Symbolisches Rechnen Vorlesung Wintersemester 2006, 2014 Sommersemester 2021

Johannes Waldmann, HTWK Leipzig

6. Juli 2021

1 Einleitung

Symbolisches Rechnen: Beispiele: Zahlen

- numerisches Rechnen mit Maschinenzahlen

```
sqrt 2 + sqrt 3  ==> 3.1462643699419726
(sqrt 2 + sqrt 3)*(sqrt 2 - sqrt 3) ==> ...
```

- exaktes Rechnen (mit algebraischen Ausdrücken)
  $(\sqrt{2} + \sqrt{3}) \cdot (\sqrt{2} - \sqrt{3}) = \ldots$,
  maxima: `expand(%)`

Symbolisches Rechnen: Beisp.: Funktionen

- auf konkreten Daten: `let f x = (x+1)^2 in f 3.1 - f 3`
- auf symbolischen Daten: `diff((x+1)^2,x)`
- `subst([x=3],diff((x+1)^2,x))`
- eigentlich `diff(\x -> (x+1)^2)`
  mit `diff::(R -> R) -> (R -> R)`,
  aber da die Mathematiker Funktionen (höhere Ordnung) immer unzweckmäßig bezeichnen, um den Lambda-Kalkül zu vermeiden ...

Symbolisches Rechnen: Motivation

hat weitreichende Anwendungen:

- Lösen von (parametrisierten) Aufgaben*klassen*
  (für numerisches Rechnen muß Parameter fixiert werden)
- *exaktes* Lösen von Aufgaben
  (numer. R. mit Maschinenzahlen: nur Approximation)
- experimentelle, explorative, exakte Mathematik

ist nützlich im Studium, benutzt und vertieft:

- Mathematik (Analysis, Algebra)
- Algorithmen-Entwurf, -Analyse
- Prinzipien von Programmiersprachen

Überblick

- Zahlen (große, genaue)
- Vektoren (Gitterbasen)
- Polynome
- Terme, Term-Ersetzungs-Systeme
  (Anwendung: Differentiation, Vereinfachung)
- Gröbnerbasen (Termination, Vervollständigung)
- Geometrische Konfigurationen
- . . . und Beweise (Anwendung von Gröbnerbasen)
- Ausblick: $A = B$, Musik, Logik, Refactoring

Literatur

- Wolfram Koepf: *Computeralgebra*, Springer, 2006. http://www.mathematik.uni-kassel.de/~koepf/CA/
- Hans-Gert Gräbe: *Einführung in das Symbolische Rechnen, Gröbnerbasen und Anwendungen*, Skripte, Universität Leipzig http://www.informatik.uni-leipzig.de/~graebe/skripte/
- Franz Baader and Tobias Nipkow: *Term Rewriting and All That*, Cambridge, 1998. http://www21.in.tum.de/~nipkow/TRaAT/
- weitere Literatur siehe z.B. https://portal.risc.jku.at/Members/hemmecke/teaching/ppscs

Software

- wir benutzen
  - Maxima http://maxima.sourceforge.net/
  - FriCAS https://github.com/fricas/fricas/
  - Geonext http://geonext.uni-bayreuth.de/
  - GHC http://www.haskell.org/ghc/
- ist alles im Pool installiert (ssh, tmux, x2go)
- allgemeine Hinweise, auch zum Selbstbauen https://imweb.imn.htwk-leipzig.de/~waldmann/etc/cas/

Beispiel: S.R. und Term-Ersetzung

Regeln für symbolisches Differenzieren (nach `t`):

```
D(t) -> 1          D(constant) -> 0
D(+(x,y)) -> +(D(x),D(y))
D(*(x,y)) -> +(*(y,D(x)),*(x,D(y)))
D(-(x,y)) -> -(D(x),D(y))
```

Robert Floyd 1967, zitiert in: Nachum Dershowitz: *33 Examples of Termination*, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.9447

- Korrektheit? Termination? Komplexität?
- Strategie (Auswahl von Regel und Position)?
- ausreichend? angemessen?

Beispiel: Termersetzung (cont.)

```
data E = Zero | One | T
       | Plus E E | Times E E  deriving Show

e :: E
e = let b = Plus T One in Times b b

d :: E ->  E
d e = case e of
    Zero -> Zero ; One -> Zero ; T -> One
    Plus x y -> Plus (d x) (d y)
    Times x y ->
        Plus (Times y (d x)) (Times x (d y))
```

Beispiel: Inverse Symbolic Calculator

- `http://wayback.cecm.sfu.ca/projects/ISC/ISCmain.html`

  zur Bestimmung ganzzahliger Relationen (z.B. zwischen Potenzen einer numerisch gegebenen Zahl)

- `sqrt(2+sqrt 3)  ==>  1.9318516525781366`

  ```
  integer relations algorithm, run:
  K = 1.9318516525781366
                                  4      2
  K satisfies the polynomial, X  - 4 X  + 1
  ```

  mit LLL-Algorithmus (Lenstra, Lenstra, and Lovasz, 1982), der kurzen Vektor in geeignetem Gitter bestimmt.

Hausaufgaben KW 14, Organisatorisches

1. zum Haskell-Programm zum Symb. Differenzieren:
   - füge Syntax und Regel für Quotienten hinzu
   - schlage Regeln zur Vereinfachung vor
2. ISC Simple Lookup and Browser sagt für $\sqrt{2+\sqrt{3}}$:

   ```
   Mixed constants with 5 operations
   1931851652578136 = 1/2/sin(Pi/12)
   ```

   begründen Sie das (geometrisch oder schriftlich)
3. ein Polynom mit Nullstelle $\sqrt[2]{2} + \sqrt[3]{3}$ bestimmen, nachrechnen.
4. Geonext: Satz von Napoleon illustrieren (gleichseitige Dreiecke über den Seiten eines beliebigen Dreiecks)
5. eigener Rechner: `rlwrap maxima` installieren,

   Rechner im Pool: `ssh` und `tmux` ausprobieren, auch Management von Sessions, Windows, Panes (split horizontal, vertikal), vgl. `https://news.ycombinator.com/item?id=26670708`

Organisatorisches:

- in Gitlab.Imn-Projekt einschreiben
- Hausaugabe: Wiki anmelden, Issue: diskutieren, ggf. MR
- Prüfungszulassung: Hausaufgaben, autotool
- Prüfung: mündlich, ggf. mit Bezug auf Projekt (= längere Hausaufgabe)
- Hausaufgaben (und Projekt): jeweils 2 Leute

2 Zahlen

Überblick

- exakte Zahlen: natürlich, ganz, rational

  Darstellungen: Positionssystem, Bruch

  Rechnungen: Addition, Multiplikation

- beliebig genau genäherte Zahlen

  (berechenbare reelle Zahlen)

