Theorembeweiser und ihre Anwendungen

Prof. Dr.-Ing. Gregor Snelting
Dipl.-Inf. Univ. Daniel Wasserrab

Lehrstuhl Programmierparadigmen
IPD Snelting
Universität Karlsruhe (TH)

Teil III

Anwendungen in der Mathematik

Computer in der Mathematik

mathematische Beweise üblicherweise “Papier und Bleistift”
für neuere Aussagen (vgl. Satz von Fermat) Beweise sehr komplex

uns interessiert:

- Können Computer (und speziell Theorembeweiser) helfen?
- Wenn ja, wie? Nur verifizierend oder selbst “beweisend”?
- Sind Mathematiker bereit, Beweise mit Computerhilfe zu akzeptieren?

Computer in der Mathematik

mathematische Beweise üblicherweise “Papier und Bleistift”
für neuere Aussagen (vgl. Satz von Fermat) Beweise sehr komplex

uns interessiert:

- Können Computer (und speziell Theorembeweiser) helfen?
- Wenn ja, wie? Nur verifizierend oder selbst “beweisend”?
- Sind Mathematiker bereit, Beweise mit Computerhilfe zu akzeptieren?

Computer in der Mathematik

mathematische Beweise üblicherweise “Papier und Bleistift”
für neuere Aussagen (vgl. Satz von Fermat) Beweise sehr komplex

uns interessiert:

- Können Computer (und speziell Theorembeweiser) helfen?
- Wenn ja, wie? Nur verifizierend oder selbst “beweisend”?
- Sind Mathematiker bereit, Beweise mit Computerhilfe zu akzeptieren?

Computer in der Mathematik

mathematische Beweise üblicherweise “Papier und Bleistift”
für neuere Aussagen (vgl. Satz von Fermat) Beweise sehr komplex

uns interessiert:

- Können Computer (und speziell Theorembeweiser) helfen?
- Wenn ja, wie? Nur verifizierend oder selbst “beweisend”?
- Sind Mathematiker bereit, Beweise mit Computerhilfe zu akzeptieren?

Computer in der Mathematik

mathematische Beweise üblicherweise “Papier und Bleistift”
für neuere Aussagen (vgl. Satz von Fermat) Beweise sehr komplex

uns interessiert:

- Können Computer (und speziell Theorembeweiser) helfen?
- Wenn ja, wie? Nur verifizierend oder selbst “beweisend”?
- Sind Mathematiker bereit, Beweise mit Computerhilfe zu akzeptieren?

Mathematische “Top 100”

Freek Wiedijk: Liste der “Top 100” der mathematischen Sätze
`http://www.cs.ru.nl/~freek/100/`
z.B.: $\sqrt{2}$ ist irrational, Fundamentalsatz der Algebra,
Satz des Pythagoras, Fermats letzter Satz, Regel von L’Hôpital etc.

insgesamt der Sätze formalisiert (in beliebigem Beweiser)

Mathematische “Top 100”

Freek Wiedijk: Liste der “Top 100” der mathematischen Sätze
`http://www.cs.ru.nl/~freek/100/`
z.B.: $\sqrt{2}$ ist irrational, Fundamentalsatz der Algebra,
Satz des Pythagoras, Fermats letzter Satz, Regel von L’Hôpital etc.

insgesamt 81% der Sätze formalisiert (in beliebigem Beweiser)

Mathematische “Top 100”

Freek Wiedijk: Liste der “Top 100” der mathematischen Sätze
`http://www.cs.ru.nl/~freek/100/`
z.B.: $\sqrt{2}$ ist irrational, Fundamentalsatz der Algebra,
Satz des Pythagoras, Fermats letzter Satz, Regel von L'Hôpital etc.

insgesamt 81% der Sätze formalisiert (in beliebigem Beweiser)

1.	HOL Light	72%
2.	Mizar	45%
3.	ProofPower (HOL)	42%
4.	Isabelle	41%
	Coq	41%
6.	PVS	15%
7.	nqthm/ACL2	12%
8.	NuPRL	8%

Mathematische "Top 100"

Freek Wiedijk: Liste der "Top 100" der mathematischen Sätze
`http://www.cs.ru.nl/~freek/100/`
z.B.: $\sqrt{2}$ ist irrational, Fundamentalsatz der Algebra,
Satz des Pythagoras, Fermats letzter Satz, Regel von L'Hôpital etc.

insgesamt 81% der Sätze formalisiert (in beliebigem Beweiser)

1.	HOL Light	72%
2.	Mizar	45%
3.	ProofPower (HOL)	42%
4.	Isabelle	41%
	Coq	41%
6.	PVS	15%
7.	nqthm/ACL2	12%
8.	NuPRL	8%

Auch Element dieser Liste: 4-Farben-Satz

Der 4-Farben-Satz

Georges Gonthier

A computer-checked proof of the Four Colour Theorem.

Microsoft Research.

`http://research.microsoft.com/en-us/um/people/gonthier/4colproof.pdf`

4-Farben-Satz

Aussage: “Jede planare Karte kann mit nur 4 Farben gefärbt werden”

4-Farben-Satz

Aussage (verbessert): “Die Regionen einer einfachen planaren Karte können mit nur 4 Farben gefärbt werden, so dass zwei benachbarte Regionen verschiedene Farben haben”

4-Farben-Satz

Aussage (verbessert): “Die Regionen einer einfachen planaren Karte können mit nur 4 Farben gefärbt werden, so dass zwei benachbarte Regionen verschiedene Farben haben”

- 1852: Vermutung erstmals geäußert von Francis Guthrie
- viele berühmte Mathematiker versuchen sich daran (z.B. De Morgan, Hamilton, Cayley, Lebesgue etc.)
- 1879: falscher Beweis durch Kempe
- 1976: Beweis durch Appel und Haken (basierend auf 1879er “Beweis”)

4-Farben-Satz

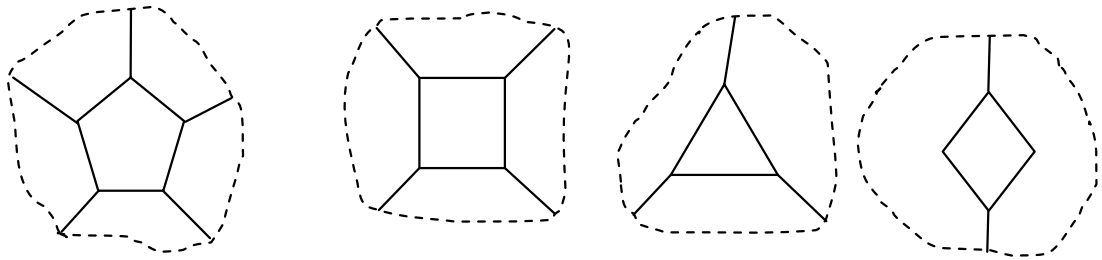
Aussage (verbessert): “Die Regionen einer einfachen planaren Karte können mit nur 4 Farben gefärbt werden, so dass zwei benachbarte Regionen verschiedene Farben haben”

- 1852: Vermutung erstmals geäußert von Francis Guthrie
- viele berühmte Mathematiker versuchen sich daran (z.B. De Morgan, Hamilton, Cayley, Lebesgue etc.)
- 1879: falscher Beweis durch Kempe
- 1976: Beweis durch Appel und Haken (basierend auf 1879er “Beweis”)

Repräsentation: Region als Knoten, “benachbart” als Kante, also Graph

Struktur des Beweises (Kempe)

1. Reduktion auf kubische Karten (an jeder Ecke treffen sich 3 Kanten)
2. in kubischen Karten, Euler-Formel $2E = 6F - 12$
   (allgemein $F - E + N = 2$, F Flächen, E Kanten, N Knoten)
3. suche (kubisches) minimales Gegenbeispiel zu 4-Farben-Satz
4. wg. 2. muss minimales Gegenbeispiel eine dieser Regionen einhalten:



   Induktion: kleinere, durch Entfernung von (einer oder zwei) Kanten entstandene kubische Karte 4-färbbar
5. wenn Fläche nicht Pentagon, Farbe für Fläche trivial
6. wenn Fläche Pentagon, Umfärben der Karte

Korrektur des Beweises

Kempe hat im 6. Schritt ein paar “unangenehme” Fälle vergessen
Fehler erst nach 10 Jahren gefunden, fast 100 Jahre später erst behoben

