y in range(x-2, x) if y >= 0 )
```

```
def nu(x):
    if x == 0: return 0   # Wer nicht mehr ziehen kann, verliert.
    return 1 - min( nu(y) for y in range(x-2, x) if y >= 0 )
```

Analyse: (a) Was ist gut / schlecht an dieser Implementierung?
(b) Bestimmen Sie die Anzahl $f(x)$ der Funktionsaufrufe!

Antwort: (b) Wir finden $f(0) = 0$ und $f(1) = 1$ sowie für alle $x \in \mathbb{N}_{\geq 2}$ rekursiv $f(x) = f(x-1) + f(x-2)$. Dies ist die Fibonacci–Folge!

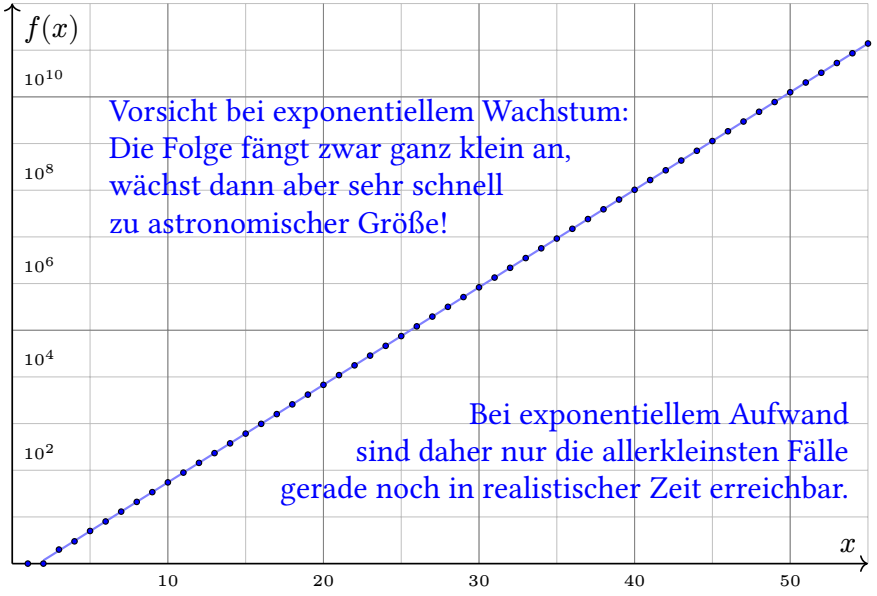
(a) Der Aufwand wächst exponentiell! Explizit gilt die Binet–Formel:

$$f(x) = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^x - \left(\frac{1-\sqrt{5}}{2}\right)^x\right] \gtrapprox 0.447 \cdot 1.618^x$$

Übung: Beweisen Sie die Binet–Formel durch Induktion über $x \in \mathbb{N}$.

Nimm eins oder zwei! – naiv implementiert

Die Fibonacci–Folge in logarithmischer Darstellung:



Nimm eins oder zwei! – naiv implementiert

(0) Die Funktionen `min` und `max` bietet Python bereits standardmäßig, im Gegensatz dazu müssen wir die Funktion `mex` selbst programmieren:

```
def mex(theSet):          # mex = minimal excludant
    m = 0
    while m in theSet: m += 1
    return m
```

Damit implementieren wir die Sprague–Grundy–Funktion $\gamma : X \to \mathbb{N}$ ebenso leicht (und naiv) wie zuvor die Gewinnfunktion $\nu : X \to \{0,1\}$:

```
def gamma(x):             # Die Sprague-Grundy-Funktion
    return mex({ gamma(y) for y in range(x-2, x) if y >= 0 })
for x in range(0, 101):   # Probieren ergänzt Studieren!
    print( x, gamma(x) )  # Es gibt eine Überraschung...
```

Probe: Ist die Berechnung korrekt? Berechnen Sie kleine Beispiele, von Hand und mit Python, und vergleichen Sie die beiden Ergebnisse.

Aufwand: Ist die Berechnung schnell genug? Probieren Sie es selbst aus! Bis $x \approx 30$ geht es noch recht flott, dann nur noch quälend langsam.

Nimm eins oder zwei! – clever memoisiert

Wir merken uns, was wir berechnet haben, und recyclen es! Diese genial-einfache Idee heißt **Memoisation**, abgeleitet von lat. **Memorandum**, kurz **Memo**, *das zu Erinnernde.*

(1) In Python können wir Memoisation elegant wie folgt implementieren:

```
def memoize(f):           # Memoisation einer Funktion f
    memo = {}             # Liste aller vorigen Berechnungen
    def wrapper(x):       # Wir verpacken f in eine Hilfsfunktion.
        if x not in memo: memo[x] = f(x) # Berechnen und speichern
        return memo[x]    # Jeder Wert wird nur einmal berechnet.
    return wrapper        # Nur wrapper ist von außen sichtbar.
@memoize
def gamma(x):             # Die Sprague-Grundy-Funktion
    return mex({ gamma(y) for y in range(x-2, x) if y >= 0 })
```

☺ Naive Rekursion ist zwar korrekt, aber allzu verschwenderisch. Nachhaltige Buchführung reduziert den Rechenaufwand erheblich!

☺ In diesem Beispiel ist die Rechenzeit für γ nur noch linear in x. Probieren Sie es aus: vorher noch quälend langsam, nun blitzschnell!

Nimm eins oder zwei! – mathematisch optimiert

Lösung: (1) Wir rechnen rekursiv, geschickt sortiert *bottom-up*:

$x=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mu=$	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1
$\nu=$	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1
$\gamma=$	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2

Ebenso gut gelingt der rekursive Ansatz *top-down* mit Memoisation.
(2) Wie lautet die allgemeine Regel? Wir krönen unsere Bemühungen:

Satz C1A: einzeiliges Nim mit Zugoptionen $S = \{1, 2, \dots, n-1\}$, $n \geq 2$

Misèrespiel: Genau dann ist x eine Verlustposition, wenn $x \operatorname{rem} n = 1$.
Normalspiel: Genau dann ist x eine Verlustposition, wenn $x \operatorname{rem} n = 0$.
Die Sprague–Grundy–Funktion des Spiels ist $\gamma : \mathbb{N} \to \mathbb{N} : x \mapsto x \operatorname{rem} n$.

☺ So erkennen und nutzen Sie Gewinnpositionen, wunderbar effizient.
Genial! Das ist Mathematik: (1) erkennen, (2) beweisen, (3) anwenden.

▶ Als mechanischer Computer erklärt von Matt Parker aka Stand-up Maths: *The Unbeatable Game from the 60s: Dr NIM.* `youtu.be/9KABcmczPdg`

Nimm eins oder zwei! – mathematisch optimiert

Aufgabe: (2) Beweisen Sie diesen Satz per Induktion über $x \in \mathbb{N}$.

Lösung: (2a) Induktionsanfang: Zunächst gilt $\mu(0) = 1$ nach Misèreregel.
Es folgt $\mu(1) = 0$, da nur der Zug $1 \to 0$ möglich ist; der Gegner gewinnt.
Induktionsschritt: Sei $x \geq 2$ und die Aussage zu $\mu(y)$ gelte für alle $y < x$.
Zu $x \operatorname{rem} n \neq 1$ existiert ein Zug $x \to y$ mit $x - y \in S$ und $y \operatorname{rem} n = 1$,
also $\mu(y) = 0$; dieser Gewinnzug garantiert $\mu(x) = 1$. Von $x \operatorname{rem} n = 1$
führen alle Züge $x \to y$ mit $x - y \in S$ zu $y \operatorname{rem} n \neq 1$, also $\mu(y) = 1$.

(2b) Induktionsanfang: Zunächst gilt $\nu(0) = 0$ nach Normalspielregel.
Induktionsschritt: Sei $x \geq 1$ und die Aussage zu $\nu(y)$ gelte für alle $y < x$.
Zu $x \operatorname{rem} n \neq 0$ existiert ein Zug $x \to y$ mit $x - y \in S$ und $y \operatorname{rem} n = 0$,
also $\nu(y) = 0$; dieser Gewinnzug garantiert $\nu(x) = 1$. Von $x \operatorname{rem} n = 0$
führen alle Züge $x \to y$ mit $x - y \in S$ zu $y \operatorname{rem} n \neq 0$, also $\nu(y) = 1$.

(2c) Sei $x \in \mathbb{N}$ und $r = x \operatorname{rem} n$. Wir zeigen $\gamma(x) = r$.
Induktionsvoraussetzung: Für alle $y < x$ gelte $\gamma(y) = y \operatorname{rem} n$.
Für $x < n$ gilt $r = x$ und wir finden $\gamma(x) = \operatorname{mex}\{0, \dots, r-1\} = r$.
Für $x \geq n$ gilt ebenso $\gamma(x) = \operatorname{mex}\{0, \dots, r-1, r+1, \dots, n-1\} = r$. $\boxed{\text{QED}}$

Nimm eins oder zwei! – mathematisch optimiert

☺ Die **Dynamische Programmierung** [*Dynamic Programming*, DP] ist ein Werkzeugkasten zur Optimierung rekursiver Berechnungen.
Eine systematische Darstellung finden Sie in dem populären Lehrbuch
📖 Cormen, Stein, Leiserson, Rivest: *Introduction to Algorithms*, §15.
Oben haben wir zwei komplementäre Implementierungen gesehen:

Top-down mit Memoisation: Wir formulieren unsere Funktion rekursiv und speichern das Ergebnis jedes gelösten Teilproblems, etwa in einem assoziativen Array [*map*, *dictionary*] oder einer Tabelle [*hash table*].

Bottom-up topologisch sortiert: Jedes Teilproblem nutzt jeweils nur kleinere, bereits gelöste. Beide Formulierungen leisten meist dasselbe; Einsparungen entstehen, wenn top-down große Lücken überspringt.

☺ Idealerweise können Algorithmen mit exponentiellem Aufwand so auf polynomiellen Aufwand reduziert werden, wie in unserem Beispiel: Unsere Lösung ist zunächst exponentiell in x, dann polynomiell in x, hier sogar linear in x, dank Satz C1A schließlich sogar nur logarithmisch in x.
Das natürliche **Komplexitätsmaß** ist hier die Bitlänge $\operatorname{len}(x) \sim \log_2(x)$.

Nimm eins oder zwei! – mathematisch optimiert

☹ **Rekursion** hat unter Anfänger:innen meist einen schlechten Ruf:
Zuerst sind Denkweise und Progammiertechnik recht anspruchsvoll.
Ist diese Hürde genommen, so folgt gleich die größe Ernüchterung:
Naive Implementierung führt meist zu exponentiellem Aufwand.
Im Jobinterview machen Sie damit allein keinen guten Eindruck.
„Never hire a developer who computes the factorial using recursion.“

☺ Rekursion entfaltet ihre wahre Kraft erst durch raffiniert-effiziente Implementierung: Die genial-einfache Idee hierzu heißt **Memoisation**, das geschickte Speichern der zuvor berechneten Zwischenergebnisse.
Im vorliegenden Falle vollendet Satz C1A unsere Lösung durch eine weitere dramatische Optimierung: Die Berechnung von $x \operatorname{rem} n$ benötigt nur noch logarithmischen Aufwand, gemäß Bitlänge $\operatorname{len}(x) \sim \log_2(x)$.

☺ Sie ist zudem so einfach, dass wir sie im Kopf ausführen können!
Das spüren Sie deutlich, wenn Sie dieses Spiel gegeneinander spielen.
Diesen Vorteil nutze ich natürlich, wenn ich erstmals gegen Sie spiele.
Beobachten Sie Ihren Lernprozess von *Whaaa?* über *Aha!* zu *Alles klar!*

Aufgabe: Untersuchen Sie ebenso einzeiliges Nim mit den Zugoptionen (3) $S = \{1, 2, 4\}$ und (4) $S = \{1, 3, 4\}$ und (5) $S = \{1, 3, 6\}$. **Lösung:**

$x=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mu=$	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1
$\nu=$	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1
$\gamma=$	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2

$x=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mu=$	1	0	1	0	1	1	1	1	0	1	0	1	1	1	1	0	1	0	1	1	1
$\nu=$	0	1	0	1	1	1	1	0	1	0	1	1	1	1	0	1	0	1	1	1	1
$\gamma=$	0	1	0	1	2	3	2	0	1	0	1	2	3	2	0	1	0	1	2	3	2

$x=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mu=$	1	0	1	0	1	0	1	1	1	1	0	1	0	1	0	1	1	1	1	0	1
$\nu=$	0	1	0	1	0	1	1	1	1	0	1	0	1	0	1	1	1	1	0	1	0
$\gamma=$	0	1	0	1	0	1	2	3	2	0	1	0	1	0	1	2	3	2	0	1	0

Aufgabe: Wir betrachten weiterhin einzeiliges Nim, nun jedoch mit den Zugoptionen $S = \{1, 4, 5, 7\}$. Schreiben Sie ein Python-Skript zur Berechnung von $\mu(x), \nu(x), \gamma(x)$ für $x = 0, \dots, 100$. Gelingt Ihnen diese Berechnung schnell genug? mit nur linearem Zeitaufwand?

Lösung: Für den Bereich $x = 0, \dots, 100$ genügt die naiv-rekursive Implementierung nicht. Wir nutzen daher unsere Kenntnisse zur Memoisierung, hier *bottom-up* durch den Aufbau einer Wertetabelle:

```
S = { 1, 4, 5, 7 }        # die vorgegebenen Zugoptionen
Mu = { 0: 1 }             # Startwert für das Misèrespiel
Nu = { 0: 0 }             # Startwert für das Normalspiel
Gamma = { 0: 0 }          # Startwert für Sprague-Grundy
for n in range(1, 101):
    Mu[n] =    1 - min( Mu[n-s] for s in S if s <= n )
    Nu[n] =    1 - min( Nu[n-s] for s in S if s <= n )
    Gamma[n] = mex({ Gamma[n-s] for s in S if s <= n })
```

Übung: Sie können nun nach Mustern suchen, speziell nach periodischer Wiederholung, dies als Vermutung formulieren und dann beweisen.

Richard Bellman (1920–1984) war ein US-amerikanischer Mathematiker. Er entwickelte 1953 die Kernidee der **Dynamischen Programmierung** zur algorithmischen Lösung von Optimierungsproblemen durch Zerlegung in Teilprobleme und systematische Speicherung von Zwischenergebnissen.

Ökonomen bezeichnen die Dynamische Programmierung schlicht als **Rekursionsmethode**. Sie tritt bei zahlreichen Optimierungsproblemen natürlich auf und wird gerne und erfolgreich angewendet. Sie ist daher ein beliebtes Universalwerkzeug der Wirtschaftswissenschaften.

Idealerweise entspringt die Rekursion unmittelbar der ökonomischen Fragestellung. In vereinfachter Form findet sie sich daher bereits bei Ernst Zermelo 1913 zu Schach sowie allgemein bei John von Neumann und Oskar Morgenstern, *Theory of Games and Economic Behavior*, 1944.

Die Dynamische Programmierung ist in der Informatik eine universelle Strategie, um Probleme rekursiv zu lösen. Zur gewünschten Funktion muss eine geeignete Rekursionsstruktur aber erst gefunden werden. Diese Kunst erfordert sowohl Erfahrung als auch Kreativität.

Rekursion und Dynamische Programmierung sind universell anwendbar und in zahlreichen Anwendungen nützlich. Deshalb lernen Sie diese wunderbaren Techniken, sobald Sie sich mit Programmierung im Allgemeinen oder mit Optimierung im Speziellen beschäftigen.

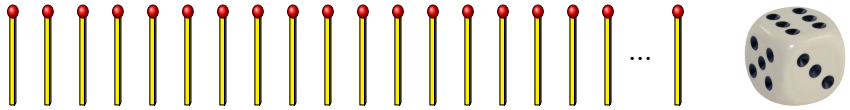
Darüber hinaus zeigt das obige Beispiel eine weitere wichtige Technik: Wenn wir ein Muster erkennen, können wir es als eine Vermutung formulieren und womöglich als einen allgemeinen Satz beweisen. Wenn dies gelingt, so ist es meist noch wesentlich effizienter.

Idealerweise wünschen wir uns die Lösung des gestellten Problems nicht als Programm (für Computer), sondern als eine geschlossene Formel (für Menschen). Dieser Unterschied in der Form ist jedoch nur oberflächlich, eigentlich gemeint ist: eine möglichst einfache und effiziente Lösung.

Wenn Sie eine geschlossene Formel *vermuten*, wie im obigen Beispiel $\gamma(x) = x \operatorname{rem} n$, dann wollen Sie diese für alle Fälle $x \in X$ *beweisen*. Deshalb lernen Sie die wunderbare Technik der vollständigen Induktion Beide Techniken ergänzen sich und arbeiten wunderbar zusammen!

Streichhölzer und Würfel

C117
Erläuterung



Aufgabe: Auf dem Tisch liegen $n = 5000$ Streichhölzer und ein Würfel mit der Augenzahl $a \in \{1, \dots, 6\}$; gegenüberliegende addieren sich zu 7. Alice und Bob ziehen abwechselnd, Alice beginnt: Wer am Zug ist, kippt den Würfel über eine Kante seiner Wahl zu einer neuen Augenzahl b und entfernt b Streichhölzer. Wer keinen solchen Zug mehr ausführen kann, hat verloren. Bei welcher Augenzahl kann Alice ihren Sieg erzwingen?

Anleitung: Zu $n \in \mathbb{N}$ und $a \in \{1, \dots, 6\}$ setzen wir $u(n, a) = 1$, falls die ziehende Spieler:in ihren Sieg erzwingen kann, andernfalls $u(n, a) = 0$.
(1) Berechnen Sie $u(n, a)$ für $n = 0, 1, \dots, 16$.
(2) Setzen Sie das Muster fort bis $n = 5000$.
(3) Alternativ: Schreiben Sie ein Programm.

BWM 2022, Runde 2, Aufgabe 2, www.mathe-wettbewerbe.de/aufgaben
DorFuchs youtu.be/nG1NlEjXeKA: schön und enthusiastisch erklärt.

Streichhölzer und Würfel

C118
Erläuterung

Lösung: Wir berechnen die Gewinnfunktion $u(n, a)$ rekursiv:

$n =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...	5000
$a = 1$	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1		1
2	0	1	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1		0
3	0	1	1	0	0	1	1	1	0	0	1	1	0	0	1	1	1		1
4	0	1	1	0	0	1	1	1	0	0	1	1	0	0	1	1	1		1
5	0	1	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1		0
6	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1		1

Dies wiederholt sich ab Spalte 2 alle 9 Schritte, also $u(5000, a) = u(5, a)$. Bei $a \in \{1, 3, 4, 6\}$ kann Alice ihren Sieg erzwingen, bei $a \in \{2, 5\}$ Bob. Die Tabelle können Sie selbst erstellen und prüfen. Alternativ in Python:

```
u = [ [1,1,1,1,1,1] for i in range(23) ];
for i in range(6,23):  # Index i = n+6 mit n = Anzahl der Streichhölzer
    for j in range(6): # Index j = a-1 mit a = Augenzahl des Würfels
        u[i][j] = 1 - min(u[i-1-k][k] for k in range(6) if k != j and k != 5-j)
```

Streichhölzer und Tetraeder

C119
Erläuterung

☺ Bei kombinatorischen Spielen drängt sich die rekursive Lösung auf. Das illustriert wunderbar die Technik der Rekursion bzw. Induktion: Sie ist elegant und effizient darzustellen und leicht nachzuvollziehen.

Aufgabe: Ebenso für einen Tetraeder mit Augenzahlen $a \in \{1, 2, 3, 4\}$.

Lösung: Wir berechnen die Gewinnfunktion $u(n, a)$ rekursiv:

$n =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	...	5000
$a = 1$	0	0	1	1	1	0	1	1	1	1	0	0	1	1		0
2	0	1	1	1	1	0	1	0	1	1	0	1	1	1		0
3	0	1	1	0	1	0	1	1	1	1	0	1	1	0		0
4	0	1	1	1	0	0	1	1	1	1	0	1	1	1		0

Dies wiederholt sich ab Spalte 0 alle 10 Schritte, also $u(5000, a) = u(0, a)$. Das bedeutet: Wenn Bob fehlerfrei spielt, kann er seinen Sieg erzwingen.

```
u = [ [1,1,1,1] for i in range(18) ];
for i in range(4,18):  # Index i = n+4 mit n = Anzahl der Streichhölzer
    for j in range(4): # Index j = a-1 mit a = Augenzahl des Tetraeders
        u[i][j] = 1 - min(u[i-1-k][k] for k in range(4) if k != j)
```

Streichhölzer und Tetraeder

C120
Erläuterung

Ist es klar, dass die Tabelle schließlich periodisch werden muss? Ja! Wir schauen immer nur endlich viele Schritte zurück (hier 6 bzw. 4), und dabei sind jeweils nur endlich viele Konfigurationen möglich. Irgendwann muss sich die Konfiguration also wiederholen… und fortan die gesamte weitere Tabelle. Das verhilft uns zur effizienten Rechnung.

Wie lang jedoch die Vor/Periode ist, lässt sich nicht leicht vorhersagen, sondern am besten durch explizite Rechnung bestimmen, so wie oben. Die Periode erweist sich hierbei als erfreulich kurz, dadurch bleibt der Rechenaufwand gering. Das spüren Sie, wenn Sie per Hand rechnen!

In der obigen Lösung habe ich die Tabelle jeweils so weit angegeben, bis die Periode sichtbar wird. Bei der Handrechnung würden Sie das Muster erkennen und die (etwas mühsame) Rechnung sofort einstellen. Der Computer verleitet hingegen dazu, einfach stur weiter zu rechnen.

Am besten kombinieren Sie beide Ansätze, Stift&Papier vs Computer: Sie ergänzen sich bestens und dienen zur gegenseitigen Überprüfung. So verbinden Sie die routinierte Methode mit kreativen Lösungen.

Wie teilen Bonnie und Clyde ihre Beute?

Auf dem Tisch liegt die Beute von $n \in \mathbb{N}$ Münzen,
davon nimmt Bonnie $b \in \{1, 2, 4\}$ Münzen weg,
dann nimmt Clyde $c \in \{1, 3, 4\}$ Münzen weg,
und dann abwechselnd immer so weiter.
Wer nicht mehr ziehen kann, verliert.

Bonnie and Clyde (1967), Warner Bros

Aufgabe: Wer von den beiden kann
durch fehlerfreies Spiel gewinnen?

Lösung: Wir berechnen den Gewinn rekursiv:

Ab $n =$	0	1	2	3	4	5	6	7	8	9	...	2025
B beginnt	C	B	B	B	B	C	B	B	B	B	...	C
C beginnt	B	C	B	C	C	B	C	B	C	C	...	B

Zunächst berechnen wir die Tabelle für $n = 0, \dots, 9$ nach obigen Regeln.
Daran erkennen wir das Muster: Alle weiteren Werte haben Periode 5.
Die so gefundene Aussage können wir per Induktion leicht beweisen.
☺ Damit können wir das Ergebnis für jedes $n \in \mathbb{N}$ effizient berechnen!

Wie teilen Bonnie und Clyde ihre Beute?

So berechnen wir die obige Tabelle in Python:

```
Bop = { 1, 2, 4}; Bnu = { 0: 0 } # Bonnies Zugoptionen und Gewinn im Normalspiel
Cop = { 1, 3, 4}; Cnu = { 0: 0 } # Clydes  Zugoptionen und Gewinn im Normalspiel
for n in range(1, 20):
    Bnu[n] = 1 - min( Cnu[n-b] for b in Bop if b <= n ) # kreuzweise Minimierung
    Cnu[n] = 1 - min( Bnu[n-c] for c in Cop if c <= n ) # kreuzweise Minimierung
print( *['B' if Bnu[n] == 1 else 'C' for n in range(0, 20)], sep=' ', end='\n' )
print( *['C' if Cnu[n] == 1 else 'B' for n in range(0, 20)], sep=' ', end='\n' )
```

Die vorigen kombinatorischen Spiele waren **symmetrisch** aka **neutral** [*impartial games*]: Die Regeln sind für beide Spieler:innen dieselben. Die Symmetrie wird lediglich dadurch gebrochen, dass die eine zieht und der andere wartet; anschließend wechselt nur dieser temporäre Status.