  Darstellungen: Positions-System, Kettenbruch

  Rechnungen: arithmetische und irrationale Funktionen

- später:

  exaktes Rechnen mit algebraischen Zahlen

Darstellung natürlicher Zahlen

- die Zahl $n \in$ im Positionssystem zur Basis B:

  $n = \sum_{k \geq 0} x_k B^k$

  mit $\forall i : 0 \leq x_i < B$ und $\{i \mid x_i \neq 0\}$ endlich

  Bsp: $25 = 1 \cdot 3^0 + 2 \cdot 3^1 + 2 \cdot 3^2 = [1, 2, 2]_3$

- Darstellung ist eindeutig, kann durch fortgesetzte Division mit Rest bestimmt werden $\quad 25 = 1 + 3 \cdot 8, 8 = 2 + 3 \cdot 2$

- praktisch wählt man die Basis

  - 10 für schriftliches Rechnen
  - historisch auch 60 (Zeit- und Winkelteilung)
  - 2 (oder 2^w) binäre Hardware, maschinennahe Software

Natürliche Zahlen, Addition

- Darstellung

```
type Digit = Word64 ; data N = Z | C Digit N
```

- Semantik

```
value :: N -> Natural
value Z = 0 ; value (C x xs) = x + 2^64 * value xs
```

- Rechnung

```
instance Num N where (+) = plus_carry False
plus_carry :: Bool -> N -> N -> N
plus_carry cin (C x xs) (C y ys) =
  let z = bool 0 1 cin + x + y
      cout = z < x || z < y || ...
  in  C z (plus_carry cout xs ys)
plus_carry ...
```

Rekursive Multiplikation

- anstatt „Schulmethode“: rekursive Multiplikation
  $(p + qB)(r + sB) = pr + (ps + qr)B + qsB^2$
  Bsp. $B = 100$, $(12 + 34 \cdot 100)(56 + 78 \cdot 100) = 12 \cdot 56 + \ldots$
  $(1 + 2 \cdot 10)(5 + 6 \cdot 10) = 1 \cdot 2 + \ldots$
- $M(w) =$ Anzahl elementarer Operationen (Plus, Mal) für Multiplikation w-stelliger Zahlen nach diesem Verfahren
  $M(1) = 1, M(w) = 4 \cdot M(w/2) + 2 \cdot w$
  $M(2^k) = 3 \cdot 4^k - 2^{k+1}$, also $M(w) \in \Theta(w^2)$
- also wie bei Schulmethode, aber das läßt sich verbessern, und darauf muß man erstmal kommen

Karatsuba-Multiplikation

- A. Karatsuba, Yu. Ofman 1962
- $(p + qB)(r + sB) = pr + (ps + qr)B + qsB^2$
  $= pr + ((p + q)(r + s) - pq - rs)B + qsB^2$
- $K(w) =$ Anzahl elementarer Operationen (Plus, Mal) für Multiplikation w-stelliger Zahlen nach diesem Verfahren
  $K(1) = 1, K(w) = 3 \cdot K(w/2) + 5 \cdot w$
  (eine Multiplikation weniger, drei Additionen mehr)
- asymptotisch mit Ansatz $K(w) \approx C \cdot w^e$
  $C \cdot w^e = 3Cw^e/2^e + 5w \Rightarrow 1 \approx 3/2^e$, also $e = \log_2 3 \approx 1.58$
- explizite Werte für $w = 2^k$,
  ab wann besser als Schulmethode?

Ganze Zahlen

- Darstellung: Bsp GMP (GNU Multi Precision Library)
  Natürliche Zahl (magnitude) mit Vorzeichen (sign)
  nicht Zweierkomplement, denn das ist nur für Rechnungen modulo 2^w sinnvoll
- GHC (`integer-gmp`)

```
data Integer = S Int | Jp BigNat | Jn BigNat
data BigNat = BN ByteArray
```

- NB: ganzzahlige Zahlbereiche in Haskell/GHC:
  - `Int` (ist immer die falsche Wahl)
  - Maschinenzahlen (mod 2^{64}): `Int64`, `Word64`,
  - beliebig große: `Integer`, `Numeric.Natural`

Rationale Zahlen

- rationale Zahl q ist Paar von ganzen Zahlen (z, n)

  ```
  data Q = Q Integer Integer
  ```

- normalisiert: $n \geq 1$ und $\gcd(|z|, |n|) = 1$
- normalisierte Darstellung ist eindeutig
- Vorteil:

  semantische Gleichheit (dieselbe Zahl wird bezeichnet) *ist* syntaktische Gleichheit (der Komponenten)

- Nachteil (?)

  nach jeder arithmetischen Operation normalisieren

- nicht normale Zahlen verhindern: Konstruktor `Q` nicht exportieren (sonst `let q = Q 8 12`)

Explosion der Stellenanzahl (Beispiel)

- Funktion $f :\rightarrow: x \mapsto x/2 + 1/x$
- $x_k = f^k(1)$, $x_0 = 1, x_1 = 3/2, x_2 = 17/12, x_3 = 577/408$
- Zähler (und Nenner) in jedem Schritt (wenigstens) quadriert, d.h., Stellenzahl verdoppelt (Bsp: x_{10})

  $p/q \mapsto p/(2q) + q/p = (p^2 + 2q^2)/(2pq)$

- das ist typisch für Folgen arithmetischer Op.,

  Normalisierung wirkt nur selten verkleinernd.

- die Folge x_k nähert $\sqrt{2}$ an: $(p_k/q_k)^2 - 2 = 1/q_k^2$

  Konvergenz ist quadratisch (jeder Schritt quadriert den Fehler, verdoppelt Anzahl gültiger Stellen)

Endliche und unendliche „Dezimal“brüche

- Darstellug von Zahlen in $[0 \ldots 1)$
  mit Basis $1/B$ und Ziffern $\in \{0 \ldots B-1\}$
- Bsp: $B = 10$, $0.314 = 0 \cdot B^0 + 3 \cdot B^{-1} + 1 \cdot B^{-2} + 4 \cdot B^{-3}$
- hier sind auch unendliche Ziffernfolgen sinnvoll,
  bezeichnet Limes der Werte der endlichen Partialfolgen
- Konversion

```
decimal :: Rational -> [ Nat ]
decimal x = let q = truncate (fromRational x)
            in  q : decimal (10 * (x-q))
```

  Anwendung: vorige Näherung von $\sqrt{2}$

Berechenbare reelle Zahlen

- beschrieben wird hier nicht die exakte (symbolische) Rechnung, sondern eine konvergente Näherung
- Def. reelle Zahl r mit $0 \leq r < 1$ heißt *berechenbar* (zur Basis B), wenn die Funktion $d :\to \{0, \ldots, B-1\}$, welche die Darstellung von x zur Basis $1/B$ bestimmt, berechenbar ist (z.B. durch eine Turingmaschine)
- jede rationale Zahl ist berechenbar
  (Beweis: Dezimalbruch ist endlich oder periodisch)
- $\sqrt{2}$ ist berechenbar (Beweis: vorige Folge x_k)
- Menge der berechenbaren Zahlen ist abgeschlossen unter arithmetischen Operationen (und weiteren)