Korrekter Beweis arbeitet mit Konfigurationen:
zusammenhängende Gruppe von (kompletten) Flächen (*Kern*),
umgeben von *Ring* aus partiellen Flächen

Pentagon durch Konfigurationen ersetzt, so dass

1. jede Landkarte mindestens eine der Konfigurationen besitzt
2. keine Konfiguration in minimalem Gegenbeispiel auftreten kann

Anzahl unvermeidbare Konfigurationen:

Appel und Haken: 1936, später 1476

Korrektur des Beweises

Kempe hat im 6. Schritt ein paar “unangenehme” Fälle vergessen
Fehler erst nach 10 Jahren gefunden, fast 100 Jahre später erst behoben

Korrekter Beweis arbeitet mit Konfigurationen:
zusammenhängende Gruppe von (kompletten) Flächen (*Kern*),
umgeben von *Ring* aus partiellen Flächen

Pentagon durch Konfigurationen ersetzt, so dass

1. jede Landkarte mindestens eine der Konfigurationen besitzt
2. keine Konfiguration in minimalem Gegenbeispiel auftreten kann

Anzahl unvermeidbare Konfigurationen:

Appel und Haken: 1936, später 1476

Korrektur des Beweises

Kempe hat im 6. Schritt ein paar “unangenehme” Fälle vergessen
Fehler erst nach 10 Jahren gefunden, fast 100 Jahre später erst behoben

Korrekter Beweis arbeitet mit Konfigurationen:
zusammenhängende Gruppe von (kompletten) Flächen (*Kern*),
umgeben von *Ring* aus partiellen Flächen

Pentagon durch Konfigurationen ersetzt, so dass

1. jede Landkarte mindestens eine der Konfigurationen besitzt
2. keine Konfiguration in minimalem Gegenbeispiel auftreten kann

Anzahl unvermeidbare Konfigurationen:

Appel und Haken: 1936, später 1476

Korrektur des Beweises

Kempe hat im 6. Schritt ein paar "unangenehme" Fälle vergessen
Fehler erst nach 10 Jahren gefunden, fast 100 Jahre später erst behoben

Korrekter Beweis arbeitet mit Konfigurationen:
zusammenhängende Gruppe von (kompletten) Flächen (*Kern*),
umgeben von *Ring* aus partiellen Flächen

Pentagon durch Konfigurationen ersetzt, so dass

1. jede Landkarte mindestens eine der Konfigurationen besitzt
2. keine Konfiguration in minimalem Gegenbeispiel auftreten kann

Anzahl unvermeidbare Konfigurationen:

Appel und Haken: 1936, später 1476

Korrektur des Beweises

Kempe hat im 6. Schritt ein paar "unangenehme" Fälle vergessen
Fehler erst nach 10 Jahren gefunden, fast 100 Jahre später erst behoben

Korrekter Beweis arbeitet mit Konfigurationen:
zusammenhängende Gruppe von (kompletten) Flächen (*Kern*),
umgeben von *Ring* aus partiellen Flächen

Pentagon durch Konfigurationen ersetzt, so dass

1. jede Landkarte mindestens eine der Konfigurationen besitzt
2. keine Konfiguration in minimalem Gegenbeispiel auftreten kann

Anzahl unvermeidbare Konfigurationen:

Appel und Haken: 1936, später 1476

Robertson et al.: verbessert auf 633 (1995)

...und jetzt kommt der Rechner!

- Appel und Haken benutzen Computer (genauer: IBM 370 Assembler), um alle Konfigurationen durchzurechnen (in $\mathcal{O}(n^4)$)
- Robertson et al. 19 Jahre später C Code (in $\mathcal{O}(n^2)$)

erste Anwendung eines Computers für mathematischen Beweis!

...und jetzt kommt der Rechner!

- Appel und Haken benutzen Computer (genauer: IBM 370 Assembler), um alle Konfigurationen durchzurechnen (in $\mathcal{O}(n^4)$)
- Robertson et al. 19 Jahre später C Code (in $\mathcal{O}(n^2)$)

erste Anwendung eines Computers für mathematischen Beweis!

Berechnung zeigt, dass keine Konfiguration in minimalem Gegenbeispiel also: 4-Farben-Satz bewiesen!

Das (mathematische) Aber

viel Kritik an Appel und Haken für ihren Beweis

"a good mathematical proof is like a poem – this is a telephone directory!"

Hauptpunkte:

- Der 4-Farben-Satz so einfach und elegant formuliert, also warum Beweis nicht auch einfach und elegant?
- keine Aussage, *warum* Aussage richtig, nur kryptischer Computercode
- Programmierung bekannterweise fehlerbehaftet und fern von präziser formaler Mathematik
- einige kleinere Fehler gefunden (konnten aber behoben werden)
- nicht formalisierbar

Das (mathematische) Aber

viel Kritik an Appel und Haken für ihren Beweis

"a good mathematical proof is like a poem – this is a telephone directory!"

Hauptpunkte:

- Der 4-Farben-Satz so einfach und elegant formuliert,
  also warum Beweis nicht auch einfach und elegant?
- keine Aussage, *warum* Aussage richtig, nur kryptischer Computercode
- Programmierung bekannterweise fehlerbehaftet und fern von präziser formaler Mathematik
- einige kleinere Fehler gefunden (konnten aber behoben werden)
- nicht formalisierbar

Das (mathematische) Aber

viel Kritik an Appel und Haken für ihren Beweis

"a good mathematical proof is like a poem – this is a telephone directory!"

Hauptpunkte:

- Der 4-Farben-Satz so einfach und elegant formuliert,
  also warum Beweis nicht auch einfach und elegant?
- keine Aussage, *warum* Aussage richtig, nur kryptischer Computercode
- Programmierung bekannterweise fehlerbehaftet und fern von präziser formaler Mathematik
- einige kleinere Fehler gefunden (konnten aber behoben werden)
- nicht formalisierbar

Das (mathematische) Aber

viel Kritik an Appel und Haken für ihren Beweis

"a good mathematical proof is like a poem – this is a telephone directory!"

Hauptpunkte:

- Der 4-Farben-Satz so einfach und elegant formuliert,
  also warum Beweis nicht auch einfach und elegant?
- keine Aussage, *warum* Aussage richtig, nur kryptischer Computercode
- Programmierung bekannterweise fehlerbehaftet und fern von präziser formaler Mathematik
- einige kleinere Fehler gefunden (konnten aber behoben werden)
- nicht formalisierbar

Das (mathematische) Aber

viel Kritik an Appel und Haken für ihren Beweis

"a good mathematical proof is like a poem – this is a telephone directory!"

Hauptpunkte:

- Der 4-Farben-Satz so einfach und elegant formuliert,
  also warum Beweis nicht auch einfach und elegant?
- keine Aussage, *warum* Aussage richtig, nur kryptischer Computercode
- Programmierung bekannterweise fehlerbehaftet und fern von präziser formaler Mathematik
- einige kleinere Fehler gefunden (konnten aber behoben werden)
- nicht formalisierbar

Das (mathematische) Aber

viel Kritik an Appel und Haken für ihren Beweis

"a good mathematical proof is like a poem – this is a telephone directory!"

Hauptpunkte:

- Der 4-Farben-Satz so einfach und elegant formuliert,
  also warum Beweis nicht auch einfach und elegant?
- keine Aussage, *warum* Aussage richtig, nur kryptischer Computercode
- Programmierung bekannterweise fehlerbehaftet und fern von präziser formaler Mathematik
- einige kleinere Fehler gefunden (konnten aber behoben werden)
- nicht formalisierbar

Das (mathematische) Aber

viel Kritik an Appel und Haken für ihren Beweis

"a good mathematical proof is like a poem – this is a telephone directory!"

Hauptpunkte:

- Der 4-Farben-Satz so einfach und elegant formuliert,
  also warum Beweis nicht auch einfach und elegant?
- keine Aussage, *warum* Aussage richtig, nur kryptischer Computercode
- Programmierung bekannterweise fehlerbehaftet und fern von präziser formaler Mathematik
- einige kleinere Fehler gefunden (konnten aber behoben werden)
- nicht formalisierbar DOCH!