Für Bonnie und Clyde sind die Spielregeln nun nicht mehr symmetrisch, das Spiel ist also nicht mehr neutral, sondern **parteiisch** [*partizan game*]: Bonnie mit $b \in \{1, 2, 4\}$ und Clyde mit $c \in \{1, 3, 4\}$ haben *verschiedene* Zugmöglichkeiten, jeweils spezifisch für diese eine Spieler:in.

Rekursion vs Induktion

Dynamische Programmierung
= Rekursion + Memoisation

Allgemeines Muster + Induktion
= effiziente Lösungsformel

Die vorigen Beispiele zeigen eindrücklich den erfolgreichen Dreischritt:
1. Die geschickte Rekursion löst das gestellte Problem *effektiv*.
2. Die Memoisation macht die rekursive Rechnung *effizient*.
3. Induktiv bewiesene Muster *trivialisieren* das Problem.

Rekursion 1 und Memoisation 2 lassen sich meist wunderbar mit einem Computer ausführen, für kleine Probleme auch noch mit Stift und Papier. Aus diesen Daten lässt sich in günstigen Fällen ein allgemeines Muster 3 destillieren; diese Aussage können wir dann induktiv beweisen. Ideal!

Rekursion vs Induktion

☺ Sie sehen hier wunderbar das Zusammenspiel beider Techniken, vermutlich klarer und realistischer als je zuvor im Mathestudium:
Per Rekursion konstruieren / berechnen wir noch unbekannte Werte.
Per Induktion beweisen / überprüfen wir eine vorgelegte Aussage.

Die Induktion sagt uns, wie wir eine Vermutung *beweisen* können, jedoch nicht, wie wir mögliche Vermutungen überhaupt erst *finden*.
In den obigen Anwendungen hilft uns die Rekursion: So berechnen wir die ersten Beispiele und suchen / finden / vermuten damit ein Muster.

Induktion ist im Wesentlichen nicht stumpfsinniges, formales Rechnen, sondern eine filigrane Kunst. Probieren Sie selbst, Sie werden es erleben! Die Formulierung der induktiv zu zeigenden Behauptung ist wesentlicher Teil der Aufgabe. Das erfordert Geschick, Kreativität und Erfahrung!

Lassen Sie sich nicht davon einlullen, dass viele Lehrbücher die ersten Induktionsaufgaben meist stereotyp als langweiligen Drill formulieren. Das ist selbstverständlich sinnvoll und hilfreich für die ersten Schritte, aber kein realistisches Vorbild für selbständige mathematische Arbeit.

Definition C1B: neutrales Spiel

Ein **neutrales Spiel** (G, v) mit konstanter Summe $c \in \mathbb{R}$ besteht aus einem artinschen Graphen $G = (X, A, \sigma, \tau)$ mit Auszahlung $v : \partial X \to \bar{\mathbb{R}}$.

Beide Spieler ziehen abwechselnd gemäß den Kanten A des Graphen. In jedem terminalen Zustand $x \in \partial X$ erhält der Ziehende schließlich die Auszahlung $v(x)$ und der Wartende das Komplement $c - v(x)$. Was ist die Auszahlung des Ziehenden bei optimalem Spiel?

Satz C1B: Gewinnfunktion durch Rückwärtsinduktion

Dazu existiert die eindeutige **Gewinnfunktion** $u : X \to \bar{\mathbb{R}}$ mit $u|_{\partial X} = v$, die in jedem aktiven Zustand $x \in X^\circ$ die Max-Min-Bedingung erfüllt:

$$\boxed{u(x) \overset{!}{=} \sup_{x \to y} \Big[c - u(y) \Big] = c - \inf_{x \to y} \Big[u(y) \Big]}$$

Eine **Lösung** oder **optimale Strategie** s ordnet jedem aktiven Zustand $x \in X^\circ$ eine optimale Aktion $s(x) = a : x \to y$ mit $u(x) = c - u(y)$ zu.

Übung: Beweisen Sie Existenz und Eindeutigkeit der Gewinnfunktion u.

☺ Die Beweisidee ist anschaulich klar und aus Beispielen vertraut: Wir beweisen Existenz durch Rekursion, Eindeutigkeit durch Induktion. Allerdings ist nicht offensichtlich, worüber hier induziert werden soll… Versuchen Sie es zunächst selbst als Fingerübung, dann hilft Satz B1L.

Bemerkung: Formal ist die Max-Min-Bedingung die Bellman–Gleichung mit konstanter sofortiger Belohnung $r(a) = c$ und Diskontfaktor $\delta = -1$. Diese Parameterwahl codiert unser Modell der konstanten Summe c.

Der Diskontfaktor $\delta \in [0, 1]$ beschreibt den schrittweisen **Wertverlust**, zum Beispiel durch Inflation, Abbruchrisiko oder allgemein Ungeduld.

Der Faktor $\delta = -1$ codiert die schrittweise **Wertumkehr** wie in C1B: Beide Spieler verfolgen dasselbe Ziel, mit umgekehrten Vorzeichen.

☺ Diese simple Beobachtung ist zunächst nur eine formale Analogie, doch sie dient zur Wiederverwendung der Techniken aus Kapitel B. Damit folgt der obige Satz C1B aus dem bereits bewiesenen Satz B1L.

Ein **Nullsummenspiel** ist ein Spiel (G, v) mit konstanter Summe $c = 0$. Jede Spieler:in trachtet, wie immer, ihren eigenen Gewinn zu maximieren; hier ist dies äquivalent dazu, den Gewinn ihres Gegners zu minimieren. Solche Spiele sind strikt kompetitiv, sie erlauben keine Kooperation.

Spiele mit konstanter Summe $c \in \mathbb{R}$ sind flexibler in der Formulierung. Allgemein: Was die eine Spieler:in gewinnt, geht dem anderen verloren. Sie sind äquivalent zu Nullsummenspielen durch Translation:

Übung: Sei (G, v) ein neutrales Spiel mit konstanter Summe $c \in \mathbb{R}$ und Gewinnfunktion u. Sei $\tilde{v} = \lambda v + \kappa$ mit $\lambda \in \mathbb{R}_{>0}$ und $\kappa \in \mathbb{R}$. Dann ist $(G, \tilde{v})$ ebenfalls ein neutrales Spiel mit konstanter Summe $\tilde{c} = \lambda c + 2\kappa$ und Gewinnfunktionen $\tilde{u} = \lambda u + \kappa$. Speziell für $(\lambda, \kappa) = (2, -c)$ folgt $\tilde{c} = 0$.

Beispiele: Bei kombinatorischen Spielen nutzen wir im Folgenden die Auszahlungen $0 =$ *Verlust* und $1 =$ *Gewinn* mit konstanter Summe $c = 1$. Ebenso möglich wäre $-1 =$ *Verlust* und $+1 =$ *Gewinn* mit Summe $c = 0$. In manchen Spielen, wie Schach, ist auch ein $0 =$ *Unentschieden* möglich.

Bemerkung: Wir können alle Aktionen mit gemeinsamem Start und Ziel zusammenfassen, da es keine sofortigen Belohnungen gibt, sondern nur die terminale Auszahlung $v : \partial X \to \bar{\mathbb{R}}$, auf die wir alles ausrichten.

Für jedes neutrale Spiel (G, v) genügt also ein **einfacher Graph** G: Von jedem Start $x \in X$ zu jedem Ziel $y \in X$ existiert höchstens eine Kante, die Randabbildung $\partial = (\sigma, \tau) : A \to X \times X$ ist somit injektiv.

Satz C1B garantiert Eindeutigkeit und Existenz der Gewinnfunktion $u : X \to \bar{\mathbb{R}}$. In vielen Anwendungen ist der Graph zudem lokal-endlich, das heißt: Jeder Zustand $x \in X$ erlaubt nur endlich viele Aktionen.

Für jeden lokal-endlichen Graphen wird das Maximum/Minimum jeweils angenommen: Es existiert also (mindestens) eine optimale Strategie. Diese explizit zu berechnen, kann jedoch beliebig komplex sein.

Genau darum geht es uns im Folgenden: Wenn wir das Spiel gewinnen wollen, dann müssen wir eine Gewinnstrategie nicht nur prinzipiell berechnen können, sondern möglichst effizient!

Was ist eine effiziente Lösung?

Definition C1c: Lösung, allgemein und polynomiell

Sei (G, v) ein neutrales Spiel mit Summe $c \in \mathbb{R}$ und der Gewinnfunktion $u : X \to \bar{\mathbb{R}}$, also $u|_{\partial X} = v$ und $u(x) = \sup_{x \to y}[c - u(y)]$ für alle $x \in X^\circ$.

Eine **Lösung** oder **optimale Strategie** s ordnet jedem aktiven Zustand $x \in X^\circ$ eine optimale Aktion $s(x) = a : x \to y$ mit $u(x) = c - u(y)$ zu.

Unter einer **polynomiellen Lösung** verstehen wir einen Algorithmus mit polynomieller Laufzeit, der eine optimale Aktion $x \mapsto a$ berechnet.

Beispiel: Die Eingabe einer natürlichen Zahl messen wir in Bitlänge:

$$\boxed{\operatorname{len} : \mathbb{N} \to \mathbb{N} : x \mapsto \min\{\, \ell \in \mathbb{N} \mid a < 2^\ell \,\}}$$

Wir betrachten einzeiliges Nim mit Aktionen $a \in S = \{1, 2, \dots, n-1\}$: Zu jedem Zustand $x \in \mathbb{N}$ erfordert die memoisierte Rekursion x Schritte, ist also *exponentiell* in $\operatorname{len}(x)$. Satz C1A bietet eine *polynomielle* Lösung: Es genügt $a = x \operatorname{rem} n$ zu berechnen. **Übung:** Führen Sie dies sorgsam aus! Warum ist der Zeitaufwand bei festem n höchstens linear in $\operatorname{len}(x)$?

Was ist eine effiziente Lösung?

Ist der Graph G **lokal-endlich und artinsch**, so existiert eine Lösung: Die Rückwärtsinduktion C1B konstruiert eine optimale Strategie $x \mapsto a$. Oft erfordert das exponentiellen Aufwand. *Ain't nobody got time for that!*

Was bedeutet das genau? Wie messen wir Komplexität und Aufwand? Zustände und Aktionen seien codiert durch $X, A \hookrightarrow \{0, 1, |, (,), \dots\}^{(\mathbb{N})}$. Wir messen die Komplexität $\operatorname{len}(x)$ als Länge über diesem Alphabet. Im binären Falle, wie oben erklärt, messen wir die Länge in Bits.

Ein vorgelegter **Algorithmus** zur Berechnung einer Strategie $s : x \mapsto a$ hat zur Eingabe x eine gewisse Laufzeit $T(x) \in \mathbb{R}_{\geq 0}$, etwa gemessen in Sekunden oder Taktzyklen. Gilt $T(x) \leq c_0 + c_1 \operatorname{len}(x)^\alpha$ mit Konstanten $c_0, c_1, \alpha \in \mathbb{R}_{\geq 0}$, so ist die Laufzeit (höchstens) polynomiell vom Grad α. Gilt hingegen $T(x) \geq c_0 \exp(c_1 \operatorname{len}(x))$ mit Konstanten $c_0, c_1 \in \mathbb{R}_{>0}$, so ist die Laufzeit (mindestens) exponentiell, und somit nicht polynomiell.

Unter einer **polynomiellen Lösung** verstehen wir einen Algorithmus mit polynomieller Laufzeit, der jedem aktiven Zustand $x \in X^\circ$ eine optimale Aktion $a : x \to y$ zuordnet, sodass $u(x) = c - u(y)$ gilt.

Normalspiel und Sprague–Grundy–Funktion

Wir vereinfachen weiter und erlauben nur zwei mögliche Auszahlungen: entweder 0 für null / falsch / Niederlage oder 1 für eins / wahr / Gewinn. Die konstante Summe sei $c = 1$, also $u \mapsto 1 - u$ die logische Negation. Wir nennen dann (G, v) ein **neutrales kombinatorisches Spiel**.

Satz C1D: Normalspiel und Sprague–Grundy–Funktion

Auf dem artinschen Graphen $G = (X, A, \sigma, \tau)$ betrachten wir:

- Das **Misèrespiel** $(G, 1)$: Wer nicht mehr ziehen kann, gewinnt.
- Das **Normalspiel** $(G, 0)$: Wer nicht mehr ziehen kann, verliert.

Hierzu existieren die eindeutigen Gewinnfunktionen $\mu, \nu : X \to \{0, 1\}$ mit den vorgegebenen Randwerten $\mu|_{\partial X} = 1$ und $\nu|_{\partial X} = 0$.

Ist G zudem lokal-endlich, so existiert die **Sprague–Grundy–Funktion**

$$\boxed{\begin{aligned} &\gamma : X \to \mathbb{N} : \gamma(x) = \operatorname{mex}\{\gamma(y) \mid x \to y\}, \\ &\operatorname{mex} : \{\, S \subsetneq \mathbb{N} \,\} \to \mathbb{N} : S \mapsto \min(\mathbb{N} \smallsetminus S). \end{aligned}}$$

Es gilt $\nu = \gamma \wedge 1$, also $\nu(x) = 0$ gdw $\gamma(x) = 0$ und $\nu(x) = 1$ gdw $\gamma(x) \geq 1$.

Normalspiel und Sprague–Grundy–Funktion

Beweis: Rückwärtsinduktion C1B garantiert Existenz und Eindeutigkeit von $\mu, \nu : X \to \{0, 1\}$, ebenso für $\gamma : X \to \mathbb{N}$ dank lokaler Endlichkeit. (Andernfalls formulieren wir mex und γ allgemein für Ordinalzahlen.)

Wir nutzen hier die übliche und bequeme Notation $a \wedge b := \min\{a, b\}$. Wir vergleichen $\nu : X \to \{0, 1\}$ und $\gamma : X \to \mathbb{N}$ und zeigen $\nu = \gamma \wedge 1$:

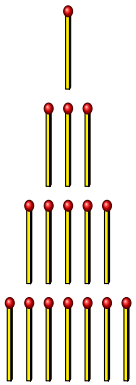
Wir betrachten eine Ecke $x \in X$. Gilt $\nu(y) = \gamma(y) \wedge 1$ für alle $x \to y$, dann folgt $\nu(x) = \gamma(x) \wedge 1$. Hierzu unterscheiden wir die beiden Fälle:

Gewinnposition: Führt ein Zug $x \to y$ zu $\nu(y) = 0$, so gilt $\nu(x) = 1$. In diesem Falle gilt $\gamma(y) = 0$, also $\gamma(x) \geq 1$, und somit $\nu(x) = \gamma(x) \wedge 1$.

Verlustposition: Führen alle Züge $x \to y$ zu $\nu(y) = 1$, so gilt $\nu(x) = 0$. In diesem Falle gilt $\gamma(y) \geq 1$, also $\gamma(x) = 0$, und somit $\nu(x) = \gamma(x) \wedge 1$.

Daraus folgt $\nu = \gamma \wedge 1$: Gäbe es $x_0 \in X$ mit $\nu(x_0) \neq \gamma(x_0) \wedge 1$, so gilt $x_0 \in X^\circ$, und es gibt einen Nachfolger $x_0 \to x_1$ mit $\nu(x_1) \neq \gamma(x_1) \wedge 1$. So fortfahrend erhalten wir einen unendlichen Weg $x_0 \to x_1 \to x_2 \to \dots$. Das widerspricht unserer Voraussetzung, dass der Graph G artinsch ist.

Das Spiel Nim: Formalisierung als Graph



$x_4 = 1$

$x_3 = 3$

$x_2 = 5$

$x_1 = 7$

Zustand $x = (x_0, x_1, x_2, \dots) \in \mathbb{N}^{(\mathbb{N})}$

Aktion $a = (x, r, s, y)$ $\begin{cases} (r,s) \in \mathbb{N}^2 \\ 1 \le s \le x_r \\ y = x - s\, e_r \end{cases}$

Zustand $y = (y_0, y_1, y_2, \dots) \in \mathbb{N}^{(\mathbb{N})}$

Aufgabe: Formulieren Sie dieses Spiel in Worten und als Graph.

Lösung: Nim ist ein Spiel für zwei Spieler; beide ziehen abwechselnd. Die Zustandsmenge ist $X = \mathbb{N}^{(\mathbb{N})}$. Aktionen $x \to y$ sind Paare $(r, s) \in \mathbb{N}^2$ mit $1 \le s \le x_r$ und $y = x - s\, e_r$. Terminal ist $x = 0$, aktiv sind alle $x \neq 0$. Wir betrachten das Normalspiel: Wer nicht mehr ziehen kann, verliert.

Das Spiel Nim: Formalisierung als Graph

Dieses Spiel ist sehr einfach, doch wenn Sie es zum ersten Mal sehen, werden Sie anfangs vermutlich wenig oder keine Struktur erkennen. Erinnern Sie sich an Ihre Erfahrung mit einzeiligem Nim! (Satz C1A)

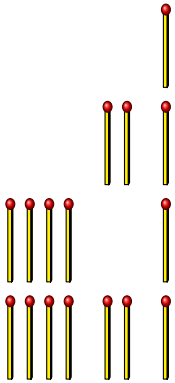
Zur Illustration spielen wir dies ausgiebig in Vorlesung oder Casino, damit Sie das Problem und seine elegante Lösung wirklich spüren. In hoffe diese experimentelle Spielphase ist gut investierte Zeit.

Manche Spezialfälle entdecken und verstehen Sie leicht im Spiel: Zum Beispiel ist $(x_1, x_2) \in \mathbb{N}^2$ mit $x_1 = x_2$ eine Verlustposition, denn der zweite Spieler kann jeden Zug des ersten spiegeln.

Eine allgemeine Position $(x_1, x_2, \dots, x_n) \in \mathbb{N}^n$ hingegen ist nicht so einfach zu durchschauen. Das müssen Sie selbst ausprobieren und persönlich erfahren! Dann können Sie die geniale Lösung würdigen.

Das Spiel Nim wurde gelöst von Charles L. Bouton: *Nim, a game with a complete mathematical theory*, Annals of Mathematics 3 (1901) 35–39. Das erklären wir in Satz C1E. Der Satz C1K von Sprague–Grundy überführt jedes neutrale kombinatorische Spiel in ein Nim-Spiel.

Das Spiel Nim: effiziente Lösung nach Charles Bouton 1901



$x_4 = 001_{\text{bin}}$

$x_3 = 011_{\text{bin}}$

$x_2 = 101_{\text{bin}}$

$x_1 = 111_{\text{bin}}$

Wir entwickeln x_i im Binärsystem

$$x_i = \textstyle\sum_{j=0}^{m} \langle x_i \rangle_j \, 2^j \quad \text{mit} \quad \langle x_i \rangle_j \in \{0, 1\}$$

und bilden die Spaltensumme

$$\langle s \rangle_j = \textstyle\sum_{i=0}^{n} \langle x_i \rangle_j \text{ rem } 2.$$

Binäre Summe ohne Übertrag (XOR):

$$s = \textstyle\sum_{j=0}^{m} \langle s \rangle_j \, 2^j =: x_0 \oplus x_1 \oplus \dots \oplus x_n$$

Satz C1E: effiziente Lösung des Nim-Spiels, Bouton 1901

Im Normalspiel sind die Verlustpositionen x genau die Null-Positionen:

(0) Ist $x \in \mathbb{N}^{(\mathbb{N})}$ eine Null-Position, also $x_0 \oplus x_1 \oplus \dots \oplus x_n = 0$, so führt jeder Zug $x \to y$ in eine Nicht-Null-Position, $y_0 \oplus y_1 \oplus \dots \oplus y_n \neq 0$.

(1) Ist x eine Nicht-Null-Position, $x_0 \oplus x_1 \oplus \dots \oplus \dots x_n \neq 0$, so existiert ein Zug $x \to y$ in eine Null-Position, also $y_0 \oplus y_1 \oplus \dots \oplus y_n = 0$.

Das Spiel Nim: effiziente Lösung nach Charles Bouton 1901

☺ Den Beweis führen wir allgemein für Satz C1F, noch besser Satz C1K. Die **binäre Summe ohne Übertrag** heißt auch **XOR** oder **Nim-Summe**. Dieser Algorithmus ist ebenso trickreich wie simpel: Er berechnet die Gewinnfunktion $\nu : X \to \{0, 1\}$ des Normalspiels, schnell und einfach.

☺ Der Zeitaufwand ist nur linear in der Bitlänge der Eingabe $x \in \mathbb{N}^n$:

$$\begin{aligned} \text{len} : \mathbb{N} \to \mathbb{N} \quad &: x \mapsto \min\{\, \ell \in \mathbb{N} \mid a < 2^\ell \,\} \\ \text{len} : \mathbb{N}^n \to \mathbb{N} &: x \mapsto \text{len}(x_1) + \dots + \text{len}(x_n) \end{aligned}$$

Linearer Aufwand ist das Beste, was wir hierzu je erhoffen können: Das Problem zu *lösen* ist kaum aufwändiger, als das Problem zu *lesen*!

☺ Boutons Lösung C1E enthüllt Gewinn- und Verlustpositionen. Der folgende Satz C1F zeigt zudem die Sprague–Grundy–Funktion und konstruiert explizit alle Gewinnzüge, also optimale Strategien!

Für das tatsächliche *Spielen* ist dies der entscheidende Schritt. Daher führe ich dies im folgenden Satz gebrauchsfertig aus.

Nim: starke Lösung nach Sprague 1935, Grundy 1939

Satz C1F: die Sprague–Grundy–Funktion des Nim-Spiels

Für Nim ist die Sprague–Grundy–Funktion $\gamma : X \to \mathbb{N}$ gegeben durch

$$\boxed{\gamma(x) = \bigoplus_{i\in\mathbb{N}} x_i = x_0 \oplus x_1 \oplus x_2 \oplus \dots .}$$

(0) Für jede Aktion $(r,s) : x \to y$ gilt $x_r \neq y_r$ und somit $\gamma(x) \neq \gamma(y)$.
(1) Zu $0 \le n < \gamma(x)$ existiert eine Aktion $(r,s) : x \to y$ mit $\gamma(y) = n$.

$1 = 001_{\text{bin}}$	$1 = 001_{\text{bin}}$	$\to$	$\cancel{2 = 010_{\text{bin}}}$	$\to$	$\cancel{3 = 011_{\text{bin}}}$	$\to$	$0 = 000_{\text{bin}}$
$3 = 011_{\text{bin}}$	$3 = 011_{\text{bin}}$	$\to$	$0 = 000_{\text{bin}}$	$\to$	$1 = 001_{\text{bin}}$	$\to$	$2 = 010_{\text{bin}}$
$5 = 101_{\text{bin}}$	$6 = 110_{\text{bin}}$	$\to$	$5 = 101_{\text{bin}}$	$\to$	$4 = 100_{\text{bin}}$	$\to$	$\cancel{7 = 111_{\text{bin}}}$
$7 = 111_{\text{bin}}$	$7 = 111_{\text{bin}}$	$\to$	$4 = 100_{\text{bin}}$	$\to$	$5 = 101_{\text{bin}}$	$\to$	$6 = 110_{\text{bin}}$
$0 = 000_{\text{bin}}$	$3 = 011_{\text{bin}}$	$\to$	$0 = 000_{\text{bin}}$	$\to$	$1 = 001_{\text{bin}}$	$\to$	$2 = 010_{\text{bin}}$

Satz C1F: formale Konstruktion

Sei $z := \gamma(x) \oplus n = 2^\ell + \sum_{k=0}^{\ell-1} z_k 2^k$. Wir wählen $r \in \mathbb{N}$ mit $\langle x_r \rangle_\ell = 1$.