Potenzreihen, Exponentialfunktion

- Satz von Taylor: wenn f oft genug diff-bar,
  dann $f(x_0 + d) = \sum_{k=0}^{n-1} f^{(k)}(x_0) d^k/k! + \Delta_n$
  mit $\exists 0 \leq d' \leq d : \Delta_n = f^{(n)}(x_0 + d') d^n/n!$
- Bsp: $f(x) = \exp(x)$, dann $f = f' = f'' = f^{(3)} = \ldots$ und
  $\exp(0 + d) = 1 + d + d^2/2 + d^3/6 + \ldots$
- ```
  take 100 $ decimal
    $ sum $ take 100 $ scanl (/) (1::Rational) [1..]
      ==>   [2,7,1,8,2,8,1,8,2,8,4,5,9,0,4 ... ]
  ```
  Fehlerabschätzung? (reicht die zweite 100 für die erste?)
- $\exp(1) = \exp(1/2)^2$, diese Reihe konvergiert schneller

Potenzreihe für Wurzelfunktion

- Taylor-Reihe von $\sqrt{x}$ an der Stelle 1
  Ableitungen mit maxima:
  `diff(sqrt(x),x,5);` $105/32 \cdot x^{-9/2}$
  `diff(sqrt(x),x,6);` $-945/64 \cdot x^{-11/2}$
  Vermutung $f^{(n)}(1) = (-1)^{n+1} \cdot (2n-1)!! \cdot 2^{-n}$
- $\sqrt{1+d} = 1 + \sum_{k>0} (-1)^{k+1} \frac{(2k-1)!!}{2k!!} d^k$
  $= 1 + \frac{1}{2} \cdot d - \frac{1 \cdot 3}{2 \cdot 4} \cdot d^2 + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \cdot d^3 - \ldots$
- für $\sqrt{2}$: Berechnung als $\sqrt{1+1}$ konvergiert langsam
  besser: $\sqrt{2} = 7/5\sqrt{1 + 1/49}$

Hausaufgaben

1. Multiplikation in GMP (GNU Multiprecision Library) `https://gmplib.org/manual/Multiplication-Algorithms`
   (a) Karatsuba-Rechnung ist dort etwas anders als hier auf der Folie, warum?
  ```

(b) GHC verwendet GMP für den Typ `Integer`.

Bestimmen Sie experimentell den Anstieg der Rechenzeit bei Verdopplung der Stellenzahl, z.B.

```
:set +s
x = 10^10^8 :: Integer
odd x -- damit x ausgewertet wird
odd ((x-1)*(x+1)) -- die eigentliche Messung
  True
  (3.73 secs, 166,159,792 bytes)
y = x*x -- hat doppelte Stellenzahl
```

Ist die Anzahl der Bytes plausibel?

Diskutieren Sie mögliche verkürzte Auswertungen für `odd ...`. Kann GMP/GHC das?

(c) Zusatz: warum ist (oder erscheint) `(x+1)^2` schneller als `(x+1)*(x+1)`?

2. diskutieren (Zusatz: implementieren) Sie die Darstellung von ganzen Zahlen mit negativer Basis $B \leq -2$

   (und nichtnegativen Ziffern $\in \{0, \ldots, |B| - 1\}$ wie bisher)

   Bsp: $B = -2$,

   $-3 = 1 \cdot B^0 + 0 \cdot B^1 + 1 \cdot B^2 + 1 \cdot B^3 = 1 + 0 + 4 - 8$

   (a) Eindeutigkeit, Konstruktion

   (b) Arithmetik (Nachfolger, Addition, Multiplikation)

3. Bestimmen Sie die Taylor-Reihe für den Arcustangens an der Stelle 0

   wie auf Folie *Potenzreihe für Wurzelfunktion*

   Bestimmen Sie damit $x = \arctan(1/2), y = \arctan(1/3)$ auf (z.B.) 20 Stellen.

   Begründen Sie $x + y = \pi/4$. Rechnen Sie den Wert für π aus und vergleichen Sie mit einer verläßlichen Quelle.

   Kann man π nach diesem Verfahren, aber mit anderen Parametern, besser bestimmen? (mehr Stellen bei gleichem Aufwand)

4. Bestimmen Sie die Taylor-Reihe für den (natürlichen) Logarithmus an der Stelle 1

   Bestimmen Sie damit $a = \log(6/5), b = \log(9/8), c = \log(10/9)$ auf (z.B.) 100 Stellen

   und daraus $\log 2$ als eine Linearkombination.

   Geben Sie eine bessere Menge solcher Brüche zur Bestimmung von $\log 2$ an.

(Definieren Sie *solches* und *besser* exakt. Mit anderen Worten, spezifizieren Sie die entsprechende Autotool-Aufgabe.)

3 Zahlen, Vektoren, Gitter

Wiederholung, Motivation

- Abstand von $\log(1+d)$ zu Taylor-Reihe bis Grad $n-1$
  ist $\Delta_n(d') = f^{(n)}(1+d') \cdot d^n/n!$ für ein $d' \in [0,d]$
  $\log^{(n)}(x) = (-1)^{n+1}(n-1)!/x^n$, $|\Delta_n(d')| \leq |\Delta_n(0)| = d^n/n$
  z.B. $d \leq 1/10$: bis Grad n: n gültige Dezimal-Stellen
- für die Bestimmung von $\log 2$ ist (z.B., u.a.) nützlich
  $1 + 1/80 = 81/80 = 2^{-4} \cdot 3^4 \cdot 5^{-1}$
  $\log(1+1/80) = -4\log(2) + 4\log(3) - 1\log(5)$
- gibt es (viele) solche Brüche? wie findet man sie?

Die Hamming-Folge

- diese Aufgabe
  - Eingabe: endliche Folge $m \in^*$, z.B. $m = [2,3,5]$
  - Ausgabe: aufsteigende Folge aller Zahlen $\prod_i m_i^{e_i}$
    $1, 2, 3, 4 = 2^2, 5, 6 = 2 \cdot 3, 8 = 2^3, 9 = 3^2, 10 = 2 \cdot 5, \ldots$
- hat eine elegante Lösung mit Iteratoren (lazy Listen)