Grundlage und Formalisierung

1995: Robertson, Sanders, Seymour und Thomas: verbesserter Beweis

- ähnliches Argument wie Appel und Haken
- C statt Assembler Code
- Fallunterscheidung deutlich kleiner

2004: Georges Gonthier: Verifizierung des Beweisskripts in Coq
letzte Schwachstellen des Beweises ausgemerzt:

- manuelle Verifikation der kombinatorischen Argumente (Aufzählung der Konfigurationen)
- manuelle Verifikation, dass Computerprogramme Teile dieser Argumente untermauern können

Grundlage und Formalisierung

1995: Robertson, Sanders, Seymour und Thomas: verbesserter Beweis

- ähnliches Argument wie Appel und Haken
- C statt Assembler Code
- Fallunterscheidung deutlich kleiner

2004: Georges Gonthier: Verifizierung des Beweisskripts in Coq
letzte Schwachstellen des Beweises ausgemerzt:

- manuelle Verifikation der kombinatorischen Argumente (Aufzählung der Konfigurationen)
- manuelle Verifikation, dass Computerprogramme Teile dieser Argumente untermauern können

Formalisierung

trotz Begriffen aus Analysis eigentlich Resultat der Kombinatorik
oftmals auch in Graphentheorie ausgedrückt:

> *"Gibt es in einem planaren Graphen eine Knotenfärbung so, dass kein Knoten adjazent zu anderem Knoten mit gleicher Farbe"*

Zwei Ansätze:

- Verwendung des Jordanschen Kurvensatzes
  (formalisiert von T. Hales in HOL Light)
  Nachteile: Beweise schwierig zu formalisieren,
  da Mix aus Kombinatorik und Topologie
- kombinatorische Struktur *Hypermaps*
  Vorteile:
  - Formalisierung eines eigenständigen kombinatorischen Satzes, beweisbar mit intuitionistischer Logik
  - Analysis umgangen durch Diskretisierung, kein Jordanscher Kurvensatz nötig

Formalisierung

trotz Begriffen aus Analysis eigentlich Resultat der Kombinatorik
oftmals auch in Graphentheorie ausgedrückt:

> *"Gibt es in einem planaren Graphen eine Knotenfärbung so, dass kein Knoten adjazent zu anderem Knoten mit gleicher Farbe"*

Zwei Ansätze:

- Verwendung des Jordanschen Kurvensatzes
  (formalisiert von T. Hales in HOL Light)
  Nachteile: Beweise schwierig zu formalisieren,
  da Mix aus Kombinatorik und Topologie
- kombinatorische Struktur *Hypermaps*
  Vorteile:
  - Formalisierung eines eigenständigen kombinatorischen Satzes, beweisbar mit intuitionistischer Logik
  - Analysis umgangen durch Diskretisierung, kein Jordanscher Kurvensatz nötig

Formalisierung

trotz Begriffen aus Analysis eigentlich Resultat der Kombinatorik
oftmals auch in Graphentheorie ausgedrückt:

> *"Gibt es in einem planaren Graphen eine Knotenfärbung so, dass kein Knoten adjazent zu anderem Knoten mit gleicher Farbe"*

Zwei Ansätze:

- Verwendung des Jordanschen Kurvensatzes
  (formalisiert von T. Hales in HOL Light)
  Nachteile: Beweise schwierig zu formalisieren,
  da Mix aus Kombinatorik und Topologie
- kombinatorische Struktur *Hypermaps*
  Vorteile:
  - Formalisierung eines eigenständigen kombinatorischen Satzes, beweisbar mit intuitionistischer Logik
  - Analysis umgangen durch Diskretisierung, kein Jordanscher Kurvensatz nötig

4-Farben-Satz in Coq

Wichtigste Definitionen und Hauptlemma im Reduzierbarkeitsteil:

```
Variable cf : config.
Definition check_reducible : bool := …
Definition cfreducible : Prop := …
Lemma check_reducible_valid : check_reducible -> cfreducible.
```

- `cf` Parameter der Definitionen und Lemmas, steht für Konfigurationen gegeben durch String von Buchstaben und Zahlen
- `check_reducible` ist boolescher Ausdruck, der komplexen kombinatorischen Reduzierbarkeitscheck durchführt
- `cfreducible` ist mathematische Aussage, dass Hypermap konstruiert aus Konfiguration wirklich reduzierbar
- Lemma `check_reducible_valid` garantiert partielle Korrektheit von `check_reducible` bzgl. `cfreducible` für jede Konfiguration `cf`

4-Farben-Satz in Coq

Wichtigste Definitionen und Hauptlemma im Reduzierbarkeitsteil:

```
Variable cf : config.
Definition check_reducible : bool := …
Definition cfreducible : Prop := …
Lemma check_reducible_valid : check_reducible -> cfreducible.
```

- `cf` Parameter der Definitionen und Lemmas, steht für Konfigurationen gegeben durch String von Buchstaben und Zahlen
- `check_reducible` ist boolescher Ausdruck, der komplexen kombinatorischen Reduzierbarkeitscheck durchführt
- `cfreducible` ist mathematische Aussage, dass Hypermap konstruiert aus Konfiguration wirklich reduzierbar
- Lemma `check_reducible_valid` garantiert partielle Korrektheit von `check_reducible` bzgl. `cfreducible` für jede Konfiguration `cf`

4-Farben-Satz in Coq

Wichtigste Definitionen und Hauptlemma im Reduzierbarkeitsteil:

```
Variable cf : config.
Definition check_reducible : bool := …
Definition cfreducible : Prop := …
Lemma check_reducible_valid : check_reducible -> cfreducible.
```

- `cf` Parameter der Definitionen und Lemmas, steht für Konfigurationen gegeben durch String von Buchstaben und Zahlen
- `check_reducible` ist boolescher Ausdruck, der komplexen kombinatorischen Reduzierbarkeitscheck durchführt
- `cfreducible` ist mathematische Aussage, dass Hypermap konstruiert aus Konfiguration wirklich reduzierbar
- Lemma `check_reducible_valid` garantiert partielle Korrektheit von `check_reducible` bzgl. `cfreducible` für jede Konfiguration `cf`

4-Farben-Satz in Coq

Wichtigste Definitionen und Hauptlemma im Reduzierbarkeitsteil:

```
Variable cf : config.
Definition check_reducible : bool := …
Definition cfreducible : Prop := …
Lemma check_reducible_valid : check_reducible -> cfreducible.
```

- `cf` Parameter der Definitionen und Lemmas, steht für Konfigurationen gegeben durch String von Buchstaben und Zahlen
- `check_reducible` ist boolescher Ausdruck, der komplexen kombinatorischen Reduzierbarkeitscheck durchführt
- `cfreducible` ist mathematische Aussage, dass Hypermap konstruiert aus Konfiguration wirklich reduzierbar
- Lemma `check_reducible_valid` garantiert partielle Korrektheit von `check_reducible` bzgl. `cfreducible` für jede Konfiguration `cf`

4-Farben-Satz in Coq

Wichtigste Definitionen und Hauptlemma im Reduzierbarkeitsteil:

```
Variable cf : config.
Definition check_reducible : bool := …
Definition cfreducible : Prop := …
Lemma check_reducible_valid : check_reducible -> cfreducible.
```

- `cf` Parameter der Definitionen und Lemmas, steht für Konfigurationen gegeben durch String von Buchstaben und Zahlen
- `check_reducible` ist boolescher Ausdruck, der komplexen kombinatorischen Reduzierbarkeitscheck durchführt
- `cfreducible` ist mathematische Aussage, dass Hypermap konstruiert aus Konfiguration wirklich reduzierbar
- Lemma `check_reducible_valid` garantiert partielle Korrektheit von `check_reducible` bzgl. `cfreducible` für jede Konfiguration `cf`

4-Farben-Satz in Coq

Damit Beweis von z.B.

```
Lemma cfred232 : (cfreducible (Config 11 33 37
    H 2 H 13 Y 5 H 10 H 1 H 1 Y 3 H 11 Y 4 H
    9 H 1 Y 3 H 9 Y 6 Y 1 Y 1 Y 3 Y 1 Y Y 1 Y)).
```

einfach, da nur 2 logische Schritte:

1. `check_reducible_valid` auf konkrete Konfiguration anwenden
2. trivialer Beweis `true = true`

Parametrisierung ermöglicht also generisches Lemma
aber Beweis für jede Konfiguration braucht ca. 1 Stunde!