Das garantiert $y_r := x_r \oplus z < x_r$ und ergibt $\gamma(y) = \gamma(x) \oplus z = n$.

Nim: starke Lösung nach Sprague 1935, Grundy 1939

Übung: Experimentieren Sie mit dem genialen Algorithmus des Satzes! Wenden Sie das Verfahren mehrfach an, dann beweisen sie den Satz.

☺ Sprague–Grundy C1F beinhaltet Boutons Lösung C1E dank Satz C1D: Jede Position $x \in X$ mit $\gamma(x) = 0$ ist eine Verlustposition, also $\nu(x) = 0$. Jede Position $x \in X$ mit $\gamma(x) \ge 1$ ist eine Gewinnposition, also $\nu(x) = 1$.

☺ Wir werden diese geniale Methode in Satz C1K perfektionieren. Der Beweis ist jeweils leicht und ich formuliere ihn detailliert aus.

☹ Wenn Sie zuerst den Beweis lesen, sagt er Ihnen noch allzu wenig. Der Beweis ist einfach genial und genial einfach: Sie können ihn Schritt für Schritt leicht durcharbeiten, doch dabei besteht die Gefahr, dass der zündende Funke nicht überspringt. Das wäre hier besonders schade!

☺ Zunächst empfehle ich, den Algorithmus in zahlreichen Bei-Spielen zu erproben. Durch eigenständiges Probieren wird Ihnen schnell klar, wie das Verfahren funktioniert und wie ein Beweis aussehen könnte. So können Sie selbst den Beweis entdecken, entwickeln und verstehen.

Nim: starke Lösung nach Sprague 1935, Grundy 1939

Beweis: Aussage (0) ist klar, da $(\mathbb{N}, \oplus)$ eine Gruppe ist, sogar abelsch. Abstrakt gesehen nutzen wir nur die Kürzbarkeit der Verknüpfung $\oplus$.

Anschaulich: Jeder Zug $(r,s) : x \to y$ ändert nur die Koordinate x_r zu y_r. Im Nim-Spiel gilt $y_r < x_r$, wir benötigen nur $y_r \neq x_r$. In der Nim-Summe $\gamma(x) = x_1 \oplus \dots \oplus x_n$ ändert sich mindestens ein Bit, also $\gamma(y) \neq \gamma(x)$.

(1) Vom Start $m = \gamma(x)$ zum gewünschten Ziel $n < m$ steht das höchste zu ändernde Bit an der Stelle ℓ. Dank $m > n$ gilt $\langle m \rangle_\ell = 1$ und $\langle n \rangle_\ell = 0$. Demnach existiert mindestens eine Koordinate $r \in \mathbb{N}$ mit $\langle x_r \rangle_\ell = 1$. Genauer existiert eine ungerade Anzahl dieser Wahlmöglichkeiten.

Diese Wahl r mit $\langle x_r \rangle_\ell = 1$ stellt sicher, dass $y_r := x_r \oplus z < x_r$ gilt. (Im Falle $\langle x_r \rangle_\ell = 0$ wäre $x_r \oplus z > x_r$ im Nim-Spiel kein erlaubter Zug.) Diese Wahl r und $s = x_r - y_r \ge 1$ definieren die Aktion $(r,s) : x \to y$ mit $\gamma(y) = \bigoplus_{i\in\mathbb{N}} y_i = (\bigoplus_{i\in\mathbb{N}} x_i) \oplus z = \gamma(x) \oplus z = n$, wie gewünscht.

Die beiden Eigenschaften (0–1) garantieren, dass die Funktion $\gamma : X \to \mathbb{N}$ tatsächlich die Sprague–Grundy–Funktion des Nim-Spiels G ist. QED

Nim: starke Lösung nach Sprague 1935, Grundy 1939

Aufgabe: Vorgelegt sei eine Position $x \in X = \mathbb{N}^{(\mathbb{N})}$ im Nim-Spiel.
(1) Wie finden Sie von x aus *alle* möglichen Gewinnzüge $(r,s) : x \to y$?
(2) Wie finden Sie zu $0 \le n < \gamma(x)$ *alle* Züge $(r,s) : x \to y$ mit $\gamma(y) = n$?
Gehen Sie den Beweis sorgsam durch und zeigen Sie folgenden Zusatz:

Satz C1F: Vollständigkeit

Die genannte Konstruktion liefert *alle* Züge $x \to y$ mit $\gamma(y) < \gamma(x)$.

Lösung: Die Wahl einer Zeile $r \in \mathbb{N}$ mit $\langle x_r \rangle_\ell = 1$ ist nicht nur hinreichend, sondern auch notwendig für $y_r := x_r \oplus z < x_r$.

Satz C1F und sein Beweis beruhen auf der folgenden Beobachtung:

Lemma C1G: Umformulierung der mex–Eigenschaft

Vorgelegt sei ein Graph $G = (X, A, \sigma, \tau)$ mit einer Funktion $\gamma : X \to \mathbb{N}$. Dann sind (0,1) äquivalent zu $\gamma(x) = \operatorname{mex}\{\gamma(y) \mid x \to y\}$ für alle $x \in X$.

Übung: Erklären Sie sorgfältig „$\Rightarrow$“ und „$\Leftarrow$“ zur Wiederholung.

Nim: algebraischer Hintergrund

Wir nutzen die **Binärentwicklung**: Jede natürliche Zahl $z \in \mathbb{N}$ schreibt sich eindeutig als Summe $z = \sum_{k\in\mathbb{N}} z_k 2^k$ mit Ziffern $(z_k)_{k\in\mathbb{N}} \in \{0,1\}^{(\mathbb{N})}$.

$$\begin{tikzcd} \mathbb{N} \arrow[r, shift left, "\langle - \rangle"] & \{0,1\}^{(\mathbb{N})} \arrow[l, shift left, "\cong"] \arrow[r, shift left, "\cong"] & \mathbb{F}_2[T] \arrow[l, shift left, "|-|"] \\ \sum_{k\in\mathbb{N}} z_k 2^k \arrow[r, mapsto, shift left] & (z_k)_{k\in\mathbb{N}} \arrow[l, mapsto, shift left] \arrow[r, mapsto, shift left] & \sum_{k\in\mathbb{N}} z_k T^k \arrow[l, mapsto, shift left] \end{tikzcd}$$

Diese Bijektion $\mathbb{N} \cong \mathbb{F}_2[T]$ ist kein Isomorphismus: $(\mathbb{N},+) \ncong (\mathbb{F}_2[T],+)$.
Algebraisch: $(\mathbb{F}_2[T],+)$ ist eine Gruppe, hingegen $(\mathbb{N},+)$ nur ein Monoid.
Arithmetisch: In $(\mathbb{N},+)$ wird mit Übertrag addiert, in $(\mathbb{F}_2[T],+)$ ohne.
Ich stelle dies hier nebeneinander, um den Kontrast zu betonen.

Wir definieren auf $\mathbb{N}$ die **binäre Addition ohne Übertrag** durch

$$\boxed{\oplus : \mathbb{N}\times\mathbb{N} \to \mathbb{N} : (x,y) \mapsto |\langle x\rangle + \langle y\rangle|.}$$

Wir definieren $\bigoplus_{i=0}^n x_i = x_0 \oplus x_1 \oplus \cdots \oplus x_n$ wie üblich und allgemein für $x \in \mathbb{N}^{(I)}$ mit endlichem Träger $J \subseteq I$ ebenso $\bigoplus x := \bigoplus_{i\in I} x_i := \bigoplus_{i\in J} x_i$.

Nim: algebraischer Hintergrund

In der Informatik ist dies eine sehr vertraute Operation: bitweises XOR! Sie ist schnell und einfach zu berechnen, für Mensch wie für Computer.

Auch in der Mathematik ist $(\mathbb{N},\oplus) \cong (\mathbb{F}_2[T],+)$ eine vertraute Struktur, nämlich eine abelsche Gruppe, genauer sogar ein $\mathbb{F}_2$–Vektorraum.

Aufgrund von Satz C1E nennen manche Autor:innen (im Rahmen der Spieltheorie) diese Verknüpfung $\oplus : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ auch **Nim-Summe**.

Fun fact: Die Summanden $x_1, x_2, \dots, x_n \in \mathbb{N}$ sind natürliche Zahlen, engl. *natural numbers*. Im Kontext des Nim-Spiels nennen manche Autor:innen dies auch *nimbers*, als Wortspiel und als Erinnerung.

Auch die Polynommultiplikation überträgt sich zu $x \odot y = |\langle x\rangle \cdot \langle y\rangle|$.
So wird $(\mathbb{N},\oplus,\odot)$ eine isomorphe Kopie des Polynomrings $(\mathbb{F}_2[T],+,\cdot)$.

Auf den ersten Blick scheinen Polynomringe vielleicht kompliziert, selbst der einfachste Vertreter $\mathbb{F}_2[T]$, doch das ist keine objektive Schwierigkeit:

Neutral betrachtet ist die Arithmetik in $\mathbb{F}_2[T]$ viel einfacher als in $\mathbb{N}$, denn in $\mathbb{F}_2[T]$ wird **ohne Übertrag** addiert und multipliziert.

Nim: algebraischer Ausblick

Aus mathematischer Sicht ist die Definition von $(\mathbb{N},\oplus)$ nicht tiefsinnig. Überaus bemerkenswert ist hingegen, dass die binäre Addition $(\mathbb{N},\oplus)$ und die Anordnung $(\mathbb{N},<)$ für Spiele so trickreich zusammenarbeiten.

L'algèbre est généreuse, elle donne souvent plus qu'on lui demande.
[Die Algebra ist großzügig, sie gibt oft mehr, als wir von ihr verlangen.]
Jean Le Rond d'Alembert (1717–1783)

Getreu diesem Leitspruch gibt es neben der Nim-Addition auch eine Nim-Multiplikation! J.H. Conway entdeckte sie 1976 als Spezialfall seiner surrealen Zahlen. Mehr hierzu finden Sie im Buch von A.N. Siegel, *Combinatorial Game Theory*, VI.5 (endlich) und VII.4 (transfinit).

Definition C1H: Nim-Arithmetik nach Conway, 1976

Für alle natürlichen Zahlen $\alpha,\beta \in \mathbb{N}$ definieren wir rekursiv

$$\alpha \oplus \beta := \operatorname{mex}(\{\alpha' \oplus \beta \mid \alpha' < \alpha\} \cup \{\alpha \oplus \beta' \mid \beta' < \beta\}),$$
$$\alpha \otimes \beta := \operatorname{mex}\{(\alpha' \otimes \beta) \oplus (\alpha \otimes \beta') \oplus (\alpha' \otimes \beta') \mid \alpha' < \alpha,\ \beta' < \beta\}).$$

Diese Konstruktion setzt sich ebenso auf alle Ordinalzahl fort.

Nim: algebraischer Ausblick

Übung: (1) Damit ist $(\mathbb{N},\oplus,0,\otimes,1)$ ein kommutativer Ring.
(2) $(\mathbb{N}_{<2},\oplus,0,\otimes,1)$ ist ein Körper, isomorph zu $(\mathbb{F}_2,+,0,\cdot,1)$.
(3) $(\mathbb{N}_{<4},\oplus,0,\otimes,1)$ ist ein Körper, isomorph zu $(\mathbb{F}_4,+,0,\cdot,1)$.
Scheiben Sie die Additions- und Multiplikationstabellen aus; eine Lösung finden Sie bei Siegel auf S.217 und unter `en.wikipedia.org/wiki/Nimber`.
Die nächste abgeschlossene Teilmenge ist $\mathbb{N}_{<16}$. Allgemein gilt:

Satz C1I: Nim-Arithmetik nach Conway, 1976

Nim-Arithmetik $(\mathbb{N},\oplus,0,\otimes,1)$ ist ein Körper. Für jede Fermat–Potenz $K = 2^{2^n}$ ist $(\mathbb{N}_{<K},\oplus,0,\otimes,1)$ ein Teilkörper isomorph zu $(\mathbb{F}_{2^{2^n}},+,0,\cdot,1)$.

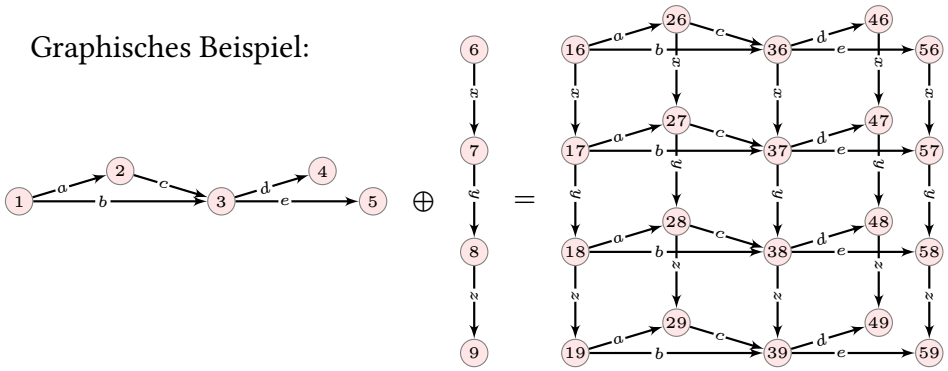
Die Klasse **Ord** aller Ordinalzahlen bildet einen Körper $(\mathbf{Ord},\oplus,0,\otimes,1)$. Dieser ist zudem algebraisch abgeschlossen. Der kleinste algebraisch abgeschlossene Teilkörper ist darin die Menge ω^{ω^ω}.

Wir werden diese faszinierende Struktur hier nicht weiter verfolgen, doch als Hinweis und Ausblick kann ich sie mir auch nicht verkneifen: Seit Conways Entdeckung offenbart die kombinatorische Spieltheorie erstaunliche Verbindungen zur Logik, Mengenlehre, Algebra, Analysis!

Summe von Spielen

Wir spielen mehrere Spiele G_i parallel, also gleichzeitig nebeneinander. Der ziehende Spieler darf sich aussuchen, in welchem Spiel G_i er zieht. Dies fassen wir zusammen zu einem gemeinsamen Spiel $G = \bigoplus_{i\in I} G_i$.

Graphisches Beispiel:



Graphische Illustration der Summe $G = G_1 \oplus G_2$: Ein Zug in G ist entweder ein Zug in G_1 (horizontal) oder ein Zug in G_2 (vertikal).
Aufgabe: Formulieren Sie explizit die Definition von $G = \bigoplus_{i\in I} G_i$.

Summe von Spielen

⚠ Es gibt mehrere Möglichkeiten, Produkte von Graphen zu definieren. Gegen mögliche Verwirrung hilft wie immer nur eine präzise Definition:

Definition C1J: Produkt und Summe von Spielen

Gegeben sei eine Familie $(G_i)_{i\in I}$ von Graphen $G_i = (X_i, A_i, \sigma_i, \tau_i)$.
Ihr **Produkt** ist der Graph $P = \prod_{i\in I} G_i = (X, A, \sigma, \tau)$ mit

$$X = \textstyle\prod_{i\in I} X_i = \{\, x : I \to \bigcup_{i\in I} X_i : i \mapsto x_i \mid x_i \in X_i \,\},$$
$$A = \{\, (x, i, a_i, y) \mid x, y \in X,\ i \in I,\ a_i : x_i \to y_i,\ \forall j \neq i : x_j = y_j \,\}$$

sowie den Projektionen $\sigma(x, i, a_i, y) = x$ und $\tau(x, i, a_i, y) = y$.

Die **Summe** $S = \bigoplus_{i\in I} G_i = (X', A', \sigma', \tau')$ ist der volle Teilgraph $S \leq P$ der Zustände $x \in X$ mit endlichem Träger $\operatorname{supp}(x) := \{\, i \in I \mid x_i \in X_i^\circ \,\}$.

Der Träger $\operatorname{supp}(x) \subseteq I$ indiziert aktive Koordinaten $i \in I$ mit $x_i \in X_i^\circ$. Ist I endlich, so ist das Produkt gleich der Summe, doch in unendlichen Summen verlangen wir, dass nur endlich viele Koordinaten aktiv sind.

Erinnerung: Produkt und Summe von Gruppen

Zu jedem Index $i \in I$ sei $G_i = (X_i, +_i, 0_i, -_i)$ eine abelsche Gruppe mit Grundmenge X_i und darauf der Addition $+_i : X_i \times X_i \to X_i$ mit neutralem Element $0_i \in X_i$ und Negation $-_i : X_i \to X_i$.

Das **Produkt** $P = \prod_{i\in I} G_i := (X, +, 0, -)$ hat als Grundmenge

$$X = \textstyle\prod_{i\in I} X_i = \{\, x : I \to \bigcup_{i\in I} X_i : i \mapsto x_i \mid x_i \in X_i \,\}.$$

Wir definieren koordinatenweise die Addition $x + y = (x_i +_i y_i)_{i\in I}$, das neutrale Element $0 = (0_i)_{i\in I}$ und die Negation $-x = (-_i\, x_i)_{i\in I}$.

Wir definieren den **Träger** $\operatorname{supp} : P \to \mathfrak{P}I : x \mapsto \{\, i \in I \mid x_i \neq 0_i \,\}$. Einschränkung definiert die **Summe** der Gruppen $(G_i)_{i\in I}$ vermöge

$$S = \textstyle\bigoplus_{i\in I} G_i := \{\, x \in P \mid \operatorname{supp}(x) \text{ endlich} \,\}.$$

Übung: Mit dieser Konstruktion ist $(P, +, 0, -)$ eine abelsche Gruppe und $S \leq P$ eine Untergruppe. Dasselbe gilt für Ringe; für $\sharp I = \infty$ hat das Einselement $1 = (1_i)_{i\in I}$ keinen endlichen Träger, also $1 \in P$, aber $1 \notin S$.

Erinnerung: Produkt und Summe von Monoiden

Allgemeiner gelingt diese Konstruktion für abelsche Monoide $(X_i, +_i, 0_i)$, noch allgemeiner Tripel aus einer Menge X_i mit innerer Verknüpfung $+_i : X_i \times X_i \to X_i$ und idempotenten Element $0_i \in X_i$, also $0_i +_i 0_i = 0_i$.

Im Falle $G_i = G$ für alle $i \in I$ erhalten wir die (eingeschränkte) Potenz:

$$P = G^I \quad := \{\, x : I \to G \,\}$$
$$S = G^{(I)} := \{\, x : I \to G \mid \operatorname{supp}(x) \text{ endlich} \,\}$$

Nur auf letzterem haben wir die Summenabbildung

$$\sum : G^{(I)} \to G : (g_i)_{i\in I} \mapsto \textstyle\sum_{i\in I} g_i.$$

Aus dem abelschen Monoid $(\mathbb{N}, +, 0)$ erhalten wir das Monoid $\mathbb{N}^{(\mathbb{N})}$ mit koordinatenweiser Addition $+$ und hierauf $\sum : \mathbb{N}^{(\mathbb{N})} \to \mathbb{N}$.

Beispiel: Für den Sprague–Grundy–Satz nutzen wir insbesondere: Aus der abelschen Gruppe $(\mathbb{N}, \oplus, 0)$ erhalten wir die Gruppe $\mathbb{N}^{(\mathbb{N})}$ mit koordinatenweiser Addition $\oplus$ und hierauf $\oplus : \mathbb{N}^{(\mathbb{N})} \to \mathbb{N}$.

Der Sprague–Grundy–Satz

Satz C1K: Sprague 1935, Grundy 1939

Gegeben seien lokal-endliche artinsche Graphen $G_i = (X_i, A_i, \sigma_i, \tau_i)$. Dann ist auch ihre Summe $G = \bigoplus_{i\in I} G_i$ lokal-endlich und artinsch, und für ihre Sprague–Grundy–Funktion gilt $\gamma(x) = \bigoplus_{i\in I} \gamma_i(x_i)$.

Genauer: (0) Für jede Aktion $(i, a_i) : x \to y$ in G gilt $a_i : x_i \to y_i$ in G_i. Für die Grundy–Zahlen folgt $\gamma_i(x_i) \neq \gamma_i(y_i)$ und somit $\gamma(x) \neq \gamma(y)$.

(1) Zu $0 \le n < \gamma(x)$ existiert eine Aktion $(i, a_i) : x \to y$ mit $\gamma(y) = n$. Insbesondere finden wir so Gewinnzüge $x \to y$ mit $\gamma(y) = 0$.

Den Satz beweisen wir durch die Konstruktion solcher Züge:

Satz C1K: formale Konstruktion

(1) Wir lösen $\gamma(x) \oplus z = n$ durch $z = \gamma(x) \oplus n = 2^\ell + \sum_{k=0}^{\ell-1} z_k 2^k$ und wählen $i \in I$ mit $\langle \gamma_i(x_i) \rangle_\ell = 1$. Somit gilt $\gamma_i(x_i) \oplus z < \gamma_i(x_i)$.

In G_i existiert eine Aktion $a_i : x_i \to y_i$ mit $\gamma_i(y_i) = \gamma_i(x_i) \oplus z$. In G erhalten wir $(i, a_i) : x \to y$ mit $\gamma(y) = \gamma(x) \oplus z = n$.

Der Sprague–Grundy–Satz

Erst dieser abschließende Satz motiviert und rechtfertigt die anfangs eingeführte Sprague–Grundy–Funktion: Genau hier liegt ihr Nutzen!

☺ Als prototypisches Beispiel haben wir oben in Satz C1F das Nim-Spiel eingehend diskutiert; dort ist alles besonders einfach dank $\gamma_i(x_i) = x_i$.

☺ Der Satz ist konstruktiv als gebrauchsfertiger Algorithmus formuliert. Sie können dieses Verfahren also direkt gewinnbringend anwenden… und sich dann nachfolgend den allgemeinen Beweis erarbeiten.

☹ Wenn Sie zuerst den Beweis lesen, sagt er Ihnen noch allzu wenig. Der Beweis ist einfach genial und genial einfach: Sie können ihn Schritt für Schritt leicht durcharbeiten, doch dabei besteht die Gefahr, dass der zündende Funke nicht überspringt. Das wäre hier besonders schade!

☺ Zunächst empfehle ich, den Algorithmus in zahlreichen Bei-Spielen zu erproben. Durch eigenständiges Probieren wird Ihnen schnell klar, wie das Verfahren funktioniert und wie ein Beweis aussehen könnte. So können Sie selbst den Beweis entdecken, entwickeln und verstehen.

Der Sprague–Grundy–Satz

Beweis: Zunächst ist G lokal-endlich und artinsch: Gäbe es in G einen unendlichen Weg $x^0 \to x^1 \to x^2 \to \dots$, so ist $J = \operatorname{supp}(x^0)$ endlich, und in mindestens einer Koordinate $i \in J$ finden wir einen unendlichen Weg $x_i^{n_0} \to x_i^{n_1} \to x_i^{n_2} \to \dots$, und somit wäre G_i nicht artinsch. Widerspruch!

Für jeden Zustand $x \in X$ ist $\operatorname{supp}(x)$ endlich, also $\gamma(x) := \bigoplus_{i\in I} \gamma_i(x_i)$ wohldefiniert, denn für fast alle $i \in I$ gilt $x_i \in \partial X_i$ und somit $\gamma_i(x_i) = 0$.

Die Aussage (0) ist klar. (Führen Sie dies zur Übung sorgfältig aus!)

(1) Die Frage lautet: Warum können wir $i \in I$ wie angegeben wählen? Vom Start $m = \gamma(x)$ zum Ziel $n < m$ steht das höchste zu ändernde Bit an der angegebenen Stelle ℓ. Dank $m > n$ gilt $\langle m \rangle_\ell = 1$ und $\langle n \rangle_\ell = 0$. Demnach existiert mindestens eine Koordinate $i \in \mathbb{N}$ mit $\langle \gamma_i(x_i) \rangle_\ell = 1$.