```
merge (x:xs) (y:ys) = case compare x y of ...
hamming m = xs where
  mul f = map (*f) xs
  xs = 1 : foldr1 merge (map mul m)
```

- benachbarte Elemente sind Kandidaten für die log-Berechnung (Bsp: $6/5, 8/9, 10/9$)

Benachbarte zerlegbare Zahlen

- benachbarte Zahlen aus `hamming [2,3,5]` sind

  $\ldots, (5,6), (8,9), (9,10), (15,16), (24,25), (80,81)$

  $[2,3,5,7] \Rightarrow \ldots, (224,225), (2400,2401), (4374,4375)$

  offenbar nur endliche viele solche Paare

- auch wenn man größere Abstände gestattet, z.B. bis 3:

  $[2,3,5] \Rightarrow \ldots (125,128), (160,162), (240,243)$

  (für Berechnung des log ist $128/125$ nützlich)

- gibt es beliebig große (a,b,c) mit $a+b=c$,

  $\gcd(a,b)=1$ und abc hat nur kleine Primfaktoren?

  Bsp: $a=5^3, b=3, c=2^7$ und $abc \in \{2,3,5\}$

Die ABC-Vermutung

- exakte Aussage über die Menge: ***große*** (a,b,c), $a+b=c$, $\gcd(a,b)=1$ und abc hat nur ***kleine*** Primfaktoren
- Def: das Radikal, (n) = Produkt d. Primfaktoren v. n,
  Bsp: $(1000) = (2^4 \cdot 5^4) = 2 \cdot 5 = 10$
- Ansatz: wenn $a+b=c, \gcd(a,b)=1$, dann $c \leq (abc)$?
  ist aber (unendlich oft) falsch, $a=1, b=3^{2^n}-1, c=3^{2^n}$
- Oesterle, Masser 1985: für jedes $\epsilon > 0$ gibt es nur endlich viele a,b,c mit: $a+b=c, \gcd(a,b)=1, c > (abc)^{1+\epsilon}$
- Beweis behauptet Mochizuki (2012), von Fachwelt nicht akzeptiert (vgl. Peter Woit: *ABC is still a conjecture*, 2021)
- Def: die Qualität $q(a,b,c) = \log(c)/\log((abc))$
  Rekord derzeit: $q(2, 3^{10} \cdot 109, 23^5) = 1.62991\ldots$

Gitter: Definition, Beispiel, SVP

- ein Gitter (lattice) L mit Basis $B = [b_1, \ldots, b_n] \subset^d$
  ist die Menge $L(B) = \{\sum_{k=1}^{n} c_k b_k \mid \forall k : c_k \in\}$
- Beispiel: $b_1 = [5, 3]^T, b_2 = [8, 5]^T, B = [b_1, b_2] = \begin{pmatrix} 5 & 8 \\ 3 & 5 \end{pmatrix}$
  weitere Elemente von $L(B)$ sind
  $[13, 8]^T = b_1 + b_2$, $[3, 2]^T = -b_1 + b_2$,...
- Shortest Vector Problem (SVP):
  - Eingabe B,
  - Ausgabe: ein kürzester Vektor in $L(B) \setminus \{0\}$
- Lösungsverfahren: $d = 2$ exakt, d größer: genähert (LLL)
- Anwendungen: ganzzahlige Beziehungen, Kryptographie

Ganzzahlige Beziehungen

- gegeben: Näherungs-Wert einer algebraischen Zahl, Bsp: $x = 0.41421356237309515$
  gesucht: das Minimalpolynom für x,
  d.h., Zahlen $c_0, \ldots, c_n \in$ mit $0 = \sum_k c_k x^k$
- Ansatz: bestimme kurzen Vektor in
  Gitter mit Basis $b_k = [0 \ldots 0, 1, 0 \ldots 0, \lceil F \cdot x^k \rfloor]$
  die 1 auf Position k und F groß (Bsp: $F = 10^5$)

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 10000 & 41421 & 17157 \end{pmatrix}$$

Exakte Rechnung für Dimension 2

- Beispiel $b_1 = [5, 3]^T, b_2 = [13, 5]^T$
- geometrische Intuition:
  - $|b_1| \leq |b_2|$ (sonst vertauschen)

  - $c \in$ bestimmen, das $|b_2 - c \cdot b_1|$ minimiert
  - falls $c \neq 0$, wiederhole mit neuer Basis $[b_2 - c \cdot b_1, b_1]$

- im Beispiel: $2b_1 = [10, 6]^T, 3b_1 = [15, 9]^T$
  Abstände zu b_2: $\sqrt{3^2 + 1^2}$, $\sqrt{2^2 + 4^2}$, Min. für $c = 2$,
  neue Basis $[b_2 - 2b_1, b_1] = [[3, 1]^T, [5, 3]^T]$, weiter ...

- Anteil von b_2 in Richtung b_1 ist $b_1 \cdot \frac{\langle b_1, b_2 \rangle}{\langle b_1, b_1 \rangle}$, also $c = \left\lceil \frac{\langle b_1, b_2 \rangle}{\langle b_1, b_1 \rangle} \right\rfloor$,
  Schluß ($c = 0$), wenn $|\langle b_1, b_2 \rangle| < (1/2) \cdot \langle b_1, b_1 \rangle$

Bemerkungen zu diesem Verfahren

- für $c \in$ würde man das Lot von b_2 auf b_1 fällen
  Verallgemeinerung auf beliebige Dimension
  ergibt Verfahren von Gram-Schmidt zur Konstruktion einer orthogonalen Basis $i \neq j \Rightarrow \langle b_i, b_j \rangle = 0$
  $b_i^* = b_i -$ Anteil von b_i in $\{b_1, \ldots, b_{i-1}\}$

- wir sind aber in (mit Absicht — Gitter)
  - Dimension 1: wiederholtes Abziehen des Kleinem vom Großen ist ... der Euklidische Algorithmus
  - Dim. 2 geht auch (eben gesehen)
  - Verallgemeinerung auf höhere Dim. ist schwierig
    und effizient nur genähert möglich

Gitterbasis-Reduktion: LLL

- Algorithmus von Lenstra, Lenstra und Lovasz (1982)
- Gram-Schmidt-Koeffizienten $\mu_{ij} = \frac{\langle b_i, b_j \rangle}{\langle b_j, b_j \rangle}$
  Gram-Schmidt-Basisvektoren $b_i^* = b_i - \sum_{j<i} \mu_{ij} b_j^*$
- Basis heißt *size reduced*, wenn $\forall i, j : |\mu_{ij}| \leq 1/2$
  d.h., kann durch keinen Austausch $b_i' = b_i - c \cdot b_j$ verbessert werden
- LLL reduced, wenn außerdem $(3/4)|b_i^*|^2 \leq |\mu_{i+1,i} b_i^* - b_{i+1}^*|^2$

- LLL-Algorithmus konstruiert solche Basis durch Reduzieren und Vertauschen, b_1 ist dann kurz:
  $|b_1| \leq 2^{(n-1)/2}\lambda_1$, wobei $\lambda_1 = \min\{|v| : v \in L(B) \setminus \{0\}\}$