Herausforderungen der Formalisierung

- eigene Programme für Testen der Reduzierbarkeit und Unvermeidbarkeit nötig
- Graphentheoriebeweise arbeiten mit visuellen Verständnis des Lesers Theorembeweiser haben diese Möglichkeit nicht
- Begriff der *Hypermap* zwar beschwerlich, ermöglicht aber einige Vereinfachungen
- finden der richtigen "Zwischendefinitionen" (ca. 1000) ermöglicht geringere Zahl an Lemmas (ca. 2500)
- Lemmas: 50% Einzeiler, 75% weniger als 5, 90% weniger als 10 Zeilen 40 länger als Bildschirmseite, größter 700 Zeilen (Korrektheitsbeweis der Erstellung der Konfigurationsreduzierbarkeit)
- Coq braucht 3 Tage, um Beweis zu prüfen, Robertson et al. 3 Stunden (vor 10 Jahren!)

Das Primzahltheorem

J. Avigad, K. Donnely, D. Gray and P. Raff.
A formally verified proof of the prime number theorem.
Transactions on Computational Logic, 9(1):2, ACM, 2007
http://dx.doi.org/10.1145/1297658.1297660

Primzahltheorem

und noch ein Satz aus den "Top 100":

Das Primzahltheorem $\lim_{x\to\infty} \frac{\pi(x)\ln x}{x} = 1$

($\pi(x)$ Anzahl der Primzahlen bis einschl. x)

Was bedeutet das?

Dichte der Primzahlen $\frac{\pi(x)}{x}$ asymptotisch zu $\frac{1}{\ln x}$,
also je größer die betrachteten Zahlen, desto seltener Primzahlen

Vermutung: Gauss, Legendre um 1800

Beweis: unabhängig Hadamard, de la Vallée Poussin 1896

Verbesserung: Selberg und Erdős 1948

Primzahltheorem

und noch ein Satz aus den "Top 100":

$$\text{Das Primzahltheorem } \lim_{x\to\infty} \frac{\pi(x)\ln x}{x} = 1$$

($\pi(x)$ Anzahl der Primzahlen bis einschl. x)

Was bedeutet das?
Dichte der Primzahlen $\frac{\pi(x)}{x}$ asymptotisch zu $\frac{1}{\ln x}$,
also je größer die betrachteten Zahlen, desto seltener Primzahlen

Vermutung: Gauss, Legendre um 1800

Beweis: unabhängig Hadamard, de la Vallée Poussin 1896

Verbesserung: Selberg und Erdős 1948

Primzahltheorem

und noch ein Satz aus den "Top 100":

$$\text{Das Primzahltheorem } \lim_{x\to\infty} \frac{\pi(x)\ln x}{x} = 1$$

($\pi(x)$ Anzahl der Primzahlen bis einschl. x)

Was bedeutet das?
Dichte der Primzahlen $\frac{\pi(x)}{x}$ asymptotisch zu $\frac{1}{\ln x}$,
also je größer die betrachteten Zahlen, desto seltener Primzahlen

Vermutung: Gauss, Legendre um 1800

Beweis: unabhängig Hadamard, de la Vallée Poussin 1896

Verbesserung: Selberg und Erdős 1948

Grundlagen des Beweises von Selberg

man kann schreiben: $\pi(x) = \sum_{p \le x} 1$

Chebyshev definiert

$$\theta(x) = \sum_{p \le x} \ln p \text{ und } \varphi(x) = \sum_{p^a \le x} \ln p = \sum_{n \le x} \Lambda(n)$$

mit $\Lambda(n) = \begin{cases} \ln p & \text{falls } n = p^a \text{ für ein } a \ge 1 \\ 0 & \text{sonst} \end{cases}$

Chebyshev zeigt:

- Primzahltheorem äquivalent zu $\lim_{x \to \infty} \frac{\theta(x)}{x} = 1$ bzw. $\lim_{x \to \infty} \frac{\varphi(x)}{x} = 1$
- gibt Grenzen an: $0.92 < \frac{\pi(x) \ln(x)}{x} < 1.11$

Grundlagen des Beweises von Selberg

man kann schreiben: $\pi(x) = \sum_{p \le x} 1$

Chebyshev definiert

$$\theta(x) = \sum_{p \le x} \ln p \text{ und } \varphi(x) = \sum_{p^a \le x} \ln p = \sum_{n \le x} \Lambda(n)$$

mit $\Lambda(n) = \begin{cases} \ln p & \text{falls } n = p^a \text{ für ein } a \ge 1 \\ 0 & \text{sonst} \end{cases}$

Chebyshev zeigt:

- Primzahltheorem äquivalent zu $\lim_{x \to \infty} \frac{\theta(x)}{x} = 1$ bzw. $\lim_{x \to \infty} \frac{\varphi(x)}{x} = 1$
- gibt Grenzen an: $0.92 < \frac{\pi(x) \ln(x)}{x} < 1.11$

Grundlagen des Beweises von Selberg

man kann schreiben: $\pi(x) = \sum_{p \le x} 1$

Chebyshev definiert

$$\theta(x) = \sum_{p \le x} \ln p \text{ und } \varphi(x) = \sum_{p^a \le x} \ln p = \sum_{n \le x} \Lambda(n)$$

mit $\Lambda(n) = \begin{cases} \ln p & \text{falls } n = p^a \text{ für ein } a \ge 1 \\ 0 & \text{sonst} \end{cases}$

Chebyshev zeigt:

- Primzahltheorem äquivalent zu $\lim_{x \to \infty} \frac{\theta(x)}{x} = 1$ bzw. $\lim_{x \to \infty} \frac{\varphi(x)}{x} = 1$
- gibt Grenzen an: $0.92 < \frac{\pi(x) \ln(x)}{x} < 1.11$

viel Summenzeichen...

Grundlagen des Beweises von Selberg

Primzahltheorem verwendet asymptotisches Verhalten von Funktionen
das kennt jeder Informatiker...

z.B. gilt $\sum_{n \leq x} \frac{1}{n} = \ln x + \mathcal{O}(1)$ oder $\sum_{n \leq x} \ln n = x \ln x - x + \mathcal{O}(\ln x)$

$\frac{\theta(x)}{x} \longrightarrow 1$ und $\frac{\pi(x) \ln x}{x} \longrightarrow 1$ folgen aus $\frac{\varphi(x)}{x} \longrightarrow 1$, also dies zu beweisen

Grundlagen des Beweises von Selberg

Primzahltheorem verwendet asymptotisches Verhalten von Funktionen
das kennt jeder Informatiker... die $\mathcal{O}$-Notation!

z.B. gilt $\sum_{n \leq x} \frac{1}{n} = \ln x + \mathcal{O}(1)$ oder $\sum_{n \leq x} \ln n = x \ln x - x + \mathcal{O}(\ln x)$

$\frac{\theta(x)}{x} \longrightarrow 1$ und $\frac{\pi(x) \ln x}{x} \longrightarrow 1$ folgen aus $\frac{\varphi(x)}{x} \longrightarrow 1$, also dies zu beweisen

Grundlagen des Beweises von Selberg

Primzahltheorem verwendet asymptotisches Verhalten von Funktionen
das kennt jeder Informatiker... die $\mathcal{O}$-Notation!

z.B. gilt $\sum_{n \leq x} \frac{1}{n} = \ln x + \mathcal{O}(1)$ oder $\sum_{n \leq x} \ln n = x \ln x - x + \mathcal{O}(\ln x)$

$\frac{\theta(x)}{x} \longrightarrow 1$ und $\frac{\pi(x) \ln x}{x} \longrightarrow 1$ folgen aus $\frac{\varphi(x)}{x} \longrightarrow 1$, also dies zu beweisen

Grundlagen des Beweises von Selberg

Primzahltheorem verwendet asymptotisches Verhalten von Funktionen
das kennt jeder Informatiker... die $\mathcal{O}$-Notation!

z.B. gilt $\sum_{n \le x} \frac{1}{n} = \ln x + \mathcal{O}(1)$ oder $\sum_{n \le x} \ln n = x \ln x - x + \mathcal{O}(\ln x)$

$\frac{\theta(x)}{x} \to 1$ und $\frac{\pi(x) \ln x}{x} \to 1$ folgen aus $\frac{\varphi(x)}{x} \to 1$, also dies zu beweisen

Grundlagen des Beweises von Selberg

Beweis benötigt zusätzlich noch

- Eulersche μ-Funktion
- Möbius Inversion
- Selbergs *Symmetrieformel*:

$$\sum_{n\le x}\Lambda(n)\ln n+\sum_{n\le x}\sum_{d|n}\Lambda(d)\Lambda(\frac{n}{d})=2x\ln x+\mathcal{O}(x)$$

- Aussage über Grenzen des Fehlerterms $R(x)=\varphi(x)-x$:

$$|R(x)|\ln^2 x\le 2\sum_{n\le x}|R(\frac{x}{n})|\ln n+\mathcal{O}(x\ln x)\text{ bedingt }\frac{R(x)}{x}\to 0$$

Formalisierung

2004: Jeremy Avigad (et al.) Formalisierung in Isabelle:

```
(λx. pi x * ln (real x) / (real x)) ----> 1
```

für den Beweis viele unterstützende Bibliotheken nötig:

- Theorie über natürliche Zahlen und Integer inklusive Primalität und Teilbarkeit und Fundamentalsatz der Arithmetik
- Aussagen über endliche Mengen, deren Summen und Produkte
- Bibliothek für reelle Zahlen, inklusive ln

Was musste formalisiert werden?