Dies garantiert $\gamma_i(x_i) \oplus z < \gamma_i(x_i)$. In G_i existiert demnach eine Aktion $a_i : x_i \to y_i$ mit $\gamma_i(y_i) = \gamma_i(x_i) \oplus z$. In G erhalten wir $(i, a_i) : x \to y$ mit $\gamma(y) = \bigoplus_{i\in\mathbb{N}} \gamma_i(y_i) = (\bigoplus_{i\in\mathbb{N}} \gamma_i(x_i)) \oplus z = \gamma(x) \oplus z = n$, wie gewünscht.

Die beiden Eigenschaften (0,1) garantieren, dass $\gamma : X \to \mathbb{N}$ tatsächlich die Sprague–Grundy–Funktion des Spiels G ist (siehe Lemma C1G). $\boxed{\text{QED}}$

Der Sprague–Grundy–Satz

Aufgabe: Vorgelegt sei eine Position $x \in X$ im Spiel $G = (X, A, \sigma, \tau)$.
(1) Wie finden Sie von x aus *alle* möglichen Gewinnzüge (x, i, a_i, y)?
(2) Wie finden Sie zu $0 \le n < \gamma(x)$ *alle* Züge (x, i, a_i, y) mit $\gamma(y) = n$?

Lösung: Frage (1) ist ein Spezialfall von (2), nämlich für $n = 0$.

Der Zug (x, i, a_i, y) im Spiel G bedeutet $a_i : x_i \to y_i$ im Spiel G_i. Somit gilt $\gamma(y) = \gamma(x) \oplus z$ mit $z = \gamma_i(x_i) \oplus \gamma_i(y_i)$. Wir wollen $\gamma(y) = n$.

Wir berechnen also zunächst $z := \gamma(x) \oplus n = 2^\ell + \sum_{k=0}^{\ell-1} z_k 2^k$ und suchen nun alle Züge $a_i : x_i \to y_i$ mit $\gamma_i(y_i) = \gamma_i(x_i) \oplus z$.

Im Falle $\langle \gamma_i(x_i) \rangle_\ell = 0$ gilt $\gamma_i(x_i) \oplus z > \gamma_i(x_i)$: Geeignete Züge $a_i : x_i \to y_i$ können im Spiel G_i existieren, sie sind möglich, aber nicht zwingend.

Im Falle $\langle \gamma_i(x_i) \rangle_\ell = 1$ gilt $\gamma_i(x_i) \oplus z < \gamma_i(x_i)$: Nach Definition der Grundy–Zahl γ_i existiert im Spiel G_i mindestens ein Zug $a_i : x_i \to y_i$.

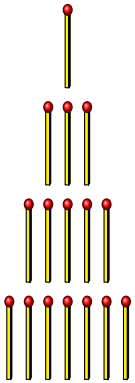
(Der Beweis zeigt, dass es mindestens eine solche Koordinate $i \in I$ gibt, genauer existiert immer eine ungerade Anzahl solcher Koordinaten.)

Wir finden so alle Züge (x, i, a_i, y) mit $\gamma_i(y_i) = \gamma_i(x_i) \oplus z$, also $\gamma(y) = n$.

Nimm eins oder zwei, nun mit mehreren Zeilen

Wir spielen „Nimm eins oder zwei“, nun parallel mit mehreren Zeilen.
Hier können wir den Sprague–Grundy–Satz C1K anwenden und testen.

Aufgabe: Nennen Sie für die gezeigte Position *alle* Gewinnzüge!



$x_4 = 1 \quad \Rightarrow \quad \gamma_4(x_4) = 01_{\text{bin}} \overset{?}{\to} 11_{\text{bin}} = \gamma_4(y_4)$ unmöglich

$x_3 = 3 \quad \Rightarrow \quad \gamma_3(x_3) = 00_{\text{bin}} \overset{?}{\to} 10_{\text{bin}} = \gamma_3(y_3) \quad \Leftarrow \quad y_3 = 2$

zusätzlicher, erhöhender Gewinnzug!

$x_2 = 5 \quad \Rightarrow \quad \gamma_2(x_2) = 10_{\text{bin}} \overset{?}{\to} 00_{\text{bin}} = \gamma_2(y_2) \quad \Leftarrow \quad y_2 = 3$

garantierter, reduzierender Gewinnzug

$x_1 = 7 \quad \Rightarrow \quad \gamma_1(x_1) = 01_{\text{bin}} \overset{?}{\to} 11_{\text{bin}} = \gamma_1(y_1)$ unmöglich

Gewinnposition? $\quad \gamma(x) = 10_{\text{bin}} \overset{!}{\to} 00_{\text{bin}} = \gamma(y) \quad$ Züge?

Nimm eins oder zwei, nun mit mehreren Zeilen

Wie spielen wir optimal? Wir nutzen den Sprague–Grundy–Satz C1K!
Damit finden wir *einen* Gewinnzug, bei genauem Hinsehen sogar *alle*.

Lösung: Wir untersuchen den gezeigten Spielzustand $x = (7, 5, 3, 1)$.
Dank Satz C1A kennen wir die Grundy–Zahlen $\gamma_i(x_i) = x_i \text{ rem } 3$.
Dank Satz C1K berechnen wir damit $\gamma(x) = 1 \oplus 2 \oplus 0 \oplus 1 = 2$.
Der vorgegebene Zustand x ist also ein Gewinnzustand.

Der Sprague–Grundy–Satz C1K konstruiert dazu einen Gewinnzug:
Wir wollen $\gamma(x) = 2$ reduzieren auf $0 = \gamma(y) = \gamma(x) \oplus z$, hier also $z = 2$.
Dazu reduzieren wir $\gamma_2(x_2) = 2$ auf $\gamma_2(y_2) = 0$ vermöge $x_2 = 5 \to y_2 = 3$.
Satz C1K liefert so *einen* Gewinnzug, wie versprochen, ganz explizit.

⚠ Satz C1K behauptet jedoch nicht, *alle* Gewinnzüge zu konstruieren.
In unserem Beispiel gibt es einen weiteren, versteckten Gewinnzug:
Wir erhöhen $\gamma_3(x_3) = 0$ auf $\gamma_3(y_3) = 2$ vermöge $x_3 = 3 \to y_3 = 2$.
(In Nim ist die Erhöhung $0 \to 2$ kein legaler Zug, hier ist es möglich.)

☺ Erst damit sind wirklich alle Gewinnzüge gefunden.
Damit ist die Aufgabe gelöst, elegant und effizient.

Vergleich von Nim-Spiel und Sprague–Grundy–Satz

Der Sprague–Grundy–Satz C1K berechnet effizient die Gewinn- und Verlustpositionen. Zudem liefert er eine konkrete Konstruktion von Gewinnzügen, allgemein von Zügen $(i, a_i) : x \to y$ mit $\gamma(y) < \gamma(x)$.
Die Konstruktion liefert alle Züge, die zudem $\gamma_i(y_i) < \gamma_i(x_i)$ erfüllen.

Für Nim liefert der Sprague–Grundy–Satz C1F eine stärkere Aussage:
Die genannte Konstruktion liefert *alle* Züge $x \to y$ mit $\gamma(y) < \gamma(x)$,
denn hier gilt $\gamma_i(x_i) = x_i$, und jeder Zug $x_i \to y_i$ ist reduzierend.
Insbesondere ist die Anzahl solcher Züge in Nim immer ungerade.

In einer allgemeinen Summe $G = \bigoplus_{i \in I} G_i$ sind neben den reduzierenden Zügen $(i, a_i) : x \to y$ mit $\gamma_i(x_i) > \gamma_i(y_i)$ manchmal auch erhöhende Züge mit $\gamma_i(x_i) < \gamma_i(y_i)$ möglich. Hierüber macht Satz C1K keine Aussagen, daher auch nicht über Anzahl oder Parität der gesuchten Züge.

Wenn es nur darum geht, *irgendeinen* Gewinnzug zu finden, dann genügt die allgemeine Konstruktion des Satzes C1K. Die ambitioniertere Frage nach *allen* Gewinnzügen erfordert eine etwas präzisese Untersuchung.
Genau dies illustriert dieses schöne und gut gewählte Beispiel.

Vergleich von Nim-Spiel und Sprague–Grundy–Satz

Slogan: *Der Sprague–Grundy–Satz überführt jedes Spiel in ein Nim-Spiel.*
Manche sagen sogar: *in ein äquivalentes Nim-Spiel.* Was heißt das genau?

Der Sprague–Grundy–Satz konstruiert und nutzt den „Morphismus“

$$\tilde{\gamma} : G = \bigoplus_{i \in I} G_i \to \mathbb{N}^{(I)} = \bigoplus_{i \in I} \mathbb{N} : (x_i)_{i \in I} \mapsto (\gamma_i(x_i))_{i \in I}.$$

von einer beliebigen Summe von Spielen zum entsprechenden Nim-Spiel.

0 Verlustpositionen in G werden zu Verlustpositionen in Nim.

1 Gewinnpositionen in G werden zu Gewinnpositionen in Nim.

2 Gewinnzüge in Nim übersetzen sich zurück zu Gewinnzügen in G.

Allgemeiner: Die Sprague–Grundy–Zahl wird erhalten, und reduzierende Züge in Nim übersetzen sich zurück zu reduzierenden Zügen in G.

Das genügt meist. Die Rückübersetzung ist allerdings nicht surjektiv:
Es kann Gewinnzüge in G geben, die nicht von Nim-Zügen herkommen.

Der Sprague–Grundy–Morphismus ist so gesehen kein Isomorphismus;
er vereinfacht etwas und lässt dazu einige unwesentliche Details weg.

Charles Bouton 1901: Wie alles begann...

NIM, A GAME WITH A COMPLETE MATHEMATICAL THEORY.

By Charles L. Bouton.

The game here discussed has interested the writer on account of its seeming complexity, and its extremely simple and complete mathematical theory.* The writer has not been able to discover much concerning its history, although certain forms of it seem to be played at a number of American colleges, and at some of the American fairs. It has been called Fan-Tan, but as it is not the Chinese game of that name, the name in the title is proposed for it.

1. Description of the Game. The game is played by two players, A and B. Upon a table are placed three piles of objects of any kind, let us say counters. The number in each pile is quite arbitrary, except that it is well to agree that no two piles shall be equal at the beginning. A play is made as follows:—The player selects one of the piles, and from it takes as many counters as he chooses; one, two, . . ., or the whole pile. The only essential things about a play are that the counters shall be taken from a single pile, and that at least one shall be taken. The players play alternately, and the player who takes up the last counter or counters from the table wins.

It is the writer's purpose to prove that if one of the players, say A, can leave one of a certain set of numbers upon the table, and after that plays without mistake, the other player, B, *cannot* win. Such a set of numbers will be called a *safe combination*. In outline the proof consists in showing that if A leaves a safe combination on the table, B at his next move cannot leave a safe combination, and whatever B may draw, A at his next move can again leave a safe combination. The piles are then reduced, A always leaving a safe combination, and B never doing so, and A must eventually take the last counter (or counters).

2. Its Theory. A *safe combination* is determined as follows: Write the number of the counters in each pile in the binary scale of notation,† and place these numbers in three horizontal lines so that the units are in the same vertical column. If then the sum of *each* column is 2 or 0 (*i. e.* congruent to 0, mod. 2), the set of numbers forms a safe combination.

Literatur zur kombinatorischen Spieltheorie

Die Originalartikel sind online erhältlich und noch immer schön zu lesen:

📖 Charles L. Bouton: *Nim, a game with a complete mathematical theory.* Annals of Mathematics 3 (1901) 35–39. doi.org/10.2307/1967631

Roland P. Sprague: *Über mathematische Kampfspiele.* Tôhoku Math. 41 (1935) 438–444. www.jstage.jst.go.jp/article/tmj1911/41/0/41_0_438/_pdf

Roland P. Sprague: *Über zwei Abarten von Nim.* Tôhoku Math. 43 (1937) 451–454. www.jstage.jst.go.jp/article/tmj1911/43/0/43_0_351/_pdf

Patrick M. Grundy: *Mathematics and Games.* Eureka 2 (1939) 6–8

Kombinatorische Spieltheorie ist heute ein riesiges Forschungsgebiet. Die monumentalen Klassiker WW und ONAG und das Lehrbuch CGT:

📖 Elwyn R. Berlekamp, John H. Conway, Richard K. Guy:
Winning Ways for Your Mathematical Plays. A K Peters 2001-2004
Gewinnen: Strategien für mathematische Spiele. Vieweg 1985–1986

John H. Conway: *On Numbers and Games.* A K Peters 2000

Aaron N. Siegel: *Combinatorial Game Theory.* AMS 2013

Roland Sprague 1935: Einleitung

Über mathematische Kampfspiele,

von

R. Sprague in Berlin-Charlottenburg.

1. Gegenstand dieser Arbeit sind Spiele mit folgendem Partieverlauf: eine Anfangsstellung wird nach einer beschränkten Anzahl von abwechselnden Zügen zweier Personen in eine Endstellung übergeführt, die keinen Zug mehr zulässt und den Sieg einer der beiden Personen ergibt.

Ohne Beschränkung der Allgemeinheit darf angenommen werden, dass der Sieg dem Partner zufällt, der die Partie beendet. Man braucht nur alle Endstellungen zu verbieten, in denen der zuletzt Ziehende verliert.

Diese Spiele mögen kurz als "Kampfspiele" bezeichnet werden.

2. E. Lasker(1), von dem der Name "mathematische Kampfspiele" herrührt, hat diese Spiele methodisch untersucht und bemerkt, dass ihre Stellungen in zwei Klassen zerfallen, nämlich "Gewinnstellungen" und "Verluststellungen". In Gewinnstellungen, kurz: "G", kann *der* Spieler den Sieg erzwingen, der am Zuge ist, in Verluststellungen, "V", der andere. Das Gewinnproblem eines Kampfspiels ist mit der Kenntnis seiner V erledigt: wer eine V herbeiführt, gewinnt, indem er seinem Gegner stets wieder eine V hinterlässt.

3. Zur Erläuterung diene der Hinweis auf ein altbekanntes Kampfspiel. Von einem Haufen von Dingen werden in jedem Zuge eine beschränkte Anzahl fortgenommen, nämlich mindestens 1, höchstens m. Jede Stellung ist dann durch die Anzahl der vorhandenen Dinge gekennzeichnet. Als V erweisen sich die ganzzahligen Vielfachen von $m+1$; denn wer eine Stellung $k(m+1)$ erreicht, gelangt bei beliebigen Zügen des Gegners schrittweise zu $(k-1)\cdot(m+1)$, $(k-2)\cdot(m+1)$ und so weiter bis zur Endstellung 0 und gewinnt. Jede andere Stellung ist G, weil sie in einem Zuge zu einer V gemacht werden kann.

Roland Sprague 1935: Hauptsatz

Satz III: Schreibt man die Zahlen $a, b, c, \ldots$ dyadisch untereinander, in einem Schema wie zur Addition dekadischer Zahlen, so lässt sich eine dyadische Zahl R bilden, deren Ziffern 0 oder 1 heissen, je nachdem die entsprechende Kolonne des Schemas eine gerade oder eine ungerade Anzahl von Einsen aufweist. Der Wert von R ist der Rang der Stellung $a, b, c, \ldots$ im Nim.

Beweis: Wegen der Eindeutigkeit der Zuordnung in Satz I genügt es zu zeigen, dass die Zahlen R die Bedingungen A) und B) erfüllen.

Zu A). Jeder Zug verändert eine der Zahlen $a, b, c, \ldots$, also eine Zeile des Schemas, und damit eine oder mehrere Kolonnen in je einer Ziffer. Demnach ändert sich auch R.

Zu B). Ist $R>0$ und P eine dyadische Zahl, für die $0 \leqq P < R$ gilt, so unterscheiden sich R und P, von links her verglichen, zum ersten Mal an einer Stelle, wo P eine 0, R eine 1 aufweist. In der entsprechenden Kolonne des Schemas steht dann eine ungerade Anzahl von Einsen, also mindestens eine. Irgendeine dieser Einsen, oder die einzige, befinde sich in der Zeile, die zum Haufen x gehört. Durch Verkleinerung von x lässt sich erreichen, dass diese 1 zu 0 wird und überdies die weiter nach rechts liegenden Ziffern der Zeile sich ändern oder nicht, je nachdem P in den entsprechenden Stellen von R abweicht oder nicht. Nach dieser Verkleinerung von x ist P die dem Schema zugeordnete dyadische Zahl.

Hiermit ist Satz III bewiesen.

Von Nim zu Misère-Nim

Die Sprague–Grundy–Theorie untersucht das Normalspiel: Wer nicht mehr ziehen kann, verliert. Beim Misèrespiel gilt umgekehrt: Wer nicht mehr ziehen kann, gewinnt. Das klingt zunächst symmetrisch, manches lässt sich übertragen, doch die Rechnungen verändern sich erheblich.

Ohne Beschränkung der Allgemeinheit darf angenommen werden, dass der Sieg dem Partner zufällt, der die Partie beendet. Man braucht nur alle Endstellungen zu verbieten, in denen der zuletzt Ziehende verliert.
Roland Sprague (1935)

Aufgabe: Wir betrachten Nim $G = (X, A, \sigma, \tau)$, doch nun als Misèrespiel. Beschreiben Sie dies als Normalspiel auf einem geeigneten Teilgraphen.

Lösung: Für Nim haben wir die Zustandsmenge $X = \mathbb{N}^{(\mathbb{N})}$. Aktionen $(r, s) : x \to y$ sind Paare $(r, s) \in \mathbb{N}^2$ mit $1 \le s \le x_r$ und $y = x - s\, e_r$. Der einzige terminale Zustand ist demnach $x = 0$, aktiv ist $x \neq 0$. Wir betrachten demnach den vollen Teilgraph $G' = (X', A', \sigma', \tau')$ auf $X' = X \setminus \{0\}$. Das Misèrespiel auf G ist dann das Normalspiel auf G'.

Von Nim zu Misère-Nim

The game [of Nim in normal play] may be modified by agreeing that the player who takes the last counter from the table loses. [...] The safe combinations are the same as before, except that an odd number of piles, each containing one, is now safe, while an even number of ones is not safe.
Charles Bouton (1901)

Aufgabe: Explizieren und beweisen Sie diese Formel der Gewinnfunktion.

Satz C1L: Boutons Lösung des Nim-Spiels, normal vs misère

Für das Normalspiel von Nim haben wir die vertraute Gewinnfunktion

$$\nu : X \to \{0,1\} : \nu(x) = \begin{cases} 0 & \text{falls } \bigoplus_{i\in\mathbb{N}} x_i = 0, \\ 1 & \text{falls } \bigoplus_{i\in\mathbb{N}} x_i \ge 1. \end{cases}$$

Im Misèrespiel von Nim hingegen gilt die abgewandelte Formel

$$\mu : X \to \{0,1\} : \mu(x) = \begin{cases} 1 - \nu(x) & \text{falls } \max(x) \le 1, \\ \nu(x) & \text{falls } \max(x) \ge 2. \end{cases}$$

Von Nim zu Misère-Nim

Als unterhaltsames *pub game* von Scam School, `youtu.be/mR6mVm4SuRw`, schön einfach, aber auch etwas unpräzise. Klarheit schafft der Beweis!

Beweis: (1) Für den Endzustand $x = 0$ gilt $\mu(x) = 1 = 1 - \nu(x)$. Für $\max(x) \le 1$ besteht x aus n Haufen der Größe 1. Jeder Zug $x \to y$ führt zu $n - 1$ Haufen der Größe 1, also $\mu(x) = 1 - (n \operatorname{rem} 2) = 1 - \nu(x)$.

Kurzum: Im Misèrespiel ist also eine un/gerade Anzahl von Einserhaufen eine Verlust/Gewinnposition, im Normalspiel jedoch genau umgekehrt.

Sei nun $\max(x) \ge 2$. Wir unterscheiden dabei zwei Fälle:

(2) Existiert nur noch ein Haufen $i \in \mathbb{N}$ mit $x_i \ge 2$, so gilt $\nu(x) = 1$ und ebenso $\mu(x) = 1$: Der ziehende Spieler kann eine gerade Anzahl von Einserhaufen übergeben, also eine Misère-Verlustposition gemäß (1).

(3) Angenommen, es existieren mehrere Haufen $i \in \mathbb{N}$ mit $x_i \ge 2$. Jeder Zug $x \to y$ übergibt dann mindestens einen Haufen mit $y_i \ge 2$, wir gelangen also immer wieder zum Fall (3) und schließlich zu (2).

Somit gilt für $\max(x) \ge 2$ immer $\mu(x) = \nu(x)$. QED

Von Nim zu Misère-Nim

🙂 Im Nim-Spiel können wir die Gewinnfunktion $\nu : X \to \{0,1\}$ leicht berechnen dank Boutons Lösung C1E. Die Sprague–Grundy–Funktion $\gamma : X \to \mathbb{N} : \gamma(x) = \bigoplus_{i\in\mathbb{N}} x_i$ ist ebenso leicht dank C1F. Allgemein für Summen von Spielen haben wir den Satz C1K von Sprague–Grundy.

🙂 Das Misèrespiel auf G entspricht dem Normalspiel auf G', in dem alle terminalen Zustände von G gelöscht wurden. Für die Theorie ist das eine elegante Formulierung, doch die konkreten Rechnungen verändern sich dramatisch. Oft erweist sich das Misèrespiel als wesentlich schwieriger.

🙂 In Misère-Nim können wir die Gewinnfunktion $\mu : X \to \{0,1\}$ und entsprechend die normale Gewinnfunktion $\nu' : X' \to \{0,1\}$ erfreulich leicht berechnen durch den Kniff C1L. Das ist ein glücklicher Zufall.

☹ Für die Sprague–Grundy–Funktion $\gamma' : X' \to \mathbb{N}$ von Misère-Nim ist jedoch keine geschlossene Formel oder effiziente Berechnung bekannt: Der Sprague–Grundy–Satz funktioniert wunderbar für Normalspiele, doch für Misèrespiele scheint er keine Entsprechung zu erlauben.

Das Spiel $\mathrm{Nim}_{<k}$ nach E.H. Moore

Der Mathematiker Eliakim Hastings Moore (1862–1932) war ab 1892 Professor an der University of Chicago und baute dort das bald schon legendäre Mathematik-Department auf. Von ihm stammt das folgende Spiel $\mathrm{Nim}_{<k}$ als eine Verallgemeinerung des Nim-Spiels:

Der ziehende Spieler entfernt Steine aus mindestens einem, aber weniger als k Haufen: Zustandsmenge ist $X = \mathbb{N}^{(\mathbb{N})}$ und ein Zug $x \to y$ bedeutet $x \geq y$ mit $0 < \sharp\,\mathrm{supp}(x - y) < k$. (Moores $\mathrm{Nim}_{<2}$ ist also Boutons Nim.)

Wir entwickeln binär $x_i = \sum_j 2^j x_{ij}$ mit $x_{ij} \in \{0, 1\}$ und definieren die **Moore–Funktion** $M(x) = \sum_j k^j (\sum_i x_{ij} \operatorname{rem} k)$. Wir bilden hier also die Spaltensumme modulo k. (Im Falle $k = 2$ ist das Boutons Nim-Summe.)

Damit konnte Moore 1910 sein Spiel $\mathrm{Nim}_{<k}$ elegant und effizient lösen:

Satz C1M: Moores Lösung seines Spiels $\mathrm{Nim}_{<k}$

Genau dann ist x eine Verlustposition in $\mathrm{Nim}_{<k}$, wenn $M(x) = 0$ gilt.

Das Spiel $\mathrm{Nim}_{<k}$ nach E.H. Moore

📖 E.H. Moore: *A generalization of the game called Nim.* Annals of Math. 11 (1910) 93–94. Moore publizierte seine Lösung ohne Beweis, vermutlich hielt er diesen für allzu leichtgewichtig für die Annals of Mathematics.

Das können Sie nun nachholen und Moores Lösung ausarbeiten:

Übung: Beweisen Sie Satz C1M zu $\mathrm{Nim}_{<k}$ nach Vorbild von C1E zu Nim:
(0) Aus $x \in X$ mit $M(x) = 0$ führt jeder Zug $x \to y$ zu $M(y) \geq 1$.