- O. Regev: `https://cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/lll.pdf`

Anwendung: Integer Relations

- (Wdhlg) für die Bestimmung von $\log 2$ ist nützlich:
  $\log(1 + 1/80) = -4\log(2) + 4\log(3) - 1\log(5)$
  wir suchen kurze Vektoren in Gitter mit Basis
  $[[1, 0, 0, F\log(2)]^T, [0, 1, 0, F\log(3)]^T, [0, 0, 1, F\log(5)]^T]$
- Implementierungen:
  - Paket/Algorithmus LLLReduction in FriCAS, `https://fricas.github.io/book.pdf` 9.52
  - `http://wayback.cecm.sfu.ca/projects/ISC/ISCmain.html`
    benutzt PSLQ, Bailey, Ferguson 1992, `http://www.davidhbailey.com/dhbpapers/pslq.pdf`

Anwendung: ABC-Tripel

- De Weger 1987, `https://research.utwente.nl/files/6561317/Weger87solving.pdf` Abschnitt 5.E

  Tripel mit $q = 1.6$ und: ... These results do not seem to yield any heuristical evidence for the truth or falsity of the above mentioned conjecture.
- Tim Dokchitser 2003, `https://arxiv.org/abs/math/0307322`
  - wähle (kleine) Primzahlen $M = \{p_1, \ldots, p_n\} \subset$
  - erzeuge Zahlen $A, B, C \in (M)$
  - bestimme integer-relation: x, y, z mit $xA + yB + zC = 0$

  mit etwas Glück stören die Primfaktoren von x, y, z wenig.

Hausaufgaben

1. für eine Taylor-Entwicklung einer Funktion f
   (z.B. $\log(1+d)$, $\sqrt{1+d}$) und (z.B.) $n=3, d=0.1$
   zeichnen Sie (zwei Kurven in ein Bild) (maxima: `plot2d`)
   - die Differenz von f zu Taylor-Reihe bis $n-1$,
     Bsp: $\log(1+x)-(x-x^2/2)$
   - das Restglied $\Delta_n(d')$, Bsp: $d^3/(1+d')^3/3$

   und überprüfen Sie den Satz (es gibt ein $d' \in [0,d] \ldots$)
   Welche Werte/Punkte in der Zeichnung sind zu vergleichen?

2. ABC-Vermutung (evtl. Teil-Aufgaben verteilen)
   (a) Beispiele und Beweis:
   es gilt $2^{n+2} \mid 3^{2^n}-1$, also $c > (abc) \cdot 2^{n+1}/3$
   vgl. Marc Saul `https://cims.nyu.edu/cmt/assets/pdfs/HS_Problems/ABC_Conjecture.pdf`
   (b) bestimmen Sie experimentell (brute-force) abc-Tripel mit hoher Qualität.
   (c) Was ist das Resultat des Projektes `ABC@Home`?

3. Integer relations: LLLReduce in FriCAS ausprobieren, für in der VL genannte Beispiele (oder andere)

4 On Numbers And Games

Einordnung

- bisher: Zahlen: alles, womit man rechnen kann (?)

  Beispiele: natürliche, gebrochene, (approx.) reelle

- jetzt: symbolisches Rechnen mit Werten von (endlichen Zweipersonen-) Spielen (mit vollständiger Information)

  Quelle: John H. Conway: ONAG, 1976

  modellieren Spiel als Graph (Zug als Kante)

- danach: abstrakte Ersetzungssysteme (ARS), (= gerichtete Graphen), Termination und Konfluenz

- später: konkrete Ersetzungssysteme:

  auf Termen, auf Polynomen (Gröbnerbasen)

Toads and Frogs

- Konfiguration ist Zeichenkette über Alphabet $\Sigma = \{T, F, \circ\}$ (Kröte, Frosch, Leer),

  Züge: Spieler L: $T\circ \to \circ T, TF\circ \to \circ FT$,

  Spieler R (frog): $\circ F \to F\circ, \circ TF \to FT\circ$

- Ü: Graph aller Konfigurationen für Start $T \circ F$, $T \circ^k F$, ...

- allgemeine Analyse ist NP-vollständig,

  T. Thanatipanonda, D. Zeilberger 2007/8: `https://arxiv.org/abs/0710.4951`, `https://arxiv.org/abs/0804.0640`

Hackenbush

- Konfiguration: Graph $G = (V, E)$, Kantenfärbung $f : E \to \{\text{rot}, \text{blau}, \text{grün}\}$, Teilmenge $B \subseteq V$,

  jeder Knoten $v \in V$ von B erreichbar.

- Spielzüge:

  Spieler L: eine blaue oder grüne Kante löschen, Spieler R: eine rote oder grüne Kante löschen,

  danach alle nicht geerdeten Komponenten löschen

- Ü: Konfigurationsgraph für Ketten (links geerdet)

  LR, LRR, ...

Abstrakte Zweipersonen-Spiele

- Spiel G: gerichteter Graph auf Konfigurationen
- Kanten sind gefärbt, Farbe zeigt an,
  welcher Spieler (L oder R) den Zug ausführen darf.
- wir vergessen die Identität der Knoten und betrachten nur noch ihre Nachfolgermengen, erhalten Definition
  ein Spiel ist ein Paar von Mengen von Spielen.
- Bsp: das einfachste Spiel ist $(\emptyset, \emptyset)$, wir nennen das 0, weitere Spiele: $1 := (\{0\}, \emptyset)$, $* = (\{0\}, \{0\})$, $(\{0\}, \{0, 1\})$,
- Bezeichnungen: $G = (G^L, G^R)$
  G^L : die Optionen (Nachfolger) von G für L, G^R : die Optionen (Nachfolger) von G für R,
- Abkürzung: $(\{0\}, \{0, 1\})$ als $\{0 \mid 0, 1\}$

Literatur

- John H. Conway: On Numbers and Games, 1976
  (review: `https://www.maa.org/press/maa-reviews/on-numbers-and-games`)
- Elwyn R. Berlekamp, John H. Conway, Richard K. Guy, Winning Ways for Your Mathematical Plays, 1981
- Donald E. Knuth: Surreal Numbers, 1974 `https://www-cs-faculty.stanford.edu/~knuth/sn.html`
- J.W., VL Kombinatorische Spieltheorie 2001 `http://www.imn.htwk-leipzig.de/~waldmann/edu/ancient/ws01/kst/` Abschnitt KST in VL KI, 2018 `http://www.imn.htwk-leipzig.de/~waldmann/edu/ss18/ki/folien/main.8.pdf`

Implementierungen

- einfache Modellierung