Isabelle Theorien guter Startpunkt, aber zusätzliche Bibliotheken nötig zu:

- Eigenschaften μ Funktion, kombinatorische Identitäten und Möbius Inversion
- Bibliothek für $\mathcal{O}$-Notation
- Gleichheiten mit Summen und ln
- Chebyshevs Theoreme

dann blieben als spezifische Komponenten des Beweises zu zeigen:

- Selbergs Symmetrieformel
- zu $R(n)$ gehörige Ungleichheit
- lange Berechnung, um zu zeigen, dass sich $R(n)$ 0 annähert

Was musste formalisiert werden?

Isabelle Theorien guter Startpunkt, aber zusätzliche Bibliotheken nötig zu:

- Eigenschaften μ Funktion, kombinatorische Identitäten und Möbius Inversion
- Bibliothek für $\mathcal{O}$-Notation
- Gleichheiten mit Summen und ln
- Chebyshevs Theoreme

dann blieben als spezifische Komponenten des Beweises zu zeigen:

- Selbergs Symmetrieformel
- zu $R(n)$ gehörige Ungleichheit
- lange Berechnung, um zu zeigen, dass sich $R(n)$ 0 annähert

Herausforderungen der Formalisierung

Asymptotisches Verhalten: $\mathcal{O}$-Notation, $+_o$ und $=_o$:

$a +_o B = \{c \mid \exists b \in B.\ (c = a + b)\}$, $a =_o B$ alternative Syntax zu $\in$

$x^2 + 3x = x^2 + \mathcal{O}(x)$ in Isabelle:

`(λx. x^2 + 3 * x) =o (λx. x^2) +o O(λx. x)`

Casting zwischen Domains: In etc. Funktionen über reelle Zahlen, Primalität, Teilbarkeit Eigenschaften der natürlichen Zahlen also explizites Casten nötig zwischen diesen Domains

Galois-Verbindung: $(n \leq \lfloor x \rfloor) \equiv (real(n) \leq x)$

"Elementare" Beweise: "Trivialität" kann viel Arbeit erfordern oftmals mathematische "Maschinerie" nicht vorhanden, deshalb umständliches Umschreiben nötig

typisches Verhältnis: 1 Seite math. Beweis $\equiv$ Seiten Beweisskript

Extremfall: 3 1/2 Seiten math. Beweis $\equiv$ Seiten Beweisskript

Herausforderungen der Formalisierung

Asymptotisches Verhalten: $\mathcal{O}$-Notation, $+_o$ und $=_o$:
$a +_o B = \{c \mid \exists b \in B.\ (c = a + b)\}$, $a =_o B$ alternative Syntax zu $\in$
$x^2 + 3x = x^2 + \mathcal{O}(x)$ in Isabelle:
`(λx. x^2 + 3 * x) =o (λx. x^2) +o O(λx. x)`

Casting zwischen Domains: In etc. Funktionen über reelle Zahlen,
Primalität, Teilbarkeit Eigenschaften der natürlichen Zahlen
also explizites Casten nötig zwischen diesen Domains
Galois-Verbindung: $(n \leq \lfloor x \rfloor) \equiv (real(n) \leq x)$

"Elementare" Beweise: "Trivialität" kann viel Arbeit erfordern
oftmals mathematische "Maschinerie" nicht vorhanden,
deshalb umständliches Umschreiben nötig
typisches Verhältnis: 1 Seite math. Beweis ≡ Seiten Beweisskript
Extremfall: 3 1/2 Seiten math. Beweis ≡ Seiten Beweisskript

Herausforderungen der Formalisierung

Asymptotisches Verhalten: $\mathcal{O}$-Notation, $+_o$ und $=_o$:
$a +_o B = \{c \mid \exists b \in B.\ (c = a + b)\}$, $a =_o B$ alternative Syntax zu $\in$
$x^2 + 3x = x^2 + \mathcal{O}(x)$ in Isabelle:
`(λx. x^2 + 3 * x) =o (λx. x^2) +o O(λx. x)`

Casting zwischen Domains: In etc. Funktionen über reelle Zahlen,
Primalität, Teilbarkeit Eigenschaften der natürlichen Zahlen
also explizites Casten nötig zwischen diesen Domains
Galois-Verbindung: $(n \leq \lfloor x \rfloor) \equiv (real(n) \leq x)$

"Elementare" Beweise: "Trivialität" kann viel Arbeit erfordern
oftmals mathematische "Maschinerie" nicht vorhanden,
deshalb umständliches Umschreiben nötig
typisches Verhältnis: 1 Seite math. Beweis ≡ Seiten Beweisskript
Extremfall: 3 1/2 Seiten math. Beweis ≡ Seiten Beweisskript

Herausforderungen der Formalisierung

Asymptotisches Verhalten: $\mathcal{O}$-Notation, $+_o$ und $=_o$:
$a +_o B = \{c \mid \exists b \in B.\ (c = a + b)\}$, $a =_o B$ alternative Syntax zu $\in$
$x^2 + 3x = x^2 + \mathcal{O}(x)$ in Isabelle:
`(λx. x^2 + 3 * x) =o (λx. x^2) +o O(λx. x)`

Casting zwischen Domains: In etc. Funktionen über reelle Zahlen,
Primalität, Teilbarkeit Eigenschaften der natürlichen Zahlen
also explizites Casten nötig zwischen diesen Domains
Galois-Verbindung: $(n \leq \lfloor x \rfloor) \equiv (\mathit{real}(n) \leq x)$

"Elementare" Beweise: "Trivialität" kann viel Arbeit erfordern
oftmals mathematische "Maschinerie" nicht vorhanden,
deshalb umständliches Umschreiben nötig
typisches Verhältnis: 1 Seite math. Beweis $\equiv$ 5 Seiten Beweisskript
Extremfall: 3 1/2 Seiten math. Beweis $\equiv$ Seiten Beweisskript

Herausforderungen der Formalisierung

Asymptotisches Verhalten: $\mathcal{O}$-Notation, $+_o$ und $=_o$:
$a +_o B = \{c \mid \exists b \in B.\ (c = a + b)\}$, $a =_o B$ alternative Syntax zu $\in$
$x^2 + 3x = x^2 + \mathcal{O}(x)$ in Isabelle:
`(λx. x^2 + 3 * x) =o (λx. x^2) +o O(λx. x)`

Casting zwischen Domains: In etc. Funktionen über reelle Zahlen, Primalität, Teilbarkeit Eigenschaften der natürlichen Zahlen
also explizites Casten nötig zwischen diesen Domains
Galois-Verbindung: $(n \leq \lfloor x \rfloor) \equiv (real(n) \leq x)$

"Elementare" Beweise: "Trivialität" kann viel Arbeit erfordern
oftmals mathematische "Maschinerie" nicht vorhanden,
deshalb umständliches Umschreiben nötig
typisches Verhältnis: 1 Seite math. Beweis $\equiv$ 5 Seiten Beweisskript
Extremfall: 3 1/2 Seiten math. Beweis $\equiv$ 37 Seiten Beweisskript

Das Robbins-Problem

William McCune.
Solution of the Robbins Problem.
Journal of Automated Reasoning, 19(3):263–276, Springer, 1997.
http://dx.doi.org/10.1145/1146809.1146811

Können Computer selbst “kreativ” sein?