(1) Aus $x \in X$ mit $M(x) \geq 1$ führt ein Zug $x \to y$ zu $M(y) = 0$.

Man könnte für alle $k \geq 2$ vermuten, dass die Sprague–Grundy–Zahl durch M gegeben ist. Im klassischen Falle $k = 2$ gilt dies nach Satz C1F.
(2) Diese Vermutung ist falsch! Finden Sie ein Gegenbeispiel für $k \geq 3$.
Damit erreichen wir unversehens schon offene Forschungsfragen:

Offene Frage: Was ist die Sprague–Grundy–Funktion des Spiels $\mathrm{Nim}_{<k}$? Können wir sie ähnlich effizient berechnen wie im klassischen Fall $k = 2$?

Das Spiel $\mathrm{Nim}_{<k}$ nach E.H. Moore

Wir betrachten weiterhin Moores $\mathrm{Nim}_{<k}$. Soweit ich weiß, ist für seine Sprague–Grundy–Funktion $\gamma : X \to \mathbb{N}$ keine einfache Lösung bekannt.

Wenn mit genau $n = k$ Stapeln gespielt wird, so gibt es eine Lösung von T.A. Jenkyns, J.P. Mayberry: *The skeletion of an impartial game and the Nim-Function of Moore's* Nim_k. Int. J. of Game Theory 9 (1980) 51–63.

Wenn genau k Stapel reduziert werden müssen, so finden Sie eine Lösung in E. Boros *et al.*: *On the Sprague–Grundy function of Exact-*Nim_k. Discrete Applied Mathematics 239 (2018) 1–14. `doi.org/10.1016/j.dam.2017.08.007`

Das Spiel $\mathrm{Nim}_{<k}$ nach E.H. Moore

Lasker–Nim: Berechnung dank Sprague–Grundy

Eine interessante Variation entsteht, wenn man die Spielregel wie folgt festsetzt: Der am Zuge Befindliche darf irgendeinen der Haufen in zwei Haufen zerteilen oder aber, nach seinem freien Ermessen, verkleinern.
Emanuel Lasker: *Brettspiele der Völker.* Scherl Verlag, Berlin 1931.

Aufgabe: Lösen Sie dieses Spiel, **Lasker–Nim**, möglichst effizient.
Berechnen Sie zu jeder Position $x = (x_1, \dots, x_n)$ die Grundy–Zahl $\gamma(x)$.

Lösung: Das Spiel ist eine Summe, dank Sprague–Grundy gilt also

$$\gamma : \mathbb{N}^n \to \mathbb{N} : (x_1, \dots, x_n) \mapsto \gamma(x_1) \oplus \dots \oplus \gamma(x_n).$$

Wir benötigen demnach nur noch $\gamma : \mathbb{N} \to \mathbb{N}$. Laut Spielregel gilt:

$$\gamma(x) = \operatorname{mex}\big[\,\{\gamma(y) \mid 0 \le y < x\} \cup \{\gamma(y) \oplus \gamma(z) \mid x = y + z,\ 1 \le y \le z\}\,\big]$$

Wir berechnen folgende Tabelle (memoisierte Rekursion, bottom-up):

$x =$	0	1	2	3	4	5	6	7	8	9	10	11	12
$\gamma =$	0	1	2	4	3	5	6	8	7	9	10	12	11

Lasker–Nim: Berechnung dank Sprague–Grundy

Übung: Rechnen Sie die obigen Fälle $x = 0, 1, 2, \dots, 12$ sorgfältig nach. Formulieren Sie die allgemeine Regel. Beweisen Sie diese per Induktion.

Übung: Welchen Zeitaufwand $T(x)$ haben beide Methoden für $\gamma(x)$?
(0) Die naive Rekursion, ohne Speicherung von Zwischenergebnissen.
(1) Die memoisierte Rekursion, wie in obiger Tabelle illustriert.
(2) Die allgemeine Regel, die Sie soeben bewiesen haben.
Vergleichen Sie dies mit unserer Diskussion zu einzeiligem Nim.

Übung: Denken Sie sich „zufällig“ einige Spielpositionen $x \in \mathbb{N}^{(\mathbb{N})}$ aus. Ist dies eine Gewinn- oder Verlustposition in Lasker–Nim? Wie nutzen Sie Ihre Vorarbeit? Nennen Sie jeweils alle Gewinnzüge in dieser Position.

Ähnliche Formeln gelten für alle **Take-and-Break-Spiele**: Der ziehende Spieler darf Objekte entfernen und/oder einen Haufen teilen, wozu viele verschiedene Regeln denkbar sind. In *Winning Ways*, Kapitel 4 „Taking and Breaking“ wird für die Familie der sogenannten **Octalen Spiele** eine konzise Notation eingeführt, siehe `en.wikipedia.org/wiki/Octal_game`.

Grundys Spiel: eine ungelöste Vermutung

Das Spiel beginnt mit einem Haufen von $x \in \mathbb{N}$ Objekten. Der ziehende Spieler teilt einen Haufen seiner Wahl in zwei Haufen ungleicher Größe. Wir vereinbaren Normalspiel: Wer nicht mehr ziehen kann, verliert.

Aufgabe: Dieses Spiel heißt **Grundys Spiel**. Lösen Sie es für kleine x:
Berechnen Sie zur Position x die Grundy–Zahl $\gamma(x)$, soweit möglich.

Lösung: Die Sprague–Grundy–Funktion $\gamma : \mathbb{N} \to \mathbb{N}$ ist gegeben durch:

$$\gamma(x) = \operatorname{mex}\big[\,\{\gamma(y) \oplus \gamma(z) \mid x = y + z,\ 1 \le y < z\}\,\big]$$

Wir berechnen folgende Tabelle (memoisierte Rekursion, bottom-up):

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
γ	0	0	0	1	0	2	1	0	2	1	0	2	1	3	2	1	3	2	4	3	0

⚠ Anders als bei Lasker–Nim erkennen wir hier kein einfaches Muster, das unsere Rekursion zu einer effizienten Lösung abkürzen könnte.

Offene Frage: Ist diese Sprague–Grundy–Funktion $\gamma : \mathbb{N} \to \mathbb{N}$ periodisch?

Grundys Spiel: eine ungelöste Vermutung

Elwyn Berlekamp, John Horton Conway und Richard Guy vermuten in ihrem Buch *Winning Ways* (1982), dass γ tatsächlich periodisch ist.
📖 Richard Guy: *Unsolved Problems in Combinatorial Games* (1996), online verfügbar unter `library.msri.org/books/Book29/files/unsolved.pdf`

Sie können selbst weitere Werte berechnen, leichter mit Computerhilfe. So hat Achim Flammenkamp die ersten $2^{38} \approx 2.74 \cdot 10^{11}$ Werte berechnet, zusammengefasst unter `wwwhomes.uni-bielefeld.de/achim/grundy.html`.
Ein periodisches Verhalten konnte allerdings noch niemand entdecken. Die zunächst simple Rekursion erzeugt eine erstaunlich komplexe Folge.

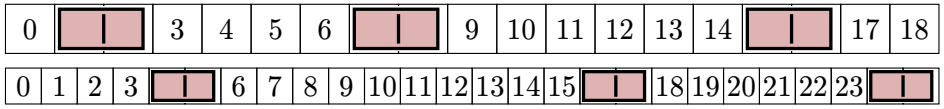
Übung: Denken Sie sich „zufällig“ einige Spielpositionen $x \in \mathbb{N}^{(\mathbb{N})}$ aus. Ist dies eine Gewinn- oder Verlustposition in Grundys Spiel? Wie nutzen Sie hier Ihre Vorarbeit? Nennen Sie alle Gewinnzüge in dieser Position.

Übung: Spielpositionen $x \in \mathbb{N}^{(\mathbb{N})}$ beschreiben wir am besten sortiert, etwa $(x_1 \ge x_2 \ge \dots \ge x_n)$ wie für Partitionen üblich. Schreiben Sie den Spielgraphen für einige kleine Startwerte (x_1) möglichst explizit aus.

Eindimensionales Fliesentetris (Klausur 2022)

Alice und Bob spielen auf einem horizontalen Spielbrett von n Feldern. Sie legen abwechselnd je einen Dominostein, jeder Stein bedeckt zwei benachbarte, noch freie Felder. Wer nicht mehr ziehen kann, verliert.

Für $n = 19$ oder $n = 26$ könnte der Zustand nach drei Zügen so aussehen:



Aufgabe: (1) Berechnen Sie die Grundy–Zahl $\gamma(n)$ für $n \le 10$.
(2) Berechnen Sie $\gamma(n)$ für $n \le 100$ mit einem Python–Skript.
(3) Welche der obigen Zustände sind Gewinnpositionen?
(4) Nennen Sie hierzu alle Gewinnzüge.

In der Klausur haben wir Hilfestellungen gegeben, um die Rechnungen etwas abzukürzen und die knappe Bearbeitungszeit effizient zu nutzen. Sie haben es jetzt viel besser und können den Knobelspaß genießen! Mit Frage (2) erproben Sie Ihre Python–Skills und die Memoisation.

Eindimensionales Fliesentetris (Klausur 2022)

Lösung: (1) Es gilt $\gamma(n) = \operatorname{mex}\{\,\gamma(k) \oplus \gamma(n-2-k) \mid 0 \le k \le n-2\,\}$.
Anfangs finden wir $\gamma(0) = \gamma(1) = \operatorname{mex}\{\,\} = 0$, und dann rekursiv

$$\begin{aligned}
\gamma(2) &= \operatorname{mex}\{\,0\,\} &&= 1,\\
\gamma(3) &= \operatorname{mex}\{\,0,0\,\} &&= 1,\\
\gamma(4) &= \operatorname{mex}\{\,1,0,1\,\} &&= 2,\\
\gamma(5) &= \operatorname{mex}\{\,1,1,1,1\,\} &&= 0,\\
\gamma(6) &= \operatorname{mex}\{\,2,1,0,1,2\,\} &&= 3,\\
\gamma(7) &= \operatorname{mex}\{\,0,2,0,0,2,0\,\} &&= 1,\\
\gamma(8) &= \operatorname{mex}\{\,3,0,3,0,3,0,3\,\} &&= 1,\\
\gamma(9) &= \operatorname{mex}\{\,1,3,1,3,3,1,3,1\,\} &&= 0,\\
\gamma(10) &= \operatorname{mex}\{\,1,1,2,1,0,1,2,1,1\,\} &&= 3.
\end{aligned}$$

$n=$	0	1	2	3	4	5	6	7	8	9	10
$\gamma=$	0	0	1	1	2	0	3	1	1	0	3

Eindimensionales Fliesentetris (Klausur 2022)

(2) Wir implementieren zunächst die Funktion `mex` wie auf Seite C107. Eine erste Berechnung mit Python, hastig und naiv, sieht dann so aus:

```
def gamma(n):
    if n <= 1: return 0;
    return mex({ gamma(k)^gamma(n-2-k) for k in range(0,n-1) })
```

☹ Diese naive Implementierung hat exponentiellen Aufwand. (Warum?) Ab 20 wird es langsam, schon 30 oder 40 dauert eine gefühlte Ewigkeit.

☺ Memoisation reduziert dies auf quadratischen Aufwand. (Warum?) Damit ist selbst die Rechnung bis 1000 blitzschnell. Probieren Sie es!

```
Gamma = { 0: 0, 1: 0 }
for n in range(2, 1001):
    Gamma[n] = mex({ Gamma[k]^Gamma[n-2-k] for k in range(0,n-1) })
```

☺ Die Funktion $\gamma : \mathbb{N} \to \mathbb{N}$ ist periodisch ab $x = 53$ mit Periode 34. Man muss also schon eine Weile rechnen um ein Muster zu erkennen. Dieses Spiel heißt auch *Dawson Kayles*, siehe Siegel: CGT, Seite 185f. Periodizität folgt dort aus dem allgemeinen Satz 2.7 für oktale Spiele.

Eindimensionales Fliesentetris (Klausur 2022)

Nun konkret zu den gezeigten Spielständen:

(3a) Dieser Zustand ist die Summe der Spiele mit 1, 4, 6, 2 freien Feldern. Die Grundy–Zahl ist $0 \oplus 2 \oplus 3 \oplus 1 = 0$. Dies ist eine Verlustposition.

(3b) Dieser Zustand ist die Summe der Spiele mit 4, 10, 6 freien Feldern. Die Grundy–Zahl ist $2 \oplus 3 \oplus 3 = 2 \ne 0$. Dies ist eine Gewinnposition.

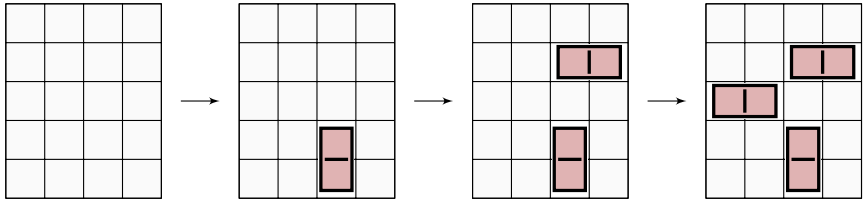
(4) Für den Zustand in (3b) gibt es genau die folgenden neun Gewinnzüge:

$$\{1,2\}, \{6,7\}, \{7,8\}, \{9,10\}, \{11,12\}, \{13,14\}, \{14,15\}, \{19,20\}, \{21,22\}$$

Ausführlich: Wir reduzieren die Grundy–Zahl von $2 \oplus 3 \oplus 3 = 2$ auf 0. Erster Summand von 2 auf 0: nur ein möglicher Zug. Zweiter Summand von 3 auf 1: weitere 6 Züge. Dritter Summand von 3 auf 1: weitere 2 Züge.

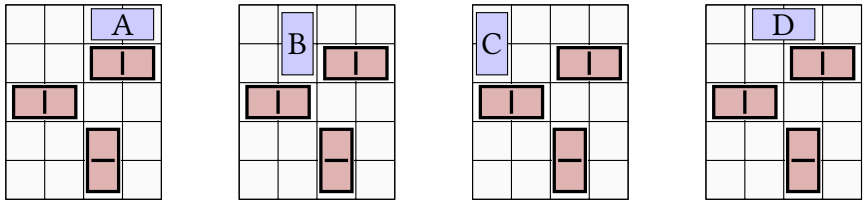
☺ Hier zahlt sich aus, dass wir in (1) die Rechnung dokumentiert haben. Die Grundy–Zahl $\gamma(n) = \operatorname{mex}\{\,\gamma(m) \mid n \to m\,\}$ entscheidet über Gewinn und Verlust. Für die systematische Konstruktion aller Gewinnzüge jedoch benötigen wir darüber hinaus alle Folgezustände $n \to m$ mit einem vorgegebenen Grundy–Wert $\gamma(m)$. Damit gelingt es leicht.

Fliesentetris: Endspiel-Analyse dank Sprague–Grundy



Das Spielfeld besteht aus Quadraten, zum Beispiel rechteckig 4×5.
Beide Spieler ziehen abwechselnd, der Ziehende legt ein Domino auf zwei benachbarte freie Quadrate. Wer nicht mehr ziehen kann, verliert.

Aufgabe: Wie lässt sich hier der Sprague–Grundy–Satz anwenden?
Im oben skizzierten konkreten Beispiel? Allgemein als Algorithmus?
Welcher der folgenden vier Züge A–D führt zum Gewinn?



Fliesentetris: Endspiel-Analyse dank Sprague–Grundy

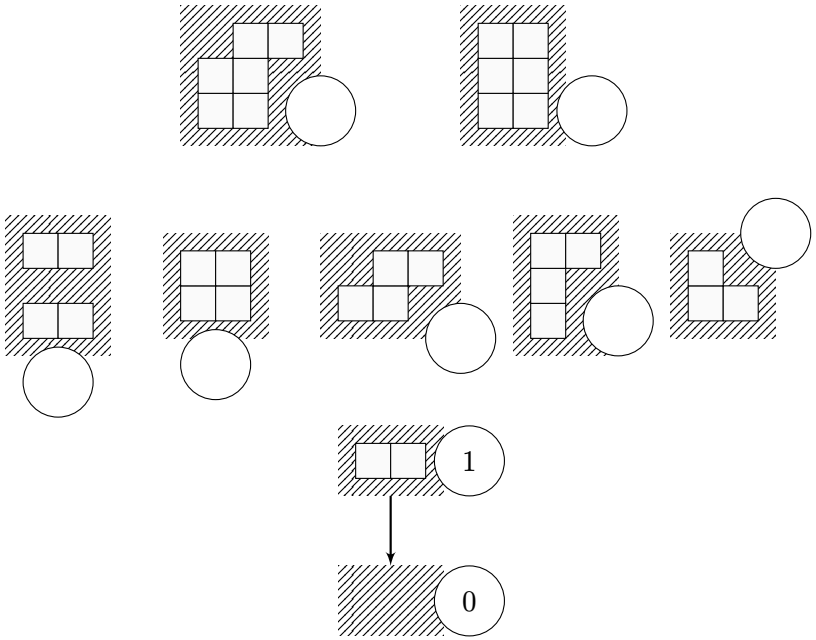
Wir parkettieren hier mit Dominos: Das ist Tetris für Fliesenleger!
Der hier entdeckte Trick gilt ganz allgemein für **Positionsspiele**:
Die Spieler erobern Positionen mit Spielsteinen und behalten diese.
Im Verlauf entstehen Inseln, also Zusammenhangskomponenten,
die sich nicht mehr gegenseitig beeinflussen. Das nutzen wir gerne:
Das Spiel zerfällt nachfolgend in die Summe seiner Komponenten!

Es genügt daher, jede **Komponente** zu analysieren, den Graphen G_i
zu erstellen und seine Sprague–Grundy–Funktion γ_i zu berechnen.
Für das gesamte (End-)Spiel gilt dann $G = \bigoplus_{i \in I} G_i$ und $\gamma = \bigoplus_{i \in I} \gamma_i$.
Die Berechnung von γ gelingt auf diesem Wege wesentlich effizienter.
Anschließend lesen wir aus $x \mapsto \gamma(x)$ alle Gewinnzüge ab. Voilà!

☺ Der Sprague–Grundy–Satz lässt sich überall bei Summen einsetzen.
Diese bilden wir willkürlich, indem wir beliebige Spiele parallel spielen.
Manchmal entstehen Summen auch von ganz alleine, ohne unser Zutun.

☺ Das entspricht übrigens der **externen** und der **internen** Summe,
wie Sie dies von Vektorräumen und ähnlichen Strukturen kennen.

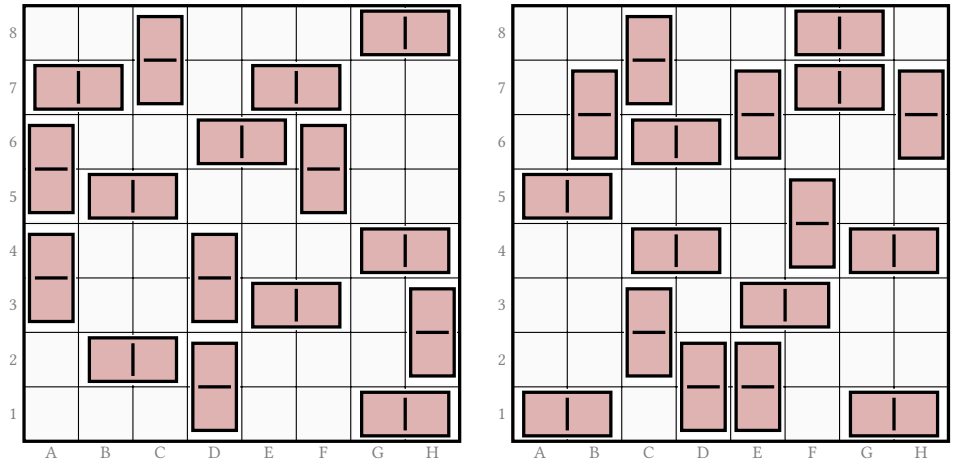
Fliesentetris: Endspiel-Analyse dank Sprague–Grundy



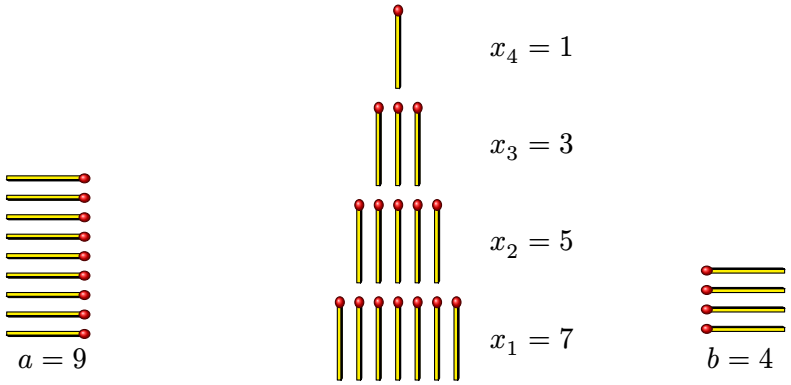
Fliesentetris: Endspiel-Analyse dank Sprague–Grundy

Aufgabe: (1) Bestimmen Sie den Spielgraphen und die Grundy–Zahlen.
Als Hilfestellung zeigt die vorige Folie die kleinsten möglichen Inseln.

(2) Sind die folgenden Spielstände Gewinn- oder Verlustpositionen?
Nennen Sie alle Gewinnzüge! Wie viele gibt es jeweils?



Poker-Nim: Geben ist seliger als Nehmen?



Das Spiel **Poker-Nim** wird gespielt wie Nim mit Spielständen $x \in \mathbb{N}^{(\mathbb{N})}$: Der ziehende Spieler A nimmt $s \geq 1$ Streichhölzer eines Haufens x_r zu seinem Haufen a, oder legt umgekehrt $s \geq 1$ Streichhölzer von a zu x_r. Entsprechend für Spieler B und seinen Haufen b.

Aufgabe: (1) Formalisieren Sie Poker-Nim als ein neutrales Spiel (G, v).
(2) Ist die oben gezeigte Position eine Gewinn- oder Verlustposition?
Allgemein: Wie erkennen Sie Gewinnpositionen und Gewinnzüge?

Poker-Nim: Geben ist seliger als Nehmen?

Bislang betrachteten wir nur Spiele (G, v) auf artinschen Graphen G, also ohne unendliche Wege $x_0 \to x_1 \to x_2 \to \dots$, somit ohne Schleifen. Egal wie gespielt wird, das Spiel endet nach endlich vielen Zügen.

Ausgehend von der terminalen Auszahlung $v : \partial G \to \{0, 1\}$ konstruieren wir per Rückwärtsinduktion C1B die **Gewinnfunktion** $u : X \to \{0, 1\}$:

(T) Für jeden terminalen Zustand $x \in \partial X$ gilt $u(x) = v(x)$.
(A) Für jeden aktiven Zustand $x \in X^\circ$ gilt $u(x) = \sup_{x \to y}[1 - u(y)]$.

Das bedeutet ausführlich als Fallunterscheidung:
(A0) Falls $u(x) = 0$: Für jeden Zug $x \to y$ gilt $u(y) = 1$.
(A1) Falls $u(x) = 1$: Es existiert ein Zug $x \to y$ mit $u(y) = 0$.

Poker-Nim und Northcotts Spiel (siehe C217) sind erste Illustrationen zu **Schleifenspielen** (engl. *loopy games* nach John H. Conway 1978).

A priori ist es hier möglich, unendlich lange zu spielen. Bei optimalem Spiel kann jedoch einer der beiden Spieler seinen Gewinn erzwingen!

☺ Zur Lösung beschränken wir explizit die Zeit bis zum Spielende.

Poker-Nim: Geben ist seliger als Nehmen?

Definition C2A: erweiterte Gewinnfunktion mit Zeitschranke

Sei $G = (X, A, \sigma, \tau)$ ein Graph mit Auszahlung $v : \partial G \to \{0, 1\}$.