  ```
  import qualified Data.Set as S
  data Game = Game (S.Set Game) (S.Set Game)
  ```

  `https://gitlab.imn.htwk-leipzig.de/autotool/all0/-/tree/master/collection/src/Game/CGT`

  Vorteil: offensichtlich korrekt,

  Nachteil: kein explizites sharing, $\Rightarrow$ keine dyn. Progr.

- David Wolfe: The Gamesman's Toolkit, 1994 `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.5659`

  aber wo sind die Quelltexte?

Der Status eines Spieles

- ...kommt drauf an, wer beginnt!

  Selbst wenn das in der Wurzel immer L ist, können in den Summanden einer *disjunkten Summe* (nächste Folie) L oder R beginnen.

- ein Spiel heißt ..., wenn bei idealem Spiel:

	L beginnt	R beginnt
positiv	L gewinnt	L gewinnt
negativ	R gewinnt	R gewinnt
null	R gewinnt	L gewinnt
unscharf (fuzzy)	L gewinnt	R gewinnt

- Ü: Status der Spiele $0 = \{|\}, 1 = \{0\,|\}, \{0 \mid 0, 1\}$

  Ü: ein größeres neutrales Spiel? ein unscharfes Spiel?

Das entgegengesetzte Spiel

- für ein Spiel G bezeichnet $-G$ das Spiel, das aus G entsteht durch Vertauschen aller Optionen in allen Teilspielen.

- Ü: Negation in Hackenbusch, in Toads and Frogs

- Def: $-G = (\{-g \mid g \in G^R\}, \{-g \mid g \in G^L\})$
- Bsp: $-0 = 0$, $-\{0,0\} = \ldots$, $-1 = \ldots$
- Satz: $-(-G) = G$
- Satz: G positiv $\iff$ $-G$ negativ;
  G null $\iff$ $-G$ null; G unscharf $\iff$ $-G$ unscharf.

Die Summe von Spielen

- die *disjunkte Summe* von Spielen:

  in $G + H$ kann jeder Spieler in G ziehen (und H bleibt unverändert), oder in H ziehen (und G bleibt).

  ieser erste Zug erreicht wieder eine disjunkte Summe
- Ü: Summe in Hackenbusch, $LR + LR$.
- wenn G und H Spiele sind, dann ist $G + H$ das Spiel
  $(\{g + H \mid g \in G^L\} \cup \{G + h \mid h \in H^L\}, \{g + H \mid g \in G^R\} \cup \{G + h \mid h \in H^R\})$
- „praktische Anwendung“ der kombinatorischen Spieltheorie (CGT): das Rechnen mit solchen Summen ist effizienter als Analyse des vollständigen Spielbaums

Die Äquivalenz von Spielen

- Def: $G - H := G + (-H)$
- Def: $G \approx G'$, falls $G + (-G')$ den Status null hat.

  Wdhlg: Status null: der Nachziehende gewinnt
- Bsp: $1 \not\approx 0$, $1 \approx 1$, $\forall G : G \approx G$.
- Bsp (Hackenbush): $L + L + L \approx LLL$, $L \approx 1$,
- $LR + LR \approx L$, deswegen bezeichnen wir LR als $1/2$
- Satz: $G \approx G' \Rightarrow (G + H) \approx (G' + H)$

  Beweis: Strategie f. Nachziehenden in $(G + H) - (G' + H)$
- wir schreiben ab jetzt $G = H$ für $G \approx H$,

  Anwendung: $G - G = 0$

Eine Halbordnung auf Spielen

- Def: G hat Status *nicht-negativ*, wenn G positiv oder G null, d.h.: wenn R beginnt, gewinnt L.
- Def: $G \leq H$, falls $(H - G)$ nichtnegativ;
  Def: $G < H$, falls $(H - G)$ positiv (L gewinnt immer)
- Bsp: $1/2 \leq 1$, $0 < 1$, $0 \not< *$, $* \not< 0$,
- Satz: Relation $\leq$ ist Halbordnung, aber nicht total.
- Def: (Hackenbush) LR^k heißt $1/2^k$,
  denn $LR^{k+1} + LR^{k+1} = LR^k$ und $LR^0 = L = 1$
- $\forall k : -(1/2^k) < * < 1/2^k$

Mehr zu Hackenbush

- Hilfssatz: $i < j \Rightarrow LR^i > LR^j$.
- Satz: $LR^k = 1/2^k$. Beweis für $LR^{k+1} + LR^{k+1} + RL^k = 0$
  falls erster Zug von L
  - das linke L, dann das oberste R in der Mitte
  - das mittlere L, dann das oberste R links
  - eines der rechten L, dann entsteht nach Induktion ein Spielwert < 0.

  falls erster Zug von R
  - eines der linken R, dann mittleres L
  - eines der mittleren R, dann linkes L
  - das rechte R, dann das linke L

Hausaufgaben

1. Hackenbush (Zahlen): den Wert von $(LR)^k$,

   einer beliebigen Kette $\in \{L, R\}^*$ bestimmen.

   mit Programm Vermutung erzeugen und beweisen.

2. Hackenbush (unscharfe Werte) Position $GL + G$ ist > 0, aber kleiner als jedes $1/2^k = LR^k$

   Programm erweitern.

3. Hackenbush für Bäume (Programm erweitern)

4. Toads and Frogs: einige Werte ausrechnen.

   offene Fragen mit brute force bearbeiten

5 Die Normalform von Spielen

Motivation

- wir hatten Äquivalenz-Relation $\approx$ auf Spielen definiert: $G \approx H \iff$ Status von $(G - H)$ ist Null

  das ist *semantische* Charakterisierung (zweiter gewinnt), erfordert *Auswertung* (Spielbaum-Bewertung)

- Ziel ist jetzt *symbolische* Charakterisierung

  durch Umformungsregeln $\to_U$ von Spielen (Termen) (Weglassen und Ersetzen von Optionen)

- so daß $G \approx H \iff G \to_U^* G' = H' \leftarrow_U^* H$

  semant. Äquivalenz = syntakt. Gleichheit der U-Nf.

- U verwendet doch etwas Semantik ($\leq$ auf Optionen)

Vergleich, Ausblick

- ähnliche Aufgabenstellung in Computeralgebra:

- Äquivalenz rationaler Zahlen $p/q \approx r/s$, $(8/12 = 10/15)$
  semantisch: $p \cdot s = q \cdot r$
  Normalform: p/q mit $\gcd(p,q) = 1$
  wird erreicht durch Division mit gcd
  dann ist semantische Äquivalenz $=$ Identität auf Nf.
- Äquivalenz von algebraischen Ausdrücken
  $1/(\sqrt{2}-1) - (\sqrt{2}+1) = 0$

Unnütze Optionen hinzufügen

- Wdhlg: $G > 0$, falls L gewinnt (egal, wer beginnt)
  $G \| 0$: der erste gewinnt, $G = 0$: der zweite gewinnt
  $G \geq 0$: L gewinnt, wenn R anfängt $(G > 0 \vee G = 0)$
- Def: $G \big|_{\triangleright} 0$: L gewinnt, wenn L anfängt $(G > 0 \vee G \| 0)$
  Def: $G \big|_{\triangleright} H$, falls $(G - H) \big|_{\triangleright} 0$; $\quad G \triangleleft\big| H$, falls $H \big|_{\triangleright} G$
- Satz: für alle $x \in G^L, y \in G^R$ gilt $x \triangleleft\big| G \triangleleft\big| y$.
  Beispiel: $x = 1 \triangleleft\big| G = \{1 \mid 0\} \triangleleft\big| 0$
  Beweis: $(G - x) \to_L (x - x) \approx 0$, $(y - G) = (y + (-G)) \to_L (y + (-y)) \approx 0$
- Satz: wenn $x \triangleleft\big| G$, dann $G' = (G^L \cup \{x\}, G^R) \approx G$.
  Bew: $(G' - G) \to_L (x - G) \triangleleft\big| 0$, $(G' - G) \to_R (y - G) \to_L (y - y) = 0$

Dominierte Optionen weglassen

- wenn $G = (G^L, G^R)$ mit $g_1, g_2 \in G^L$ mit $g_1 \geq g_2$,
  (die Option g_1 dominiert die Option g_2),
  dann $G \approx G' = (G^L \setminus \{g_2\}, G^R)$ $\quad$ (g_2 weg, g_1 bleibt)
- Bsp: $G = \{0, 1 \mid\}$ mit $g_1 = 1 > 0 = g_2$, also $G' = \{1 \mid\}$.
- Beweis: $g_2 \leq g_1 \triangleleft\big| G'$, also $g_2 \triangleleft\big| G'$, g_2 unnütz für G'
- Implementierung (Ansatz): smart constructor (`game`)