- Im Allgemeinen wird (fast) jeder Frage mit nein beantworten
- Finden von mathematischen Beweisen erfordert auch Kreativität
- Ist ein Computer (Theorembeweiser), der selbst einen neuen Beweis findet, also kreativ?

Im Folgenden Beispiel für einen von Computer entdeckten Beweis

Können Computer selbst “kreativ” sein?

- Im Allgemeinen wird (fast) jeder Frage mit nein beantworten
- Finden von mathematischen Beweisen erfordert auch Kreativität
- Ist ein Computer (Theorembeweiser), der selbst einen neuen Beweis findet, also kreativ?

Im Folgenden Beispiel für einen von Computer entdeckten Beweis

Können Computer selbst “kreativ” sein?

- Im Allgemeinen wird (fast) jeder Frage mit nein beantworten
- Finden von mathematischen Beweisen erfordert auch Kreativität
- Ist ein Computer (Theorembeweiser), der selbst einen neuen Beweis findet, also kreativ?

Im Folgenden Beispiel für einen von Computer entdeckten Beweis

Können Computer selbst “kreativ” sein?

- Im Allgemeinen wird (fast) jeder Frage mit nein beantworten
- Finden von mathematischen Beweisen erfordert auch Kreativität
- Ist ein Computer (Theorembeweiser), der selbst einen neuen Beweis findet, also kreativ?

Im Folgenden Beispiel für einen von Computer entdeckten Beweis

Können Computer selbst “kreativ” sein?

- Im Allgemeinen wird (fast) jeder Frage mit nein beantworten
- Finden von mathematischen Beweisen erfordert auch Kreativität
- Ist ein Computer (Theorembeweiser), der selbst einen neuen Beweis findet, also kreativ?

Im Folgenden Beispiel für einen von Computer entdeckten Beweis

Das Robbins-Problem

1933 präsentiert E. V. Huntington folgende 3 Gleichungen als Basis für boole'sche Algebra (n Komplement):

$$x + y = y + x \qquad \text{Kommutativität}$$
$$(x + y) + z = x + (y + z) \qquad \text{Assoziativität}$$
$$n(n(x) + y) + n(n(x) + n(y)) = x \qquad \text{Huntington-Gleichung}$$

kurz darauf: Robbins formuliert Frage, ob Huntington-Gleichung durch

$$n(n(x + y) + n(x + n(y))) = x \qquad \text{Robbins-Gleichung}$$

ersetzt werden kann (um ein n kürzer)

Robbins-Gleichung gilt in jeder boole'schen Algebra, also Frage:

Sind alle Robbins Algebren boole'sch?

Das Robbins-Problem

1933 präsentiert E. V. Huntington folgende 3 Gleichungen als Basis für boole'sche Algebra (n Komplement):

$$x + y = y + x \qquad \text{Kommutativität}$$
$$(x + y) + z = x + (y + z) \qquad \text{Assoziativität}$$
$$n(n(x) + y) + n(n(x) + n(y)) = x \qquad \text{Huntington-Gleichung}$$

kurz darauf: Robbins formuliert Frage, ob Huntington-Gleichung durch

$$n(n(x + y) + n(x + n(y))) = x \qquad \text{Robbins-Gleichung}$$

ersetzt werden kann (um ein n kürzer)

Robbins-Gleichung gilt in jeder boole'schen Algebra, also Frage:

Sind alle Robbins Algebren boole'sch?

Das Robbins-Problem

1933 präsentiert E. V. Huntington folgende 3 Gleichungen als Basis für boole'sche Algebra (n Komplement):

$$x + y = y + x \qquad \text{Kommutativität}$$
$$(x + y) + z = x + (y + z) \qquad \text{Assoziativität}$$
$$n(n(x) + y) + n(n(x) + n(y)) = x \qquad \text{Huntington-Gleichung}$$

kurz darauf: Robbins formuliert Frage, ob Huntington-Gleichung durch

$$n(n(x + y) + n(x + n(y))) = x \qquad \text{Robbins-Gleichung}$$

ersetzt werden kann (um ein n kürzer)

Robbins-Gleichung gilt in jeder boole'schen Algebra, also Frage:

Sind alle Robbins Algebren boole'sch?

Das Robbins-Problem

Robbins, Huntington, Tarski versuchten sich an diesem Problem
Steve Winker zeigt 1979, dass gilt:

$$\textit{Robbins-Algebra, die } \exists c\ d.(c + d = c) \textit{ erfüllt, ist boole'sch}$$

d.h. Reduktion der Frage *"Sind alle Robbins Algebren boole'sch?"* auf *"Erfüllen alle Robbins-Algebren* $\exists c\ d.(c + d = c)$*?"*

Der Theorembeweiser EQP

EQP automatischer Theorembeweiser
in vielen Bereichen ähnlich Otter, jedoch

- beschränkt auf Gleichungslogik
- Suchstrategie von Otter gesteuert durch vorgegebenen Algorithmus
- Fähigkeit zu *assoziativer-kommutativer* (AC) *Unifikation*

Eigenschaften von EQP

AC Unifikation: Assoziativität und Kommutativität
eines binären Operators (hier: $+$) eingebaut in Inferenzprozess
Heuristik: *super-0 Strategy*, eliminiert kompliziertere Unifikatoren,
kann aber Beweis verlängern (auch Beweise verhindern?)

Demodulation : Simplifikation für abgeleitete Gleichungen
Term t_1 "größer" als t_2, falls $\mathit{length}(t_1) > \mathit{length}(t_2)$
Gleichungen so orientiert, dass linke Seite "größer"

Eigenschaften von EQP

AC Unifikation: Assoziativität und Kommutativität
eines binären Operators (hier: $+$) eingebaut in Inferenzprozess
Heuristik: *super-0 Strategy*, eliminiert kompliziertere Unifikatoren,
kann aber Beweis verlängern (auch Beweise verhindern?)

Demodulation : Simplifikation für abgeleitete Gleichungen
Term t_1 "größer" als t_2, falls $\mathit{length}(t_1) > \mathit{length}(t_2)$
Gleichungen so orientiert, dass linke Seite "größer"

Eigenschaften von EQP

Paramodulation: Inferenzregel für Gleichheit

Paramodulation nur auf linker Seite von orientierten Gleichungen

Heuristik: optionale 'basic' Restriktion, ordnet Ableitungen an, kann u.U. auch Suche erschweren

Suchalgorithmus : gibt jeder Gleichung Gewicht und Alter

Gewicht einer Gleichung Summe der Gewichte seiner Komponenten (analog Alter)

In jeder Iteration der Suchschleife entweder leichteste noch nicht betrachtete oder älteste noch nicht betrachtete Gleichung gewählt

kann durch Angabe eines Verhältnis spezifiziert werden

Eigenschaften von EQP

Paramodulation: Inferenzregel für Gleichheit

Paramodulation nur auf linker Seite von orientierten Gleichungen

Heuristik: optionale 'basic' Restriktion, ordnet Ableitungen an, kann u.U. auch Suche erschweren

Suchalgorithmus : gibt jeder Gleichung Gewicht und Alter

Gewicht einer Gleichung Summe der Gewichte seiner Komponenten (analog Alter)

In jeder Iteration der Suchschleife entweder leichteste noch nicht betrachtete oder älteste noch nicht betrachtete Gleichung gewählt

kann durch Angabe eines Verhältnis spezifiziert werden

Die Suche

- William McCune startet Herbst 1996 Suche nach Lösung zu Robbins Problem
- bis zu 3 Unix-Workstations parallel im Einsatz
- Variation verschiedenster Parameter (Heuristiken etc.)
- nach ca. 14 Tagen und insgesamt ca. 5 CPU-Wochen Berechnungszeit dieses Ergebnis:

Die Suche

7	$n(n(n(x)+y)+n(x+y)) = y$	[Robbins equation*]
10	$n(n(n(x+y)+n(x)+y)+y) = n(x+y)$	[7 $\rightarrow$ 7]
11	$n(n(n(n(x)+y)+x+y)+y) = n(n(x)+y)$	[7 $\rightarrow$ 7]
29	$n(n(n(n(x)+y)+x+2y)+n(n(x)+y)) = y$	[11 $\rightarrow$ 7]
54	$n(n(n(n(n(x)+y)+x+2y)+n(n(x)+y)+z)+n(y+z)) = z$	[29 $\rightarrow$ 7]
217	$n(n(n(n(n(x)+y)+x+2y)+n(n(x)+y)+$ $n(y+z)+z)+z) = n(y+z)$	[54 $\rightarrow$ 7]
674	$n(n(n(n(n(n(x)+y)+x+2y)+n(n(x)+y)+$ $n(y+z)+z)+z+u)+n(n(y+z)+u)) = u$	[217 $\rightarrow$ 7]
6736	$n(n(n(n(3x)+x)+n(3x))+n(n(n(3x)+x)+5x)) = n(n(3x)+x)$	[10 $\rightarrow$ 674]
8855	$n(n(n(3x)+x)+5x) = n(3x)$	[6736 $\rightarrow$ 7,simp:54,flip]
8865	$n(n(n(n(3x)+x)+n(3x)+2x)+n(3x)) = n(n(3x)+x)+2x$	[8855 $\rightarrow$ 7]
8866	$n(n(n(3x)+x)+n(3x)) = x$	[8855 $\rightarrow$ 7,simp:11]
8870	$n(n(n(n(3x)+x)+n(3x)+y)+n(x+y)) = y$	[8866 $\rightarrow$ 7]
8871	$n(n(3x)+x)+2x = 2x$	[8865,simp:8870,flip]

letzte Zeile entspricht Weiser-Lemma $\exists c\ d.(c + d = c)$,
mit $c = 2x$ und $d = n(n(3x) + x)$

Die Suche

- erfolgreiche Suche benötigte fast 8 Tage und ca. 30 MB Speicher
- Suchparameter:
  - Längenbeschränkung der Gleichungen auf 70
  - super-0 Einschränkung
  - 'basic' Einschränkung
  - Suchalgorithmus Verhältnis Bevorzugung Gewicht:Alter = 1:1
- ca. 50.000 Gleichungen abgeleitet, Termersetzung versucht auf ca. 2,6 Mio. Termen, durchgeführt auf ca. 6.000

Ist das Kreativität? ...

Die Suche

- erfolgreiche Suche benötigte fast 8 Tage und ca. 30 MB Speicher
- Suchparameter:
  - Längenbeschränkung der Gleichungen auf 70
  - super-0 Einschränkung
  - 'basic' Einschränkung
  - Suchalgorithmus Verhältnis Bevorzugung Gewicht:Alter = 1:1
- ca. 50.000 Gleichungen abgeleitet, Termersetzung versucht auf ca. 2,6 Mio. Termen, durchgeführt auf ca. 6.000

Ist das Kreativität? ...

Die Kepler'sche Vermutung

T. C. Hales.
Cannonballs and honeycombs.
Notices of the American Mathematical Society, 47(4):440–449, 2000.
http://www.ams.org/notices/200004/fea-hales.pdf

T. Nipkow, G. Bauer and P. Schultz.
Flyspeck I: Tame Graphs.
In *Proc. of Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNCS*, pp. 21–35. Springer, 2006.
http://afp.sf.net/entries/Flyspeck-Tame.shtml

Kepler'sche Vermutung

Frage: *Wie lassen sich gleichgroße Kugeln am dichtesten stapeln?*

1611: Kepler postuliert, übliche pyramidenartige Stapelung (obere Kugel auf Dreieck aus Kugeln) mit Dichte $\frac{\pi}{\sqrt{18}}$ am dichtesten

1900: Behauptung wird Teil von Hilbert's 18tem Problem

1998: Tom Hales erklärt Aussage für bewiesen
Beweis beinhaltet 3 große Computer-Berechnungen
nach 4 Jahren Prüfung erklären die 12 Begutachter, sie wären sicher, dass Beweis korrekt

Kepler'sche Vermutung

Frage: *Wie lassen sich gleichgroße Kugeln am dichtesten stapeln?*

1611: Kepler postuliert, übliche pyramidenartige Stapelung (obere Kugel auf Dreieck aus Kugeln) mit Dichte $\frac{\pi}{\sqrt{18}}$ am dichtesten

1900: Behauptung wird Teil von Hilbert's 18tem Problem

1998: Tom Hales erklärt Aussage für bewiesen
Beweis beinhaltet 3 große Computer-Berechnungen
nach 4 Jahren Prüfung erklären die 12 Begutachter,
sie wären sicher, dass Beweis korrekt

Kepler'sche Vermutung

Frage: *Wie lassen sich gleichgroße Kugeln am dichtesten stapeln?*

1611: Kepler postuliert, übliche pyramidenartige Stapelung (obere Kugel auf Dreieck aus Kugeln) mit Dichte $\frac{\pi}{\sqrt{18}}$ am dichtesten

1900: Behauptung wird Teil von Hilbert's 18tem Problem

1998: Tom Hales erklärt Aussage für bewiesen
Beweis beinhaltet 3 große Computer-Berechnungen
nach 4 Jahren Prüfung erklären die 12 Begutachter,
sie wären sicher, dass Beweis korrekt

Kepler'sche Vermutung

Frage: *Wie lassen sich gleichgroße Kugeln am dichtesten stapeln?*

1611: Kepler postuliert, übliche pyramidenartige Stapelung (obere Kugel auf Dreieck aus Kugeln) mit Dichte $\frac{\pi}{\sqrt{18}}$ am dichtesten

1900: Behauptung wird Teil von Hilbert's 18tem Problem

1998: Tom Hales erklärt Aussage für bewiesen
Beweis beinhaltet 3 große Computer-Berechnungen
nach 4 Jahren Prüfung erklären die 12 Begutachter,
sie wären sicher, dass Beweis korrekt

Kepler'sche Vermutung

Frage: *Wie lassen sich gleichgroße Kugeln am dichtesten stapeln?*

1611: Kepler postuliert, übliche pyramidenartige Stapelung (obere Kugel auf Dreieck aus Kugeln) mit Dichte $\frac{\pi}{\sqrt{18}}$ am dichtesten

1900: Behauptung wird Teil von Hilbert's 18tem Problem

1998: Tom Hales erklärt Aussage für bewiesen
Beweis beinhaltet 3 große Computer-Berechnungen
nach 4 Jahren Prüfung erklären die 12 Begutachter,
sie wären 99% sicher, dass Beweis korrekt

Das Flyspeck-Projekt

- Hales mit Ergebnis nicht zufrieden
- startet Projekt mit Ziel formaler Beweis in Theorembeweiser
- FlysPecK: Akronym für *Formal Proof of Kepler*
- gemeinschaftliches Projekt, jeder darf sich beteiligen
- Idee: Berechnungen in OCaml, Beweise in HOL light
- heute: auch Coq und Isabelle beteiligt

Das Flyspeck-Projekt

- Hales mit Ergebnis nicht zufrieden
- startet Projekt mit Ziel formaler Beweis in Theorembeweiser
- FlysPecK: Akronym für *Formal Proof of Kepler*
- gemeinschaftliches Projekt, jeder darf sich beteiligen
- Idee: Berechnungen in OCaml, Beweise in HOL light
- heute: auch Coq und Isabelle beteiligt

Das Flyspeck-Projekt

- Hales mit Ergebnis nicht zufrieden
- startet Projekt mit Ziel formaler Beweis in Theorembeweiser
- FlysPecK: Akronym für *Formal Proof of Kepler*
- gemeinschaftliches Projekt, jeder darf sich beteiligen
- Idee: Berechnungen in OCaml, Beweise in HOL light
- heute: auch Coq und Isabelle beteiligt

Übersicht Beweis

grober Ablauf von Hales Beweis:

1. jedes mögliche Gegenbeispiel (mit dichterer Packung) führt zu tame graph
2. Aufzählung aller (endlich vielen) möglichen tame graphs (mittels Computer)
3. für jeden prüfen (wieder mittels Computer), dass kein Gegenbeispiel
4. dafür erstellt Hales *Archiv* aller tame graphs

Übersicht Beweis

grober Ablauf von Hales Beweis:

1. jedes mögliche Gegenbeispiel (mit dichterer Packung) führt zu tame graph
2. Aufzählung aller (endlich vielen) möglichen tame graphs (mittels Computer)
3. für jeden prüfen (wieder mittels Computer), dass kein Gegenbeispiel
4. dafür erstellt Hales *Archiv* aller tame graphs