Eine **erweiterte Gewinnfunktion** $\tilde{u} = (u, w) : X \to \{0, 1\} \times \mathbb{N}$ besteht aus einer Gewinnfunktion $u : X \to \{0, 1\}$ mit Zeitschranke $w : X \to \mathbb{N}$.

(T) Für jeden terminalen Zustand $x \in \partial X$ gilt $w(x) = 0$ und $u(x) = v(x)$.
(A) Für jeden aktiven Zustand $x \in X^\circ$ gilt $w(x) \geq 1$ und zudem:
(A0) Falls $u(x) = 0$: Für jeden Zug $x \to y$ gilt $u(y) = 1$ und $w(y) \leq w(x)$.
(A1) Falls $u(x) = 1$: Es gibt $x \to y$ mit $u(y) = 0$ und $w(y) < w(x)$.

Beispiel: Mit dieser genial-einfachen Technik lösen wir Poker-Nim: Zustand $(x, a, b) \in X := \mathbb{N}^{(\mathbb{N})} \times \mathbb{N}^2$, Zug $(r, s) : (x, a, b) \to (x', a', b')$ mit $r \in \mathbb{N}$, $s \in \mathbb{Z}$, $1 \leq s \leq x_r$ oder $1 \leq -s \leq a$, $x' = x - s\, e_r$, $a' = b$, $b' = a + s$.

Die Gewinnfunktion ist $u : X \to \{0, 1\} : (x, a, b) \mapsto 1 \wedge \bigoplus_{i \in \mathbb{N}} x_i$, genau wie beim klassischen Nim-Spiel, nun erweitert durch die explizite Zeitschranke $w : X \to \mathbb{N} : (x, a, b) \mapsto [1 - u(x)]a + u(x)b + \sum_{i \in \mathbb{N}} x_i$.

Übung: Dieses Paar (u, w) erfüllt alle Forderungen aus Definition C2A.

Poker-Nim: Geben ist seliger als Nehmen?

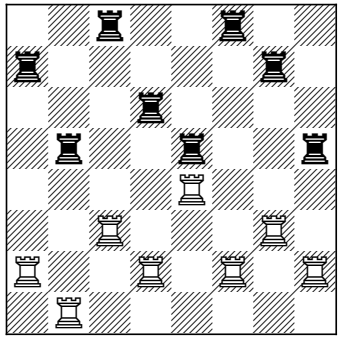
Interpretation: Beim Spielstand $x \in X$ mit $u(x) = 1$ kann der ziehende Spieler seinen Gewinn erzwingen, indem er stets reduzierend zieht (A1); er benötigt dann höchstens $w(x)$ Züge bis zu seinem sicheren Gewinn. (Wir setzen wie immer optimale Spielweise voraus.)

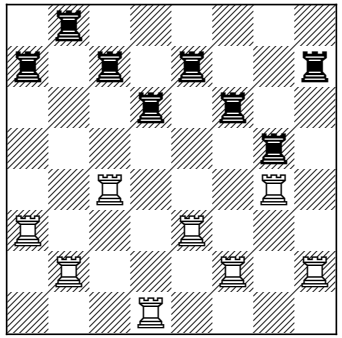
Im Falle $u(x) = 0$ wird der ziehende Spieler verlieren – wie immer bei optimalem Spiel seines Gegners. Der ziehende Spieler kann dies selbst nicht verhindern, sondern höchstens auf $w(x)$ Züge hinauszögern. (Verzögern lohnt sich, wenn der Gegner gelegentlich Fehler macht.)

☺ Erlaubt unser Spiel (G, v) eine erweiterte Gewinnfunktion (u, w), so ist damit das Spiel gelöst: Jeder optimale Spielverlauf ist endlich, wir erkennen die Gewinnpositionen an der Eigenschaft $u(x) = 1$ und die (reduzierenden) Gewinnzüge $x \to y$ an $u(y) = 0$ und $w(y) < w(x)$.

Übung: Wir spielen um Gewinn $+1$ oder Verlust -1, diesmal zusätzlich mit Diskont $\delta \in]0, 1[$, etwa $\delta = 1/2$. Existiert zu jedem Graphen (G, v) eine Gewinnfunktion $\bar{u}$? Wie interpretieren Sie $\bar{u}$? Welche Beziehung besteht zur obigen Gewinnfunktion u mit Zeitschranke w?

Northcotts Spiel: Türme ziehen ohne Ende?





Gezogen wird abwechselnd weiß und schwarz (Markierung am Rand).
Der Ziehende bewegt einen Turm nur vertikal und soweit Platz ist.
Wir vereinbaren Normalspiel: Wer nicht mehr ziehen kann, verliert.

Aufgabe: (1) Formalisieren Sie dies als ein neutrales Spiel (G, v).
Ist der Spielgraph G hier endlich? Wie viele Zustände hat er?
Ist der Spielgraph artinsch? Endet jeder (optimale) Spielverlauf?
(2) Sind die beiden obigen Positionen Gewinn- oder Verlustposition?
Allgemein: Wie erkennen Sie Gewinnpositionen und Gewinnzüge?

Northcotts Spiel: Türme ziehen ohne Ende?

Dieses bekannte Beispiel heißt auch **Northcotts Spiel**.
Es ist insofern ungewöhnlich (und interessant!), als die Regeln allein nicht garantieren, dass jeder Spielverlauf wirklich endet.
Es ist hier durchaus möglich, unendlich lange zu spielen!

Überraschend zeigt sich jedoch, dass bei optimalem Spiel einer der beiden Spieler seinen Gewinn erzwingen kann. Sehen Sie wer und wie?
Dahinter versteckt sich eine weitere Variante des obigen Poker-Nim.
Das ist genau das Ziel der obigen Aufgabe. Probieren Sie es!

Übung: Ist dieses Spiel die Summe seiner Spalten? Falls ja, so lässt sich der Sprague–Grundy–Satz C1K direkt anwenden. Falls nein, so lässt sich das Spiel vielleicht geschickt umformulieren in eine äquivalente Summe. Das vereinfacht die Analyse und ermöglicht eine effiziente Lösung!

Übung: Untersuchen Sie folgende Variante: Über bzw. unter jeder Zeile markiert eine Münze, welcher der beiden Türme als nächstes gezogen wird. Nach jedem Zug in dieser Spalte wird die Münze nach unten bzw. oben umgelegt. Ist dieses variierte Spiel die Summe seiner Spalten?

Northcotts Spiel: Türme ziehen ohne Ende?

Wir lösen Northcotts Spiel durch eine geeignete Zeitschranke:

Definition C2B: erweiterte Grundy–Funktion mit Zeitschranke

Sei $G = (X, A, \sigma, \tau)$ ein Graph. Eine **erweiterte Grundy–Funktion** $\tilde{\gamma} = (\gamma, w) : X \to \mathbb{N} \times \mathbb{N}$ besteht aus einer Grundy–Funktion $\gamma : X \to \mathbb{N}$ und einer Zeitschranke $w : X \to \mathbb{N}$ mit folgenden Eigenschaften:

(T) Für jeden terminalen Zustand $x \in \partial X$ gilt $w(x) = 0$ und $\gamma(x) = 0$.
(A) Für jeden aktiven Zustand $x \in X^\circ$ gilt $w(x) \geq 1$ und zudem:
(A0) Für jeden Zug $x \to y$ gilt $\gamma(y) \neq \gamma(x)$ und $w(y) \leq w(x)$.
(A1) Zu $0 \leq n < \gamma(x)$ existiert $x \to y$ mit $\gamma(y) = n$ und $w(y) < w(x)$.

Interpretation: Bei $\gamma(x) > 0$ gewinnt der ziehende Spieler das Spiel in $\leq w(x)$ Zügen, wenn er immer reduzierend zieht wie in (A1) erklärt.

☺ Erlaubt unser Spiel $(G, 0)$ eine erweiterte Grundy–Funktion (γ, w), so ist damit das Spiel gelöst: Jeder optimale Spielverlauf ist endlich, wir erkennen die Gewinnpositionen an der Eigenschaft $\gamma(x) \geq 1$ und die (reduzierenden) Gewinnzüge $x \to y$ an $\gamma(y) = 0$ und $w(y) < w(x)$.

Northcotts Spiel: Türme ziehen ohne Ende?

Lemma C2C: Eindeutigkeit und Existenz

Sind (u, w) und (u', w') erw. Gewinnfunktionen zu (G, v), so gilt $u = u'$.
Sind (γ, w) und (γ', w') erw. Grundy–Funktionen zu $(G, 0)$, so gilt $\gamma = \gamma'$.

Ist G artinsch, so existiert zu (G, v) eine erw. Gewinnfunktion (u, w).
Ist G artinsch und lokal-endlich, so existiert zum Normalspiel $(G, 0)$ eine erw. Grundy–Funktion (γ, w). Wie zuvor gilt dann $\nu = \gamma \wedge 1$.

☺ Für lösbare Schleifenspiele gilt der Satz von Sprague–Grundy C1K:

Satz C2D: Sprague–Grundy für Schleifenspiele

Gegeben sei eine Familie von Graphen G_i indiziert durch $i \in I$, jeder mit einer erweiterten Grundy–Funktion $(\gamma_i, w_i) : X_i \to \mathbb{N} \times \mathbb{N}$.

Dann erlaubt ihre Summe $G = \bigoplus_{i \in I} G_i$ die erweiterte Grundy–Funktion $(\gamma, w) : X \to \mathbb{N} \times \mathbb{N}$ mit $\gamma(x) = \bigoplus_{i \in I} \gamma_i(x_i)$ und $w(x) = \sum_{i \in I} w_i(x_i)$.

Übung: Rechnen Sie Satz und Lemma sorgfältig nach! Damit lösen Sie Summen von Schleifenspielen. Wie lösen Sie damit Northcotts Spiel?

Das Spiel Chomp! nach David Gale

Beim Spiel Chomp$(m \times n)$ geht es um eine $m \times n$–Tafel Schokolade. Alice und Bob ziehen abwechselnd; die ziehende Spieler:in isst ein Feld $(i, j) \in \{0, \dots, m-1\} \times \{0, \dots, n-1\}$ und alle rechts-oben davon. Das Feld $(0, 0)$ links unten ist jedoch bitter; wer es isst, verliert.

Aufgabe: (1) Welcher der beiden Spieler kann den Gewinn erzwingen? (2) Welche Startzüge sind gewinnend? Finden Sie Gewinnstrategien! Inwiefern können Sie hier die Sprague–Grundy–Theorie anwenden?

Das Spiel Chomp! nach David Gale

Fun fact: „Chomp!“ heißt soviel wie „Mampf!“ oder „Schmatz!“. *Consommer avec modération!* (Mit Mäßigung zu genießen!)

Das Spiel Chomp ähnelt Nim, mit einem entscheidenden Unterschied: Beim Nim-Spiel sind die Zeilen unabhängig und werden parallel gespielt. Bei Chomp hingegen sind Zeilen und Spalten miteinander verbunden, das Spiel ist daher keine Summe von Zeilen / Spalten-Teilspielen!

Eine Anwendung von Summen ist, wie wir wissen, die Endspiel-Analyse: Im Spiel Chomp tritt sie nur auf, wenn zwei getrennte Arme übrig sind.

Die geschickte Zerlegung und überaus effiziente Berechnung des Sprague–Grundy–Satzes steht uns hier sonst nicht zur Verfügung. Ohne dieses Werkzeug müssen wir meist mühsam rekursiv rechnen. Das erklärt die erhöhte Komplexität, die wir im Folgenden spüren.

Natürlich können wir dennoch die Sprague–Grundy–Funktion dieses Spiels berechnen: Wir nutzen sie wie nach wie vor zur externen Summe, etwa wenn wir mehrere Chomp-Spiele (oder andere) parallel spielen. Aber sie nützt uns eben leider nicht für eine interne Summenzerlegung.

Das Spiel Chomp: Gewinnen ist möglich, aber wie?

Satz C2E: Symmetrie und Strategieklau

(0) Das Spiel Chomp$(1 \times n)$ entspricht einzeiligem Nim, kurz Nim$(n-1)$.

(1) Beim quadratischen Spiel Chomp$(n \times n)$ beliebiger Größe $n \in \mathbb{N}_{\geq 2}$ hat Alice eine Gewinnstrategie: Sie zieht $(1, 1)$ und erhält das Nim-Spiel Nim$(n-1) \oplus$ Nim$(n-1)$. Anschließend spiegelt sie jeden Zug von Bob.

(2) Bei jedem rechteckigen Spiel Chomp$(m \times n)$ mit $m, n \in \mathbb{N}_{\geq 2}$ hat Alice eine Gewinnstrategie, jedoch nur implizit, im Allgemeinen unbekannt.

Beweis: Aussagen (0) und (1) sind klar.

(2) Angenommen, Bob hätte eine Gewinnstrategie $s_B : X^\circ \to A$. Das gilt insbesondere nach Alice' erstem Zug $(m-1, n-1) : x_0 \to x_1$, das heißt, Bob hat einen Gewinnzug $s_B(x_1) = (i, j) : x_1 \to x_2$. Dann hat auch Alice den Gewinnzug $(i, j) : x_0 \to x_2$. QED

Das Spiel Chomp: Gewinnen ist möglich, aber wie?

⚠ Quadratisches Chomp ist gemäß dieser Lösung trivial, als Spiel somit langweilig. Schon rechteckiges Chomp jedoch ist notorisch schwierig!

☺ Wir stehen vor einer erstaunlichen Situation: Alice hat nachweislich eine Gewinnstratgie, diese ist im Allgemeinen jedoch (noch) unbekannt!

Der obige Beweis wird *Strategieklau* genannt, siehe Satz C2F. Es ist ein indirekter Beweis, durch Widerspruch, und demnach nicht konstruktiv. Das ist trickreich und elegant, wundersam und lehrreich, aber nicht konstruktiv: Der Beweis sagt uns nur, dass für Alice eine Gewinnstrategie existiert, aber unser Beweis verrät uns nicht, wie diese Strategie aussieht.

☺ *Fun fact:* Strategieklau lässt sich in manchen Brettspielen anwenden, so etwa im Spiel *Hex*. Daraus folgt ein wichtiges Resultat der Topologie: Brouwers Fixpunktsatz! Lesen Sie hierzu den preisgekrönten Artikel

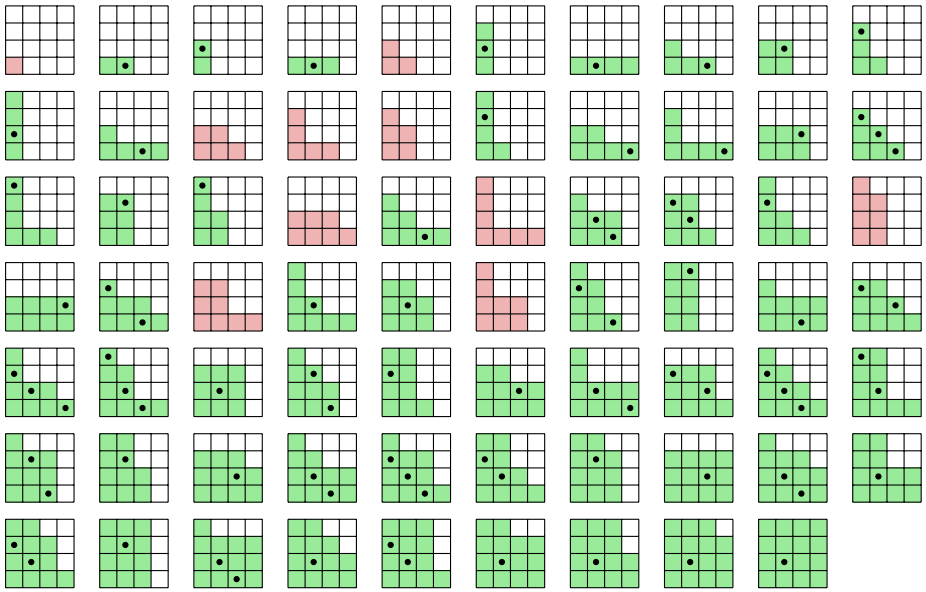
📖 David Gale: *The Game of Hex and the Brouwer Fixed-Point Theorem.* Amer. Math. Monthly 86 (1979) 818–827.

Das Spiel Chomp! nach David Gale

Lösung des Spiels Chomp(4×3); markiert sind alle Gewinnzüge.

Das Spiel Chomp! nach David Gale

Lösung des Spiels Chomp(4×4); markiert sind alle Gewinnzüge.



Sprague–Grundy–Zahlen des Spiels Chomp(4×4)

0	10	0 1	201	1 01	1 0 2	3012	2 310	10 21	0 1 32
2 1 0 3	3 2103	23 012	3 2 023	2 13 02	3 0 1 23	32 4120	2 3 1230	230 412	0 20 120
0 3 2 132	20 13 42	0 2 12 43	3240	4 31 5201	1 2 3 0321	4 201 320	01 20 324	1 0 21 534	4 24 12 03
3240 5124	0 315 4203	45 31 0243	5 4 30 1345	43 203 124	3 4 215 034	3 05 21 430	40 24 12 53	5 3154 6203	04 310 5241
4 0 301 2340	0 41 30 2304	431 203 524	1 40 214 530	34 05 21 635	56 3105 7241	6 5 3012 7340	045 310 6245	5 02 302 1305	02 41 30 7356
5 405 214 630	15 40 216 735	567 3106 8245	7 57 3021 4305	6 021 302 4306	51 02 307 4357	56 407 216 835	5678 3106 9245	8 574 3024 1306	74 57 3074 1357
64 024 307 1358	568 407 216 935	9 5741 3024 8306	81 571 3071 2358	641 024 307 8359	98 5712 3071 4358	812 571 3078 4359	984 5714 3078 2359	9842 5714 3078 6359	

Sprague–Grundy–Zahlen des Spiels Chomp(4×4)

Welche Information beinhalten diese Zahlen? Wozu sind sie gut?

Für jede Position x steht links unten ihre Sprague–Grundy–Zahl $\gamma(x)$. Sie berechnet sich rekursiv wie gewohnt: Für jeden Zug $x \to y$ schauen wir bei der Position y ihren zuvor bereits berechneten Wert $\gamma(y)$ nach. Nach Definition (Satz C1D) finden wir damit $\gamma(x) = \operatorname{mex}\{\gamma(y) \mid x \to y\}$. Dank $\nu = \gamma \wedge 1$ berechnen wir so zugleich auch alle Gewinnpositionen x mit $\gamma(x) \geq 1$, und auch alle möglichen Gewinnzüge $x \to y$ mit $\gamma(y) = 0$.

Mit Geduld und Sorgfalt können wir also alle Werte $\gamma(x)$ berechnen, wie in obiger Tabelle. Leider erkennen wir hier kein einfaches Muster, das mühsame Rekursion zu einer effizienten Lösung abkürzen könnte.

Nach unserem derzeitigen Erkenntnisstand ist demnach Chomp spürbar komplexer als Nim: Für Nim haben wir eine effiziente Lösung, denn die Rechnung lässt sich in polynomieller Zeit durchführen, sogar linear!

Für Chomp hingegen kommen wir um die rekursive Rechnung nicht herum (soweit wir wissen), daher ist Zeitaufwand zur Berechnung von $x \mapsto \gamma(x)$ exponentiell, und so nur für kleine x realistisch durchführbar.

Das allgemeine Spiel Chomp!

Das Spiel $\mathrm{Chomp}(P,\leq)$ wird gespielt auf einer (partiell) geordneten Menge $(P,\leq)$. Gezogen wird abwechselnd. Die ziehende Spieler:in wählt $a \in P$, neues Teilspiel ist $\mathrm{Chomp}(Q,\leq)$ auf der Restmenge

$$Q = P \smallsetminus P_{\geq a} = \{\, b \in P \mid b \not\geq a \,\}.$$

Misèrespiel: Wir fordern ein kleinstes Element $0 \in P$, also $0 \leq P$. Wer dieses Element 0 zieht (als somit letzten Zug), verliert. Das entspricht dem **Normalspiel** auf $P' = P \smallsetminus \{0\}$.

Beispiel: Das Spiel $\mathrm{Chomp}(n)$ mit $n = \{\, 0 \leq 1 \leq 2 \leq \ldots \leq n-1 \,\}$ ist einzeiliges $\mathrm{Nim}(n-1)$. Ebenso für die unendliche Menge $(\mathbb{N},\leq)$.

Bemerkung: Sei $(P,\leq)$ total geordnet. Dann ist das Spiel $\mathrm{Chomp}(P,\leq)$ genau dann artinsch, wenn die Ordnung artinsch ist, das heißt: In $(P,\leq)$ existiert keine unendliche absteigende Kette $a_0 > a_1 > a_2 > \ldots$.

Beispiel: Das Spiel $\mathrm{Chomp}(m \times n)$ mit der Produktordnung entspricht der Schokoladentafel. Ebenso Quader $p \times q \times r$ oder allgemein $(\mathbb{N},\leq)^n$. Im Produkt $(P,\leq) = \prod_{i\in I}(P_i,\leq_i)$ ist die Ordnung $a \leq b$ erklärt durch $a_i \leq b_i$ für jede Koordinate $i \in I$: punktweiser Vergleich von Funktionen.

Wer gewinnt bei Chomp! und wie?

Für jedes rechteckige Spiel $\mathrm{Chomp}(p \times q)$ existiert eine Gewinnstrategie, doch im Allgemeinen ist sie nicht explizit bekannt! Allgemein gilt hierzu:

Satz C2F: Strategieklau im Spiel Chomp

(1) Existiert in $P' = P \smallsetminus \{0\}$ ein Element $m \in P'$, das mit allen $x \in P'$ vergleichbar ist, so hat Bob keine Gewinnstrategie. (2) Ist das Spiel $\mathrm{Chomp}(P,\leq)$ zudem artinsch, so hat Alice eine Gewinnstrategie.

Beweis: (1) Angenommen, Bob hätte eine Gewinnstrategie. Das gilt insbesondere nach Alice' erstem Zug $m : P \to P_1 = P \smallsetminus P_{\geq m}$, das heißt, Bob hat einen Gewinnzug $a : P_1 \to P_2 = P_1 \smallsetminus P_{\geq a}$. Dann hat auch Alice den Gewinnzug $a : P \to P_2$. Das ist ein Widerspruch. QED

Offene Probleme: Für $\mathrm{Chomp}(n \times n \times n)$ ist die Lösung unbekannt.

David Gale bot \$100 für eine vollständige Lösung von 3D-Chomp, also allen Spielen $\mathrm{Chomp}(p \times q \times r)$; das beinhaltet 2D-Chomp.

David Gale bot zudem \$200 für die Antwort zu folgender Frage: Hat der erste Spieler in $\mathrm{Chomp}(\mathbb{N} \times \mathbb{N} \times \mathbb{N})$ eine Gewinnstrategie?

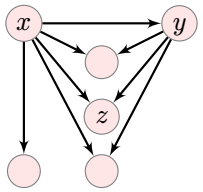
Strategieklau als allgemeine Methode

Diese Beweismethode durch Strategieklau gilt recht allgemein:

Satz C2G: Strategieklau in neutralen Spielen

Sei (G,v) ein neutrales kombinatorisches Spiel auf dem artinschen Graphen $G = (X, A, \sigma, \tau)$ mit terminaler Auszahlung $v : \partial X \to \{0,1\}$.

Vorgelegt sei $x \in X$. Angenommen, es gibt einen Zug $x \to y$ mit $y \in X^\circ$, sodass für jeden Folgezug $y \to z$ auch $x \to z$ gilt. Dann folgt $u(x) = 1$.



Beweis: Zu (G,v) sei $u : X \to \{0,1\}$ die Gewinnfunktion. Angenommen, es gälte $u(x) = 0$. Daraus folgt $u(y) = 1$. Somit existiert mindestens ein Zug $y \to z$ mit $u(z) = 0$. Dank $x \to z$ gilt dann $u(x) = 1$. Widerspruch! QED

Für jeden solchen (in y vollen) Teilgraphen ist x eine Gewinnposition.