```
data Game = Game (S.Set Game) (S.Set Game)
game :: [Game] -> [Game] -> Game
game xs ys = Game (maxima xs) (minima ys)
minima = negate . maxima . map negate
```

Reversible Optionen ersetzen

- wenn $G = (G^L, G^R)$ mit $x \in G^L$ so daß $y \in x^R$ und $y \leq G$,
  dann $G \approx G' = (G^L \setminus \{x\} \cup y^L, G^R)$
- Idee: falls $G \to_L x$, dann $x \to_R y$, ist für R besser als G
- Beispiel, mit: $* = \{0 \mid 0\}, \uparrow = \{0 \mid *\}$ (Name: Up)
  $G = \{\uparrow|\uparrow\}, x = \uparrow, y = *$, aus $y \leq G$ folgt $G \approx \{0 \,|\uparrow\}$, ist Nf.
- Beweis: Strategie für zweiten in $G' - G$
  - für $z \in y^L$: $z \triangleleft \big| y \leq G$, ($z$ als unnütze L-Option in G)
  - $(G' - G) \to_R (G' - x) \to_L (G' - y)$. Jetzt R nicht in $-y$, sondern in G', zu $\to_R (p - y) \geq (p - G) \big| \triangleright 0$
- Implementierung?

Satz über Normalformen

- zu jedem Spiel G gibt es *genau ein* Spiel H mit $G \approx H$ und H hat keine dominierten und reversiblen Optionen.
- Beweisplan (Existenz):
  - wiederholtes Anwenden der Vereinfachungsregeln
  - *Termination*, weil jede Regelanwendung den Ausdruck verkleinert (Optionen entfernt)
- Beweisplan (Eindeutigkeit):
  - *jede* Strategie führt zur gleichen Normalform
- praktisch nützlich:

- Strategie mit geringer *Ableitungskomplexität*
- ... falls alle Optionen schon in Normalform sind

Zahlen

- Def: G heißt *Zahl*, falls $\forall g \in G^L, h \in G^R : g < h$,
  und alle Optionen von G Zahlen sind.
- Bsp: 0 ist Zahl, $1 = \{0 \mid\}$ ist Zahl, $-1 = \ldots$,
  $1/2 = \{0 \mid 1\}$ ist Zahl, $* = \{0 \mid 0\}$ ist keine Zahl.
- Satz: Summe von Zahlen ist Zahl.
- Satz: Jede Zahl ist mit 0 vergleichbar (keine Zahl ist unscharf). Folgerung: Auf Zahlen ist $\leq$ total.
- Folgerung: $\max G^L$ dominiert alle in G^L,
  jede Zahl G hat Darstellung mit $|G^L| \leq 1, |G^R| \leq 1$.
- ... die der Normalform-Algorithmus auch finden sollte

Ganze Zahlen und dyadische Brüche

- Def: für $a \in$: $(a + 1) := \{a \mid\}$
- Satz: diese ganzen Zahlen kann man addieren, subtrahieren
- Def: für $a \in$ ungerade, $b \in$: $a/2^b := \{(a-1)/2^b \mid (a+1)/2^b\}$.
  Anwendung: $3/8 := \{1/4 \mid 1/2\}$, $1/4 := \{0 \mid 1/2\}$, $1/2 := \{0 \mid 1\}$.
- Satz: diese *dyadischen* Zahlen verhalten sich wie erwartet

Switches

- Def: ein *Switch* ist ein Spiel $\{x \mid y\}$ mit Zahlen $x \geq y$.
  Def: $\pm x$ ist das Spiel $\{x \mid -x\}$ (symmetrischer Switch)
- Satz: für alle Zahlen $x \geq y > z$ gilt $z < \{x \mid y\}$
  Beispiel? Beweis?

- Satz: jeder Switch ist die Summe einer Zahl und eines symmetrischen Switches
- exakte Formulierung, Beispiel, Beweis
- (evtl. HA) wie ist der Status einer Summe von Switches?

  wo ist der Gewinnzug?

Die Temperatur eines Spieles

- in einer disjunkten Summe sollte man niemals in einer Komponente ziehen, die eine Zahl ist

  es sei denn, es gibt nur noch solche, aber dann kann man die Summe ausrechnen.
- den Nicht-Zahlen ordnet man eine *Temperatur* zu,

  die die Dringlichkeit ausdrückt (heiß $=$ wichtig)
- die Temperatur bestimmt man durch *Abkühlung*.
- damit lassen sich (auch für Menschen) schwierige Go-Endspiele ausrechnen.

  Elwyn Berlekamp and David Wolfe: *Mathematical Go: Chilling Gets the Last Point* 1994, `https://math.berkeley.edu/~berlek/cgt/go.html`

Hausaufgaben

1. (die einfache erste Aufgabe): Gegeben ist ein Spiel in Normalform. L ist dran. Welche Option wählt er?
2. Normalisierungs-Strategien ausprobieren, Ableitungslängen diskutieren.
3. autotool-Aufgabe entwerfen zu: dominiert, reversibel, Normalform. — Mit sinnvollen Fehlermeldungen, die aber die Lösung nicht verraten.
4. Aaron Siegel: *Combinatorial Games Suite* ausprobieren `http://cgsuite.sourceforge.net/`
5. Def: $\Uparrow = \uparrow + \uparrow$. Normalform?
6. (Ergänzung zu voriger Woche) Hackenbush: $G = (LR)^\omega$, strategischer Beweis für $G + G + G = 2$.

6 Abstrakte Reduktionssysteme (ARS)

Definition, Motivation

- Def: ein ARS $A = (U, \to)$ ist ein Universum U
  und ein System von Relationen $\to_i \subseteq U^2$ für $i \in I$.