Übersicht Beweis

grober Ablauf von Hales Beweis:

1. jedes mögliche Gegenbeispiel (mit dichterer Packung) führt zu tame graph
2. Aufzählung aller (endlich vielen) möglichen tame graphs (mittels Computer)
3. für jeden prüfen (wieder mittels Computer), dass kein Gegenbeispiel
4. dafür erstellt Hales *Archiv* aller tame graphs

Übersicht Beweis

grober Ablauf von Hales Beweis:

1. jedes mögliche Gegenbeispiel (mit dichterer Packung) führt zu tame graph
2. Aufzählung aller (endlich vielen) möglichen tame graphs (mittels Computer)
3. für jeden prüfen (wieder mittels Computer), dass kein Gegenbeispiel
4. dafür erstellt Hales *Archiv* aller tame graphs

Übersicht Beweis

grober Ablauf von Hales Beweis:

1. jedes mögliche Gegenbeispiel (mit dichterer Packung) führt zu tame graph
2. Aufzählung aller (endlich vielen) möglichen tame graphs (mittels Computer)
3. für jeden prüfen (wieder mittels Computer), dass kein Gegenbeispiel
4. dafür erstellt Hales *Archiv* aller tame graphs

Damit Behauptung bewiesen!

erinnert alles etwas an 4-Farben-Satz...

Formalisierung in Teilen

Formalisierung des Beweises in Teilen (für eine Formalisierung zu groß):

Gertrud Bauer, Tobias Nipkow:

Formalisierung der Aufzählung der tame graphs in Isabelle/HOL
als einziges Teilprojekt bereits abgeschlossen
im Folgenden vorgestellt

Roland Zumkeller:

Optimierungsprobleme in Coq
nichtlineare Ungleichungen in 6 Variablen
verwendet für Aussagen über Tetraeder (Volumen, Winkel, etc.)

Stephen Obua:

Verifikation von linearen Optimierungsschranken in Isabelle/HOL
Integration von linearer Optimierung in Isabelle

Planare Graphen

Formalisierung eines planaren Graphen:

- Fläche gegeben durch Knotenliste
  Äquivalenzrelation für Knotenlisten bzgl. Rotation
  finale und *nichtfinale* Flächen
- Graph durch Flächenliste (orientiert)

Wie formalisiert man eine Aufzählung von planaren Graphen?

Planare Graphen

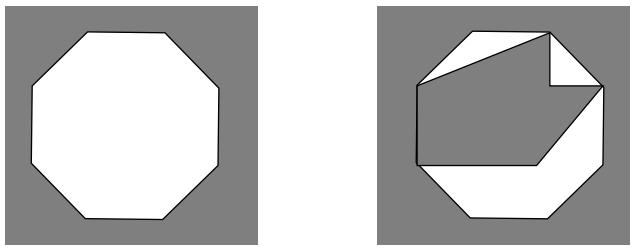
Formalisierung eines planaren Graphen:

- Fläche gegeben durch Knotenliste
  Äquivalenzrelation für Knotenlisten bzgl. Rotation
  finale und *nichtfinale* Flächen
- Graph durch Flächenliste (orientiert)

Wie formalisiert man eine Aufzählung von planaren Graphen?
Formalisierung von Tom Hales Ansatz: induktiv

1. Start mit Graphen aus 2 Flächen, einer finalen, einer nichtfinalen
2. solange Graph nichtfinale Fläche, Aufteilen dieser in eine finale und mehrere nichtfinale durch Einfügen von Knoten

Bsp:

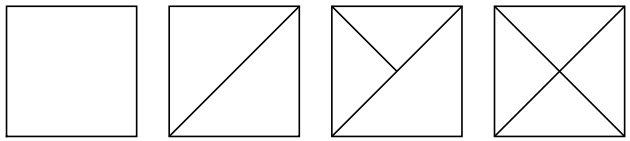


grau final, weiß nichtfinal

tame graphs

"*tameness*" zugesichert durch 8 Bedingungen:

1. Größe jeder Fläche mindestens 3, höchstens 8
2. jeder Knoten-3er-Zyklus ist Fläche
3. jeder Knoten-4er-Zyklus umgibt eine dieser Konfigurationen



4. jeder Knoten hat Grad höchstens 6
5. falls Knoten adjazent zu anderer Fläche als Drei- oder Viereck, Knotengrad höchstens 5
6. eine bestimmte Ungleichung bzgl. einer Tabelle muss gelten
7. eine zulässige Zuordnung von Gewichten zu Flächen existiert, die gesamt kleiner als 14800 ist (sichert Endlichkeit der Aufzählung)
8. keine 2 Knoten, die nur adjazent zu 4 Dreiecken sind, sind benachbart

Aufzählung der tame graphs

modifizierte Aufzählung planarer Graphen

- wirft Graphen weg, die nicht tame sind
- schneidet Suchbäume ab, wenn tame graphs nicht erreichbar

Formalisierung:

- verlässt sich auf Sicherheit der Aufzählung planarer Graphen
- liefert aber für tame Aufzählung das zentrale Resultat, dass modifizierte Aufzählung alle tame graphs generiert

Aufzählung der tame graphs

modifizierte Aufzählung planarer Graphen

- wirft Graphen weg, die nicht tame sind
- schneidet Suchbäume ab, wenn tame graphs nicht erreichbar

Formalisierung:

- verlässt sich auf Sicherheit der Aufzählung planarer Graphen
- liefert aber für tame Aufzählung das zentrale Resultat, dass modifizierte Aufzählung alle tame graphs generiert

Beweis durch Ausführung

- während Beweis Auswertung einer Funktion durch Übersetzung in effizientes Format und Ausführung
- existiert in anderen Theorembeweisern schon länger (ACL2, Coq, PVS)
- in Isabelle:
  1. Term t und alle Funktionen in t übersetzt in ML (falls möglich)
  2. Präprozessor ersetzt nichtausführbare Konstrukte durch Lemmas mit ausführbaren Funktionen
  3. t wird in ML zu u reduziert
  4. Resultat wird zu Theorem $t = u$

Beweis der Aufzählung der tame graphs

1. 8 Bedingungen zusammengefasst im Prädikat *tame*
2. Menge der tame graphs *TameEnum* (verwendet *tame*)
3. ausführbare Version der Menge *tameEnum*
4. *tameEnum* muss gegen das *Archiv* geprüft werden:
   "wenn planarer Graph tame, dann muss er im Archiv existieren"
   für jeden Graphen damit Test auf Graphisomorphie!

Übersicht Formalisierung

- Formalisierung stellt fest, dass 8. “tameness” Bedingung nur im Programmcode, nicht im Papierbeweis
- Archiv kann verkleinert werden von 5128 auf 2278 Graphen enthält bisher redundante Fälle
- Programm zur Aufzählung der tame graphs:
  statt 2200 Zeilen Java Code 600 Zeilen ML
- gesamte Formalisierung: 17000 Zeilen Isabelle
- Laufzeit 165 Minuten (Xeon):
  - Vollständigkeitsbeweis: 15 min
  - Ausführen der Aufzählung: 105 min
  - Vergleiche der resultierenden Graphen mit Archiv: 45 min
- 23 Mio. Graphen generiert und überprüft, davon 35000 final

Zusammenfassung

- mehr und mehr mathematische Beweise benutzen Programmierung (4-Farben-Satz, Kepler'sche Vermutung)
- Programmcode kann nicht durch "review" verifiziert werden
- um Korrektheit zu zeigen, Verifikation absolut nötig, dafür Theorembeweiser gut geeignet
- allerdings noch Probleme mit mathematischem Schliessen, Aufwand oft sehr gross im Vergleich mit Resultat
- "Kreativität" nur geschickte, erschöpfende Suche ohne Verständnis des Problems

noch ein paar Links

- `http://www.sciencemag.org/cgi/content/full/307/5714/1402a`
  populärwissenschaftlicher Artikel, der einige der hier gezeigten Themen bespricht
- `http://www.ams.org/notices/200811/`
  unter anderem noch ein Artikel zum 4-Farben-Satz und eine kurze Einleitung zu formalem Beweisen durch Tom Hales
  (aus mathematischem Journal *AMS*)