Übung: Welche weiteren Teilgraphen mit dieser Eigenschaft finden Sie?
☺ Der Beweis zeigt $u(x) = 1$, denn ein Gewinnzug $x \to ?$ existiert.
☹ Der indirekte Beweis gibt keinen Gewinnzug $x \to ?$ explizit an.

Strategieklau als allgemeine Methode

Übung: (1) Vergleichen Sie die abstrakten Existenzaussagen der beiden Sätze mit explizit-konstruktiven Lösungen. Was nützt wozu?

(2) Warum ist das unendliche Spiel Chomp auf $(\mathbb{N},\leq)^d$ artinsch? Wir versehen $\mathbb{N}^d$ mit der Produktordnung $x \leq y$ gdw $\forall i : x_i \leq y_i$.

(3) Hat in $\mathrm{Chomp}(\mathbb{N}^2)$ der erste Spieler eine Gewinnstrategie? Beweisen Sie dies konstruktiv-explizit oder abstrakt-existentiell?

Hermann Weyl (1885–1955) formulierte die Problematik sehr treffend: Ein Existenzsatz verkündet „das Vorhandensein eines Schatzes, ohne jedoch zu verraten, an welchem Ort. [...] Nicht das Existenztheorem ist das Wertvolle, sondern die im Beweise geführte Konstruktion." (*Über die neue Grundlagenkrise der Mathematik*, 1921)

Die Existenz einer Lösung ist wichtig, doch meist nur ein erster Schritt. Die explizite Lösung ist eine ungleich schwierigere Frage, doch gerade für Anwendungen unabdingbar. Das gilt insbesondere für Spiele!

Wie viel Rationalität benötigen wir?

Wir erinnern an unsere Annahme unbeschränkter Rationalität (A2A):

$\mathscr{R}_0$: Jeder Spieler will sein Ergebnis (Nutzen, Gewinn, ...) maximieren.

$\mathscr{R}_1$: Jeder Spieler versteht zudem alle Spielregeln und Konsequenzen.

$\mathscr{R}_2$: Es gilt die vorige Aussage $\mathscr{R}_1$, und jeder Spieler weiß dies.

Wie viel Rationalität benötigen wir speziell für kombinatorische Spiele?

Angenommen die ziehende Spielerin ist in einer Gewinnposition.
Dann genügt ihr eine optimale Strategie, also eigene Rationalität ($\mathscr{R}_1$):
Damit kann sie gewinnen, unabhängig davon wie ihr Gegner sich verhält.

Angenommen jedoch der ziehende Spieler ist in einer Verlustposition.
Wenn er rational ist ($\mathscr{R}_1$) und dies auch von seiner Gegnerin weiß ($\mathscr{R}_2$),
dann kann er das aussichtslose Spiel aufgeben, denn er wird verlieren.

Wenn er hingegen hofft, dass seine Gegnerin Fehler machen könnte,
dann lohnt sich erwartungsgemäß weiterhin, geschickt zu spielen ($\mathscr{R}_1$).
Sprichwörtlich: *„It ain't over till it's over."* / *„... till the fat lady sings."*
Optimist: „Wunder geschehen." Pessimist: „Die Hoffnung stirbt zuletzt."

Wie viel Rationalität benötigen wir?

Rationalität ist eine zentrale, aber manchmal allzu starke Annahme:
Im Schach kenne ich zwar alle Regeln, aber nicht alle Konsequenzen;
mir fehlt die Rechenkapazität, ausreichend viele Züge vorauszudenken.
Für Menschen wie für Computer ist die Komplexität entscheidend!

Unsere Vorlesungsexperimente zeigen: Fehler treten tatsächlich auf,
selbst in einfachen Spielen wie einzeiligem Nim (vor dessen Lösung).
Teilnehmer verstehen vollständig die Regeln und wollen gewinnen ($\mathscr{R}_0$),
sie wissen auch, wie die Berechnung prinzipiell durchzuführen wäre...

Doch ohne genügend Rechenzeit, ohne Hilfsmittel wie Stift und Papier,
überschreitet der exponentielle Aufwand die verfügbare Kapazität:
Schon bei Spielbeginn in Position $x = 20$ scheint der Weg meist lang und
schwierig genug, um den einen oder anderen Fehler zu provozieren.

Befindet man sich in einer Verlustposition, so sind zwar in der *Theorie*
alle Züge gleich schlecht, aber in der *Praxis* eben nicht! Zum Beispiel
lohnen sich Züge, die dem Gegner die Analyse *erschweren*, damit kann
man ihn vielleicht *überraschen* und *verwirren* und so Fehler *provozieren*.

Wie viel Rationalität benötigen wir?

Nach der Lösung C1A sollte einzeiliges Nim fehlerfrei gespielt werden.
Vernichtet die mathematische Lösung jeglichen Spielspaß? Vielleicht.
Sie gewinnen dafür das intellektuelle Vergnügen, ein Problem zu lösen.
Zudem gibt es noch viele weitere Spiele, der Spielspaß dauert also an!

Dieselben Beobachtungen wiederholen sich noch dramatischer bei Nim.
Anfangs erkennen Sie wenig Struktur. Selbst wenn Sie sehen, dass Ihr
Gegenüber über eine Gewinnstrategie verfügt, so können Sie diese
schwerlich erraten. Alles ändert sich nach Boutons genialem Satz C1E.

Beobachten Sie dabei ihren Lernprozess vom *Whaaa?* bis zum *Aha!*
Diese Interaktion in der Vorlesung kostet enorm viel Zeit und Aufwand.
Viel schneller wäre, die Antworten nacheinander paradieren zu lassen,
selbst wenn Sie sich zu diesem Zeitpunkt noch gar keine Fragen stellen.

Es fordert eine große Investition für alle Beteiligten, aber es lohnt sich:
Sie spüren die Probleme, Sie stellen Fragen, Sie äußern erste Ideen,
dafür will ich Ihnen Zeit lassen und genügend Raum für Diskussion.
Im Nachgang sollen Sie alles gründlich nacharbeiten und festigen.

Wie viel Rationalität benötigen wir?

Sie verstehen jetzt auch praktisch und anschaulich, warum ich Spiele
in der Vorlesung (bzw. begleitend im Casino) tatsächlich spielen will:
Sie sammeln dort Erfahrungen, die die rationale Theorie illustrieren,
oft genug bestätigen, aber manchmal auch darüber hinausgehen!

Alle Informationen liegen offen, Spielregeln und Zielsetzung sind klar.
Um das Verhalten vorherzusagen, benötigen wir weitere Annahmen,
vereinfachend setzen wir hierzu unbegrenzte Rationalität voraus. Doch
Menschen verhalten sich nicht immer rational, aus diversen Gründen.

Das gilt selbst für sehr einfache Spiele, wie unsere ersten Beispiele:
Denken Sie an Ihre eigene Erfahrung und an ihre ersten Versuche!
Die Spannung zwischen (rationaler) Theorie und (oft sehr begrenzt
rationaler) Praxis ist immer wieder überraschend und faszinierend.

Die große Herausforderung der Spieltheorie ist es, diese Kluft zu
verringern, und in günstigen Fällen die Lücke gänzlich zu schließen.
Sie sollten beide Sichtweisen verstehen, beurteilen und nutzen lernen.
Es gelingt sicher nicht immer, aber es lohnt sich danach zu streben.

Zermelos Satz: Schach ist determiniert.

Satz C3A: Zermelo 1913

Schach ist determiniert: Entweder besitzt Weiß eine Gewinnstrategie, oder Schwarz, oder jeder Spieler kann ein Unentschieden erzwingen.

Aufgabe: (0) Formalisieren Sie das Spiel Schach als einen Graphen. (1) Ist dieser Graph artinsch? (2) Beweisen Sie damit Zermelos Satz!

Lösung: Es gibt mehrere (äquivalente) Möglichkeiten, ein gegebenes Spiel wie Schach durch einen Graphen $G = (X, A, \sigma, \tau)$ zu codieren. In weiser Voraussicht notieren wir den gesamten Verlauf der Partie:

Wir konstruieren einen **Spielbaum** ausgehend vom Startzustand 0 durch Aufzeichnen aller bisher gespielten Halb/Züge $x = z_1 z_2 \dots z_n$. Jeder nun legale Halb/Zug z ergibt eine Fortsetzung $y = z_1 z_2 \dots z_n z$. (Was wir einen Zug nennen, heißt beim Schach traditionell Halbzug.)

(0) Die Zustandsmenge X besteht aus allen legalen Halb/Zugfolgen. Jeder Halb/Zug ist eine legale Fortsetzung, formal ausgeschrieben $A = X \setminus \{0\}$ mit $\sigma(z_1 \dots z_n) = z_1 \dots z_{n-1}$ und $\tau(z_1 \dots z_n) = z_1 \dots z_n$.

Zermelos Satz: Schach ist determiniert.

(1) Schach hat die **50-Züge-Regel** (de.wikipedia.org/wiki/50-Züge-Regel): Die Partie endet remis, wenn in den letzten 100 Halb/Zügen weder ein Stein geschlagen noch ein Bauer gezogen wurde.

Genauer: Die Partie endet nach 50 Zügen noch nicht automatisch, sondern das Remis muss von einem der Spieler reklamiert werden. Erst nach 75 Zügen wie in (1) beendet der Schiedsrichter die Partie.

Daraus folgt sofort: Der oben konstruierte Spielgraph G ist artinsch! (Die Stellung allein genügt dazu nicht, die Regeln benötigen den Verlauf. Daher herrscht Notationspflicht bei Turnierpartien.)

(2) Wir codieren Schach als den Spielgraphen $G = (X, A, \sigma, \tau)$. Die terminalen Zustände bewerten wir mit $v : \partial X \to \{-1, 0, +1\}$ als *Verlust, Remis, Gewinn* für den gerade ziehenden Spieler, und $-v$ für den Gegner. Dies ist demnach ein Nullsummenspiel.

Dank C1B existiert hierzu genau eine Gewinnfunktion $u : X \to \{\pm 1, 0\}$. Im Falle $u(0) = \pm 1$ besitzt Weiß / Schwarz eine Gewinnstrategie. Im Falle $u(0) = 0$ kann jeder Spieler ein Unentschieden erzwingen.

Algorithmische Komplexität: Wie schwierig ist Schach?

Zermelos grundlegendes Ergebnis zeigt, dass Schach determiniert ist; es definiert die Funktion u, sagt jedoch nichts über den Wert $u(0)$ aus. Viele haben versucht, diesen Wert zu bestimmen, und erhebliche Mühe investiert. Bis heute ist unbekannt, welcher der drei Fälle tatsächlich gilt.

Im Prinzip können wir die Gewinnfunktion u durch Rückwärtsinduktion berechnen. Der obige Beweis ist *konstruktiv*, das Vorgehen ist *effektiv*, doch die Berechnung nicht ausreichend *effizient*. Dieser Erfolg und die gleichzeitige Enttäuschung hat seit Zermelo viele Menschen fasziniert:

With chess it is possible, in principle, to play a perfect game or construct a machine to do so as follows: One considers in a given position all possible moves, then all moves for the opponent, etc., to the end of the game (in each variation). The end must occur, by the rules of the games after a finite number of moves (remembering the 50 move drawing rule). Each of these variations ends in win, loss or draw. By working backward from the end one can determine whether there is a forced win, the position is a draw or is lost.
Claude Shannon (1916–2001), *Programming a Computer for Playing Chess* (1950)

Algorithmische Komplexität: Wie schwierig ist Schach?

Shannon schätzte die Größe des Spielbaums G grob ab: Ein typisches Schachspiel dauert etwa 80 Halb/Züge, der ziehende Spieler hat etwa 30 mögliche Halb/Züge zur Auswahl, das ergibt grob $\sharp X \approx 30^{80} \approx 10^{120}$. Genaueres finden Sie unter en.wikipedia.org/wiki/Shannon_number.

Das ist eine astronomisch große Zahl, im wahrsten Sinne des Wortes: Die Anzahl aller Atome im beobachtbaren Universum wird auf etwa 10^{80} geschätzt. Shannons Zahl 10^{120} ist noch um 40 Zehnerpotenzen größer. Bei solch großen Zahlen versagt schnell unsere menschliche Intuition.

Die Berechnung von u mit brutaler Gewalt [*brute force*] ist demnach ganz offensichtlich ausgeschlossen. Nichtsdestotrotz ist es denkbar, durch geschickte, tiefsinnige Optimierung eine schnelle Berechnung zu finden und Zermelos Ergebnis C3A schließlich explizit zu berechnen.

Dazu betone ich einen naiv-optimistischen Vergleich: Auch Nim mit 60 Zeilen zu 100 Objekten hat etwa $100^{60} \approx 10^{120}$ Spielzustände. Dennoch können wir Nim effizient lösen, denn es hat Struktur (als Summe), und wir haben Werkzeuge (dank Sprague und Grundy). Mathematik hilft!

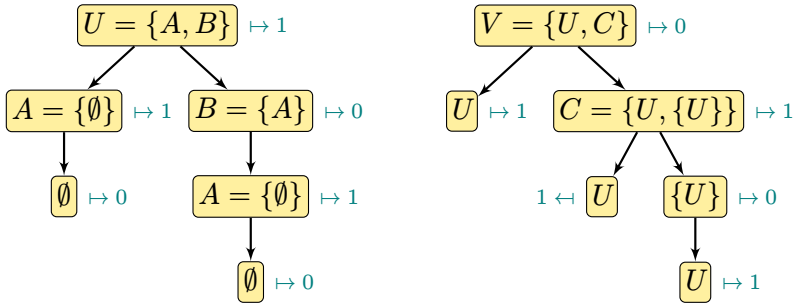
Jede Menge Spiel!

Jede Menge X_0 ist ein kombinatorisches Spiel!

Aufgabe: Alice wählt ein Element $X_1 \in X_0$, dann wählt Bob $X_2 \in X_1$, usw. immer abwechselnd. Wer nicht mehr ziehen kann, verliert.

Bei der Startmenge $U = \{\{\{\}\}, \{\{\{\}\}\}\}$ und fehlerfreiem Spiel gewinnt...?
Bei der Startmenge $V = \{U, \{U, \{U\}\}\}$ und fehlerfreiem Spiel gewinnt...?

Lösung:



Jede Menge Spiel!

☺ Diese verblüffende Interpretation der Mengenlehre und elegante Knobelaufgabe kennen Sie aus dem Mock Exam, hier nun die Lösung.

Lösung: Zunächst in Worten, durch Scharfsinn ohne weitere Technik:

(U) Alice hat genau zwei mögliche Züge. Wenn Alice die Menge $\{\{\}\}$ spielt, dann gibt Bob ihr $\{\}$ zurück, und Alice verliert. Spielt Alice jedoch $\{\{\{\}\}\}$, so muss Bob $\{\{\}\}$ ziehen, Alice spielt dann $\{\}$ und gewinnt.

(V) Auch hier hat Alice zwei mögliche Züge. Wenn Alice die Menge U spielt, so gewinnt Bob, wie eben gesehen. Wenn Alice $\{U, \{U\}\}$ spielt, so wählt Bob geschickt $\{U\}$, Alice muss U ziehen, und Bob gewinnt.

☺ Alternativ mit den Werkzeugen dieses Kapitels (Graphen, Rekursion):

Wir stellen die Menge U bzw. V als Spielgraphen $G = (X, A, \sigma, \tau)$ dar. Damit berechnen wir rekursiv die Gewinnfunktion $u : X \to \{0, 1\}$ durch $u(x) = \sup_{x \to y}[1 - u(y)] = 1 - \inf_{x \to y}[u(y)]$, routiniert und übersichtlich.

Bei der Analyse von V können wir U gewinnbringend wiederverwenden, das ist sowohl graphisch übersichtlicher als auch rechnerisch effizienter.

Jede Menge ist ein kombinatorisches Spiel...

☺ Jede Zermelo–Fraenkel–Menge M ist ein kombinatorisches Spiel!

Beispiele: Die leere Menge $\emptyset$ ist terminal, also eine Verlustposition. Somit ist $\{\emptyset\}$ eine Gewinnposition, ebenso jede Menge M mit $\emptyset \in M$.

Nach John von Neumann definieren wir die natürlichen Zahlen durch $0 = \emptyset$, $1 = \{0\}$, $2 = \{0, 1\}$, allgemein $n + 1 = \{0, 1, \dots, n\}$. Das ist Nim!

Satz C3B: Jede Menge definiert einen Graphen.

Jede Menge M definiert einen artinschen Graphen $G = (X, A, \sigma, \tau)$ mit Wurzel M und den Kanten $x \to y$ genau dann wenn $x \ni y$.

Beweis: Offenkundig ist G demnach ein Graph. Warum ist G artinsch? Hier retten uns die Zermelo–Fraenkel–Axiome der Mengenlehre: Sie verbieten unendlich lange Elementketten $x_0 \ni x_1 \ni x_2 \ni \dots$. QED

☺ Dies ist das *Fundierungsaxiom*, das John von Neumann 1925 in die Mengenlehre einführte. Zeitgleich begründete er die Spieltheorie. Ist die Geschichte der Mathematik nicht wunderbar?

... und umgekehrt!

Satz C3C: Jeder Graph definiert eine Menge.

Sei $G = (X, A, \sigma, \tau)$ ein artinscher Graph. Dank noetherscher Rekursion ordnen wir jedem Zustand $x \in X$ seine Menge $\zeta(x) := \{\zeta(y) \mid x \to y\}$ zu.

Das Graphenspiel G im Zustand x ist äquivalent zum Mengenspiel $\zeta(x)$:
(0) In beiden sind terminale Zustände Verlustpositionen. Dann induktiv:
(1) Jeder Zug $x \to y$ entspricht dem zugehörigen Zug $\zeta(x) \ni \zeta(y)$.
(2) Umgekehrt, zu jedem $z \in \zeta(x)$ existiert $y \leftarrow x$ mit $\zeta(y) = z$.

Übung: Übersetzen Sie die obigen Spiele U und V hin und zurück!

Übung: Betrachten Sie einzeiliges Nim mit einem (kleinen) Startzustand $x \in \mathbb{N}$ (wie auf Seite C101) oder Nim mit $x \in \mathbb{N}^{(\mathbb{N})}$ (wie auf Seite C133) und codieren Sie dies durch die zugehörige Menge $\zeta(x)$.

Übung: Sind die oben erklärten Zuordnungen $\xi : M \mapsto (G, M)$ und $\zeta : (G, x) \mapsto \zeta(x)$ zwischen Mengen und Graphen zueinander invers? Falls keine Bijektionen, in welchem Sinne sind es Äquivalenzen?

Von Mengen und Spielen…

Mit **Mengen** können wir alle mathematischen Objekte formulieren, von den Zahlbereichen $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$ über Funktionen $f : X \to Y$ hin zu weiteren Strukturen wie zum Beispiel Skalarprodukte, Normen, Metriken, Topologien, σ–Algebren, Wahrscheinlichkeits/Maße, usw.

Der Phantasie sind dabei keine Grenzen gesetzt. Erstaunlicherweise lässt sich all dies dank Mengen präzise formulieren und bequem nutzen. Die Mengenlehre dient so der gesamten Mathematik als Fundament, auch nach über einhundert Jahren erfüllt sie treu ihren Zweck.

Es ist daher zunächst überhaupt nicht verwunderlich, dass wir auch Spiele im bewährten Rahmen der Mengenlehre formulieren können. Genau das haben wir mit Graphen und ggf. weiteren Daten erreicht. Die Sprache der Mengen bewährt sich auch hier wieder wunderbar.

Durchaus überraschend ist jedoch, wie nahe sich die Grundstrukturen stehen: Mengen auf der einen Seite, artinsche Graphen auf der anderen. Hier haben wir eine direkte, einfache Übersetzung in beide Richtungen, die alle wesentlichen Aspekte bewahrt, wie oben skizziert.

… zur mathematischen Grundlagenforschung

John H. Conway perfektionierte die enge Verbindung von Spielen und Mengen in seinem berühmten Buch *On Numbers and Games* (1976).

Als sympathisch-spielerische Einführung schrieb Donald E. Knuth 1974 *Surreal Numbers*, ein Dialog zwischen Alice und Bob über die ersten 20 Seiten von *ONAG*. Er richtet sich an Studienanfänger:innen, mit dem Wunsch, ihnen spielerisch mathematische Forschung nahezubringen.

Of course, I wrote this mostly for fun, […] but I must admit that I also had a serious purpose in the back of my mind. Namely, I wanted to provide some material that would help to overcome one of the most serious shortcomings in our present educational system, the lack of training for research work. […] Conway's recent approach to numbers struck me as the perfect vehicle for illustrating the important aspects of mathematical explorations, because it is a rich theory that is almost self-contained, yet with close ties to both algebra and analysis, and because it is still largely unexplored.

Donald E. Knuth, *Surreal Numbers* (1974)

Numberphile, `youtu.be/mPn2AdMH7UQ`

Conways surreale Zahlen

Conways surreale Zahlen ähneln Dedekind–Schnitten: Wir erzeugen jede Zahl $\langle L : R \rangle$ durch zwei Mengen L und R von Zahlen mit $L \not\geq R$, d.h. kein Element $a \in L$ darf größer oder gleich einem Element $b \in R$ sein.

Konstruktion: Seien $L = \{a, a', \dots\}$ und $R = \{b, b', \dots\}$ zwei Mengen surrealer Zahlen mit $L \not\geq R$, das heißt für alle $(a, b) \in L \times R$ gelte $a \not\geq b$.

Dann ist das Paar $x = \langle L : R \rangle \overset{\text{kurz}}{=} \langle a, a', \dots : b, b', \dots \rangle$ eine surreale Zahl. Jede surreale Zahl lässt sich auf diese rekursive Weise beschreiben.

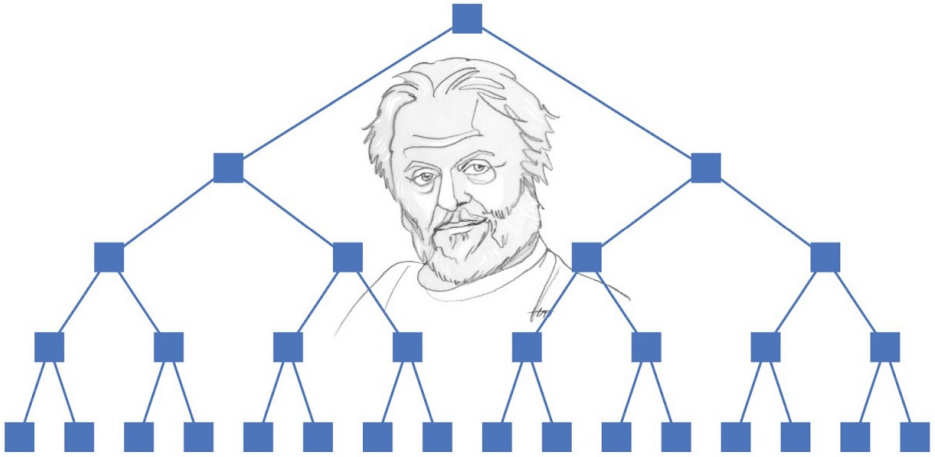
Vergleichsregel: Für surreale Zahlen $x = \langle L_x \mid R_x \rangle$ und $y = \langle L_y \mid R_y \rangle$ definieren wir $x \leq y$ durch die Bedingungen $L_x \not\geq \{y\}$ und $\{x\} \not\geq R_y$.

Diese Relation ist total, reflexiv und transitiv, aber nicht antisymmetrisch. Wir definieren daher die Äquivalenz $x \equiv y$ durch $x \leq y \wedge y \leq x$.

Aufgabe: Konstruieren Sie rekursiv die einfachsten surrealen Zahlen!

Lösung: Die leere Menge $\emptyset$ führt zur surrealen Zahl $0 := \langle : \rangle$ mit $0 \leq 0$. Sodann entstehen $-1 := \langle 0 : \rangle$ und $+1 := \langle : 0 \rangle$ mit $-1 < 0 < +1$. Hingegen ist $\langle 0 : 0 \rangle$ keine surreale Zahl, denn hier gilt $0 \leq 0$.

Conways surreale Zahlen



Prof. Edmund Weitz: *Surreale Zahlen.* `youtu.be/Q9gGVUiMo04`

Knuth schrieb sein Buch als Inspiration zur mathematischen Forschung. Machen Sie sich die Freude und entdecken Sie! Lauschen Sie dem obigen Interview mit Knuth oder dem schönen Video von Prof. Weitz. *Enjoy!*

Die Frucht der Erkenntnis... von wahr und falsch

∀

∃

Die Frucht der Erkenntnis... von wahr und falsch

Dieses Gemälde von Tizian (Tiziano Vecellio, 1488–1576) zeigt den Sündenfall des Menschen. Die Paradieserzählung im Buch Genesis der Bibel spricht vom **Baum der Erkenntnis von Gut und Böse** (Gen 2,9). Ich interpretiere dies freizügig als Erkenntnis von **wahr und falsch**.

Im Folgenden spielen ausnahmsweise mal nicht Alice und Bob, sondern zwecks Diversität und Abwechslung nun Eva und Adam, genauer ∃va und ∀dam, denn uns geht es speziell um die Quantoren. Diese beiden hauchen bewährter mathematischer Logik ihre Seele ein.

Erste Quantorenspiele erlernt jede:r Studierende im ersten Semester der Mathematik. Wir gehen einen mutigen Schritt weiter und wollen diese Quantorenspiele wirklich *spielen*, live und interaktiv im Hörsaal/Casino. Wie immer machen wir uns auf einige Überraschungen gefasst!

Was winkt als Bonus? Das Spiel spornt an. Konkrete Zahlen üben das Rechnen. Der Wunsch nach Systematisierung vertieft das Verständnis. Die Schwierigkeiten werden greifbar: schrittweise Lernen und Üben, formales Protokol, rechnerisch-algorithmische Komplexität, usw.

Quantorenspiele: ∃va gegen ∀dam

Jede Aussage mit Existenz- und Allquantoren ist ein Spiel!

Jede Aussage bekommt ein faires Verfahren, Anklage und Verteidigung:
∃va spielt die Existenzquantoren und möchte die Behauptung erfüllen.
∀dam spielt die Allquantoren und möchte die Behauptung anfechten.

Nulltes Beispiel: Zum Aufwärmen betrachten wir

$$(1) \quad \forall q \in \mathbb{Q}_{>0} \ \exists z \in \mathbb{Z} \ : \ 2^z \geq q$$

$$(2) \quad \forall q \in \mathbb{Q}_{>0} \ \exists z \in \mathbb{Z} \ : \ 2^{z-1} < q \leq 2^z$$

Sei $\mathbb{P} = \{p_0 < p_1 < p_2 < \dots\}$ die Menge der Primzahlen und $p_{-1} = 0$.

$$(3) \quad \forall n \in \mathbb{N} \ \exists p \in \mathbb{P} \ : \ p > n$$

$$(4) \quad \forall n \in \mathbb{N} \ \exists k \in \mathbb{N} \ : \ p_{k-1} \leq n < p_k$$

Jede (vollständig ausformulierte) Gewinnstrategie für ∃va ist ein *Beweis*.
Jede Gewinnstrategie für ∀dam hingegen *widerlegt* die Behauptung.

Quantorenspiele: ∃va gegen ∀dam

Je nach Formulierung des Spiels ist die Rechenlast / Beweislast spürbar unterschiedlich für ∀dam und ∃va... und evtl. sogar für die Spielleitung!

Für (1,2) genügt $z = \lceil \log_2(q) \rceil$. Für (3) genügt $p = \mathrm{lpf}(n!) + 1$, den Wert k für (4) hingegen berechnen wir nur mühsam, selbst mit einem Computer. Zum Spielspaß müssen daher die Regeln präzisiert werden: Wie dürfen / müssen die Daten codiert werden? Dezimal? Als Formel? Als Programm? Wer muss den Wahrheitswert von Aussagen überprüfen / nachweisen ?

Sei $h_n = \sum_{k=1}^{n} 1/k$ die harmonische Reihe, mit $h_0 = 0$ und $h_{-1} = -\infty$.

$$(5) \quad \forall q \in \mathbb{Q} \ \exists n \in \mathbb{N} \ : \ h_n \geq q$$

$$(6) \quad \forall q \in \mathbb{Q} \ \exists n \in \mathbb{N} \ : \ h_{n-1} < q \leq h_n$$

Wir spielen die Summe $x_n = \sum_{k=1}^{n} 1/k^2$ der reziproken Quadratzahlen.

$$(7) \quad \exists q \in \mathbb{Q} \ \forall \varepsilon \in \mathbb{Q}_{>0} \ \exists m \in \mathbb{N} \ \forall n \in \mathbb{N}_{\geq m} \ : \ q - \varepsilon < x_n \leq q$$

$$(8) \quad \forall \varepsilon \in \mathbb{Q}_{>0} \ \exists q \in \mathbb{Q} \ \exists m \in \mathbb{N} \ \forall n \in \mathbb{N}_{\geq m} \ : \ q - \varepsilon < x_n \leq q$$

Wer gewinnt, ∀dam oder ∃va? Wie genau, mit welcher Strategie?

Quantorenspiele: ∃va gegen ∀dam

Jede Aussage mit Existenz- und Allquantoren ist ein Spiel!

Jede Aussage bekommt ein faires Verfahren, Anklage und Verteidigung:
∃va spielt die **E**xistenzquantoren und möchte die Behauptung **e**rfüllen.
∀dam spielt die **A**llquantoren und möchte die Behauptung **a**nfechten.

Erstes Beispiel: Als Spieldaten betrachten wir die Folge

$$(1) \quad x_n := \textstyle\sum_{k=0}^{n} 2^{-k}.$$

Zwei mögliche Spielregeln sind:

$$\text{(K)} \quad \exists a \in \mathbb{Q} \;\; \forall \varepsilon \in \mathbb{Q}_{>0} \;\; \exists m \in \mathbb{N} \;\; \forall n \in \mathbb{N}_{\geq m} \;:\; |x_n - a| < \varepsilon$$

$$\text{(C)} \quad \forall \varepsilon \in \mathbb{Q}_{>0} \;\; \exists a \in \mathbb{Q} \;\; \exists m \in \mathbb{N} \;\; \forall n \in \mathbb{N}_{\geq m} \;:\; |x_n - a| < \varepsilon$$

Weitere Spieldaten, für noch mehr Spielspaß:

$$(2) \quad x_n := \textstyle\sum_{k=0}^{n} (-2)^k$$

$$(3) \quad x_n := \textstyle\sum_{k=0}^{n} 1/k!$$

Quantorenspiele: ∃va gegen ∀dam

Die „Epsilontik“ ist eine geniale Errungenschaft der Analysis. Sie hat im 19. Jh. der intuitiven Vorstellung des infinitesimal Kleinen eine tragfähige Grundlage geschaffen und ist seither überall phantastisch erfolgreich. Es lohnt, dieses Juwel menschlicher Erkenntnis zu verstehen.

Manchmal kann es helfen, Quantorenfolgen als Spiele zu begreifen. (K) ist Konvergenz, (C) ist Cauchy, es gilt (K) ⇒ (C). In (1) hat Eva für beides eine Gewinnstrategie, in (2) Adam. In (3) gewinnt Eva (C), doch Adam gwinnt (K), denn in $\mathbb{Q}$ ist $(x_n)_{n\in\mathbb{N}}$ Cauchy, aber nicht konvergent!

Eine bewährte mathematische Weisheit besagt: Wenn wir eine Aussage beweisen wollen, sollten wir zugleich versuchen, unsere Argumente zu widerlegen. Das erzieht uns, ja zwingt uns, zu (selbst)kritischer Prüfung. Wir verteilen nun die Rollen explizit und erleben manche Überraschung!

Meist spielt leider nur eine Person, in der Vorlesung, beim Rechnen, als Hausaufgabe, in der Klausur, etc. Wir spielen dies im Casino Royal als „Team ∃va“ gegen „Team ∀dam“. Selbst wer theoretisch alles versteht, erlebt interaktiv erfahrungsgemäß neue Aspekte und Aha-Momente.

Quantorenspiele: ∃va gegen ∀dam

Jede Aussage mit Existenz- und Allquantoren ist ein Spiel!

Jede Aussage bekommt ein faires Verfahren, Anklage und Verteidigung:
∃va spielt die **E**xistenzquantoren und möchte die Behauptung **e**rfüllen.
∀dam spielt die **A**llquantoren und möchte die Behauptung **a**nfechten.

Zweites Beispiel: Als Spieldaten betrachten wir die Funktion

$$f : \mathbb{Q} \to \mathbb{Q} : x \mapsto \operatorname{sign}(x^2 - 2) \qquad \text{auf } X = \mathbb{Q}.$$

Zwei mögliche Spielregeln sind:

$$\forall a \in X \;\; \forall \varepsilon \in \mathbb{Q}_{>0} \;\; \exists \delta \in \mathbb{Q}_{>0} \;\; \forall x \in X \;:\; |a - x| < \delta \Rightarrow |f(a) - f(x)| < \varepsilon$$

$$\forall \varepsilon \in \mathbb{Q}_{>0} \;\; \exists \delta \in \mathbb{Q}_{>0} \;\; \forall a \in X \;\; \forall x \in X \;:\; |a - x| < \delta \Rightarrow |f(a) - f(x)| < \varepsilon$$

Weitere Spieldaten, für noch mehr Spielspaß:

$$g : \mathbb{R} \to \mathbb{R} : x \mapsto \operatorname{sign}(x^2 - 2) \qquad \text{auf } X = \mathbb{R},$$

$$h : \mathbb{R} \to \mathbb{R} : x \mapsto \textstyle\sum_{k=1}^{\infty} \operatorname{dist}(x, \frac{1}{k!}\mathbb{Z}) \qquad \text{auf } X = \mathbb{R}.$$

Quantorenspiele: ∃va gegen ∀dam

Was ist nach einer Episode bewiesen? Vermutlich noch nahezu nichts. Nach vielen Episoden wächst die Überzeugung. Gewinnt immer ∃va? Gewinnt immer ∀dam? Können wir eine Gewinnstrategie formulieren? Jede vollständig ausgearbeitete Gewinnstrategie ist ein formaler Beweis!

Durchführung des Spiels ist näherungsweise ein **Null-Wissen-Beweis** [*zero knowledge proof*]: ∃va überzeugt ∀dam, dass sie einen Beweis hat, ohne ihren Beweis selbst preiszugeben; der Nachweis geschieht durch wiederholtes Spiel, bis ∀dam überzeugt ist (oder ∃va aufgibt).

Ich habe im Spiel bewusst auf Stichworte wie „konvergente Folge“ oder „Cauchy–Folge“, „stetige Funktion“ oder „gleichmäßig stetig“ verzichtet. Das ist Teil des Entdeckens, Ausprobierens, Spielens, … Wer sie kennt, genießt gewisse Vorteile, und verspürt doch erneut Anfängerfreuden.

Das obige Spielprinzip interpretiert die Logik als „dialogisches Spiel“. Dazu existiert inzwischen eine umfangreiche mathematische Literatur, speziell in der (infinitären) Logik und der (deskriptiven) Mengenlehre.
`en.wikipedia.org/wiki/Axiom_of_determinacy` `youtu.be/Kj5RCs1FHcc`

Isomorphiespiele: Samson gegen Delila

Isomorphiespiele: Samson gegen Delila

Wir spielen auf zwei totalen Ordnungen, hier ein konkretes Beispiel:

$$X = \{0 < 1 < 2 < 3 < 4 < 5 < 6 < 7\}$$
$$Y = \{0 < 1 < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9\}$$

In Runde $n = 1, 2, 3, \dots$ ziehen die Spieler Elemente $(a_n, b_n) \in X \times Y$:
(0) Verfügbar sind $X_n = X \setminus \{a_1, \dots, a_{n-1}\}$ und $Y_n = Y \setminus \{b_1, \dots, b_{n-1}\}$.
(1) Samson (*spoiler*) wählt ein Element, entweder $a_n \in X_n$ oder $b_n \in Y_n$.
(2) Delila (*duplicator*) wählt komplementär dazu $b_n \in Y_n$ oder $a_n \in X_n$.

Dabei muss stets Monotonie gelten, also $a_i < a_j \Leftrightarrow b_i < b_j$ für alle i, j.
Anschaulich verbinden wir alle Paare (a_i, b_i) ohne Überkreuzungen.

Samson bekommt anfangs 5€ und zahlt 1€ an Delila für jede Antwort.
Samson will einen Unterschied zwischen $(X, \leq)$ und $(Y, \leq)$ aufdecken.
Delila will dies verhindern oder zumindest möglichst lange verzögern.

Im Casino spielen zwei Teams, Links und Rechts. Um die unglückliche Asymmetrie aufzuheben, spielt Team Links auf der linken Tafel Samson und auf der rechten Tafel Delila, Team Rechts entsprechend umgekehrt.

Isomorphiespiele: Samson gegen Delila

Erfolgreiche Strategien ähneln einer binären Suche: Teile und herrsche!
Sie wollen eine nachweislich optimale Strategie? Hier ist die Challenge:

Satz C3D: Isomorphiespiel zwischen endlichen Totalordnungen

Zu $k := |X| < |Y| < \infty$ sei $\nu(k)$ die kleinste Zahl n, für die Samson eine Strategie hat, mit der er spätestens in Runde $n + 1$ gewinnt. Dann gilt

$$\nu(k) = \lfloor \log_2(k + 1) \rfloor.$$

Aufgabe: Wie findet man dieses schöne Ergebnis? Bestimmen Sie $\nu(k)$ für kleine Werte $k = 0, 1, 2, 3, 4, \dots$. Daraus entsteht eine Vermutung!

Lösung: Kleine Werte können Sie leicht austüfteln. Dabei entsteht die folgende Tabelle, dazu eine allgemeine Vermutung und Beweisidee.

$k =$	0	1	2	3	4	5	6	7	…	14	15	…	30	31	…
$\nu(k) =$	0	1	1	2	2	2	2	3	…	3	4	…	4	5	…

Die Werte entstehen aus $\lambda(0) = 0$ rekursiv durch $\lambda(k + 1) = \lambda(\lfloor k/2 \rfloor) + 1$.
Inkremente entstehen nur bei $1, 3, 7, 15, \dots$, also gilt $\lambda(k) = \lfloor \log_2(k + 1) \rfloor$.

Isomorphiespiele: Samson gegen Delila

Beweis: Wir beweisen (1) $\nu(k) \leq \lambda(k)$ und (2) $\nu(k) \geq \lambda(k)$ per Induktion über $k \in \mathbb{N}$. Für $k = 0$ gilt $\nu(0) = 0 = \lambda(0)$. Im Folgenden sei also $k \geq 1$.

(1) Samson halbiert $Y = Y_{<b} \sqcup \{b\} \sqcup Y_{>b}$ mit $|Y_{>b}| - |Y_{<b}| \in \{0, 1\}$. Nach Delilas Zug $X = X_{<a} \sqcup \{a\} \sqcup X_{>a}$ gilt $|X_{<a}| \neq |Y_{<b}|$ oder $|X_{>a}| \neq |Y_{>b}|$. Daraus wählt Samson (X', Y') mit $|X'|$ minimal. Damit erreicht Samson $|X'| \leq \lfloor (k-1)/2 \rfloor$ und spielt weiter auf (X', Y'). Per Induktion folgt:

$$\nu(k) \;\leq\; 1 + \nu(\lfloor (k-1)/2 \rfloor) \;\leq\; 1 + \lambda(\lfloor (k-1)/2 \rfloor) \;=\; \lambda(k)$$

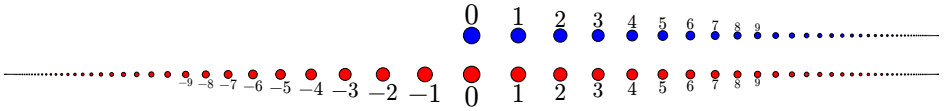
(2) Egal wie Samson zieht, entweder $a \in X$ oder $b \in Y$, Delila kann stets so parieren, dass jedes der beiden Intervallpaare (X', Y') links und rechts $\lfloor (k-1)/2 \rfloor \leq |X'| < |Y'|$ oder $|X'| = |Y'|$ erfüllt. Per Induktion folgt:

$$\nu(k) \;\geq\; 1 + \nu(\lfloor (k-1)/2 \rfloor) \;\geq\; 1 + \lambda(\lfloor (k-1)/2 \rfloor) \;=\; \lambda(k)$$

Das beweist den Satz durch Konstruktion optimaler Strategien. QED

Ganz praktisch können Sie diese Strategien direkt erproben: Spielen!
Sowohl (1) als auch (2) erfordern einige Fallunterscheidungen: Übung!

Isomorphiespiele: Samson gegen Delila



Wir spielen $(\mathbb{N}, \leq)$ gegen $(\mathbb{Z}, \leq)$. Samsons Gewinnstrategie als Formel:

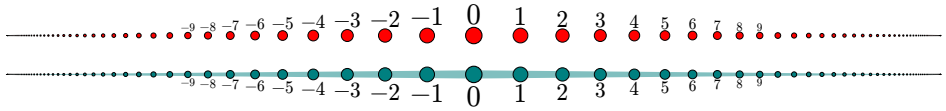
$$\exists a_1 \in \mathbb{N}\ \ \forall a_2 \in \mathbb{N} : a_1 \leq a_2$$
$$\forall b_1 \in \mathbb{Z}\ \ \exists b_2 \in \mathbb{Z}\ : b_1 > b_2$$

Die erste Aussage ist wahr für $(\mathbb{N}, \leq)$. Die zweite Aussage negiert die erste und ist wahr für $(\mathbb{Z}, \leq)$; wir ersetzen $\mathbb{N}$ durch $\mathbb{Z}$ und a_n durch b_n. Diese Formel unterscheidet also beide Strukturen! Das übersetzt Samson in seine Gewinnstrategie, indem er jeweils die Existenzquantoren spielt.

Ausführlich: Samson wählt das kleinste Element $a_1 = 0$ in $(\mathbb{N}, \leq)$. Delila antwortet mit irgendeinem Element b_1 in $(\mathbb{Z}, \leq)$. Samson wählt dazu nun b_2 in $(\mathbb{Z}, \leq)$ mit $b_2 < b_1$. Daraufhin muss Delila aufgeben.

☺ Die Anzahl der Runden entspricht der Anzahl der Variablen.

Isomorphiespiele: Samson gegen Delila



Wir spielen $(\mathbb{Z}, \leq)$ gegen $(\mathbb{Q}, \leq)$. Samsons Gewinnstrategie als Formel:

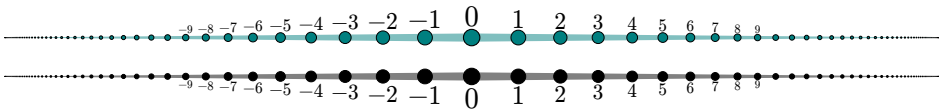
$$\exists a_1 \in \mathbb{Z}\ \ \exists a_2 \in \mathbb{Z}\ \ \forall a_3 \in \mathbb{Z} : [\, a_1 < a_2 \land (a_1 \geq a_3 \lor a_3 \geq a_2) \,]$$
$$\forall b_1 \in \mathbb{Q}\ \ \forall b_2 \in \mathbb{Q}\ \ \exists b_3 \in \mathbb{Q} : [\, b_1 \geq b_2 \lor (b_1 < b_3 \land b_3 < b_2) \,]$$

Die zweite Aussage ist wahr für $(\mathbb{Q}, \leq)$; sie besagt, die Ordnung ist dicht. Die erste Aussage negiert die zweite und ist wahr für $(\mathbb{Z}, \leq)$, da nicht dicht. Diese Formel unterscheidet also beide Strukturen! Das übersetzt Samson in seine Gewinnstrategie, indem er die Existenzquantoren spielt.

Ausführlich: Samson wählt zwei Elemente a_1 und $a_2 = a_1 + 1$ in $(\mathbb{Z}, \leq)$. Delila antwortet mit Elementen b_1 und b_2 in $(\mathbb{Q}, \leq)$ mit $b_1 < b_2$. Samson wählt b_3 in $(\mathbb{Q}, \leq)$ mit $b_1 < b_3 < b_3$. Daraufhin muss Delila aufgeben.

☺ Die Anzahl der Runden entspricht der Anzahl der Variablen.

Isomorphiespiele: Samson gegen Delila



Wir spielen $(\mathbb{Q}, \leq)$ gegen $(\mathbb{R}, \leq)$. Hier kommt das Spiel nie zum Ende!

Die geordneten Mengen $(\mathbb{Q}, \leq)$ und $(\mathbb{R}, \leq)$ sind nicht isomorph, schon die zugrundeliegenden Mengen $\mathbb{Q}$ und $\mathbb{R}$ erlauben keine Bijektion: Die rationalen Zahlen $\mathbb{Q}$ sind abzählbar, die reellen Zahlen $\mathbb{R}$ überabzählbar.

Dennoch führt das Isomorphiespiel (in nur endlicher Zeit) zu keiner Entscheidung: Weder Samson noch Delila kann einen Gewinn erzwingen, da dem jeweils anderen immer noch weitere Zugmöglichkeiten bleiben.

Gleichbedeutend: Diese beiden Ordnungen sind **elementar äquivalent**, jede Aussage der Logik erster Stufe (endlich viele Quantoren je über eine Elementvariable) hat für $(\mathbb{Q}, \leq)$ und $(\mathbb{R}, \leq)$ denselben Wahrheitswert.

☺ Es gibt nicht-isomorphe Modelle, die elementar äquivalent sind!

Isomorphiespiele: Samson gegen Delila

Wir erklären die **n–Äquivalenz** $(X, \leq) \equiv_n (Y, \leq)$ zweier Ordnungen dadurch, dass Delila immer mindestens n Runden überstehen kann.

Dem gegenüber steht der **Quantorenrang** $\mathrm{qr}(\varphi)$ einer Formel $\varphi \in \mathrm{FO}(\leq)$ als die Schachtelungstiefe der Quantoren: Ist φ atomar, also von der Form $(x \leq y)$, so gilt $\mathrm{qr}(\varphi) := 0$. Rekursiv definieren wir $\mathrm{qr}(\neg\varphi) := \mathrm{qr}(\varphi)$ und $\mathrm{qr}(\varphi \lor \psi) = \mathrm{qr}(\varphi \land \psi) := \max\{\mathrm{qr}(\varphi), \mathrm{qr}(\psi)\}$ für alle Junktoren sowie $\mathrm{qr}(\forall x : \varphi) = \mathrm{qr}(\exists x : \varphi) := \mathrm{qr}(\varphi) + 1$ für die Quantoren. Wir schreiben $\mathrm{FO}_n(\leq)$ für die Menge aller Formeln von Quantorenrang höchstens n.

Satz C3E: Korrespondenzsatz von Ehrenfeucht–Fraïssé

Genau dann gilt n–Äquivalenz $(X, \leq) \equiv_n (Y, \leq)$, wenn jede Formel $\varphi \in \mathrm{FO}_n(\leq)$, vom Quantorenrang $\leq n$, auf beiden Modellen denselben Wahrheitswert ergibt, formal geschrieben $((X, \leq) \models \varphi) \Leftrightarrow ((Y, \leq) \models \varphi)$.

Der Beweis gelingt per Induktion über n. Dies wird hier nicht ausgeführt.
📖 N. Immermann: *Descriptive Complexity.* Springer 1999, Thm. 6.10.
L. Libkin: *Elements of Finite Model Theory.* Springer 2012, Thm. 3.9.