  für $I = \{1\}$ oder $I = \{1, 2\}$ Notation $(U, \to)$, $(U, \to_1, \to_2)$
- das abstrahiert von *konkreten* Reduktionssystemen, z.B.
  - U Spielpositionen, $\to_1, \to_2$: Züge für Spieler 1,2
  - U Terme, $\to_i$ durch Ersetzungs/Vereinfachungs-regeln
  - U Speicherbelegungen, $\to$ durch Programm-Befehle
- wichtige Eigenschaften von ARS sind
  - Termination (keine unendlichen Ketten)
  - Konfluenz (Zusammenführbarkeit divergenter Ketten)

Literatur, Quellen

- Bücher
  - (Lehrbuch) Franz Baader, Tobias Nipkow: *Term Rewriting and All That*, Cambridge Univ. Press 1998, Kapitel 2
  - (Monographie) TeReSe, *Term Rewriting Systems*, Cambridge Univ. Press 2003,
- Forschungsgruppen (Termersetzung und Anwendungen)
  - Jan Willem Klop, VU Amsterdam `https://www.cs.vu.nl/~tcs/`
  - Aart Middeldorp, U Innsbruck `http://cl-informatik.uibk.ac.at/`

Wiederholung: Relationen

- (zweistellige) Relation $R \subseteq U^2$, gerichteter Graph
- Operationen:
  - Mengen-Operationen (Vereinigung, Durchschnitt)

  - Produkt $R \circ S$, Potenz R^k, inverse Relation R^{-1}
  - symmetrische Hülle $R^1 \cup R^{-1}$
  - R^+ transitive Hülle $= \bigcup_{k>0} R^k$
  - $R^=$ reflexive Hülle $= R^0 \cup R^1$
  - R^* transitive und reflexive Hülle $= \bigcup_{k\geq 0} R^k$
- Eigenschaften:
  - symmetrisch, reflexix, transitiv; linear (total),
  - terminierend (wohlfundiert)
- alle zweist. Relationen auf U bilden Halbring (mit $\cup, \circ$)

Beispiele

- $S = \{(n, n+1) \mid n \in\}$ die Nachfolger-Relation auf
- $P = S^-$ die Vorgänger-Relation
- Ü: beschreibe S^+, S^*, $(S \cup P)^+$
- die (umgekehrte) Teilbarkeits-Relation
  $x \to y$ falls $\exists z : x = y \cdot z$
- auf $U =^2$ die Relationen $(x_1, x_2) \to_i (y_1, y_2)$ falls $x_i > y_i$
  Ü: beschreibe $\to_1 \cup \to_2,\ \to_1 \cap \to_2,\ \to_1 \circ \to_2$

Bezeichnungen (I)

- alles für $(U, \to)$
- $x \in U$ ist *reduzierbar*, wenn $\exists y : x \to y$
- $x \in U$ ist *Normalform* (irreducible), wenn $\neg\exists y : x \to y$
- y ist *Normalform von* x, falls $x \to^* y$ und y ist Normalform
- x und y sind *zusammenführbar* (joinable), wenn $\exists z : x \to^* z \leftarrow^* y$. Notation dafür: $x \downarrow y$
- in ARS mit mehreren Relationen $(R, S, \dots)$
  sind die Bezeichnungen:
  R-Normalform, S-zusammenführbar usw.

Bezeichnungen (Eigenschaften v. Elementen)

- für eine Relation $\to$ auf U: das Element $x \in U$ heißt ...
  - *terminierend*, $\mathsf{SN}(x)$: es gibt keine unendliche Kette, die bei x beginnt: $x = x_0 \to x_1 \to \ldots$
  - *normalisierend*, $\mathsf{WN}(x)$: x hat Normalform
  - hat *Diamant-Eigenschaft*, $\mathsf{DP}(x)$: wenn $\forall y_1, y_2 : y_1 \leftarrow x \to y_2 : \exists z : y_1 \to z \leftarrow y_2$
  - *lokal konfluent*, $\mathsf{WCR}(x)$: wenn $\forall y_1, y_2 : y_1 \leftarrow x \to y_2 : y_1 \downarrow y_2$
  - *konfluent*, $\mathsf{CR}(x)$: wenn $\forall y_1, y_2 : y_1 \leftarrow^* x \to^* y_2 : y_1 \downarrow y_2$
- Beispiele, trennende Beispiele
  - normalisierend, aber nicht terminierend
  - lokal konfluent, aber nicht konfluent

Bezeichnungen (Eigenschaften v. Relationen)

- ist *terminierend* (SN): $\forall x \in U : \mathsf{SN}(x)$
- ist *normalisierend* (WN): $\forall x \in U : \mathsf{WN}(x)$
- hat *eindeutige Normalformen* (UN): für alle Normalformen y_1, y_2 gilt: $y_1 (\leftarrow \cup \to)^* y_2 \implies y_1 = y_2$
- hat *eindeutige NF für Reduktion* (UNR): für alle x, alle Normalformen y_1, y_2 gilt: $y_1 \leftarrow^* x \to^* y_2 \implies y_1 = y_2$
- DP: $\forall x \in U : \mathsf{DP}(x)$, äquiv.: $\leftarrow \circ \to \subseteq \to \circ \leftarrow$
- *lokal konfluent* (WCR): $\forall x \in U : \mathsf{WCR}(x)$
  äquivalente Spezifikation: $\leftarrow \circ \to \subseteq \to^* \circ \leftarrow^*$
- *konfluent* (CR): jedes $x \in U$ ist konfluent
  äquivalente Spezifikation: $\leftarrow^* \circ \to^* \subseteq \to^* \circ \leftarrow^*$
- *Church-Rosser* (CR), wenn $(\leftarrow \cup \to)^* \subseteq \to^* \circ \leftarrow^*$
- *konvergent*, wenn $\to$ terminierend und konfluent ist

Hausaufgaben

1. Toads and Frogs: UN, CR,... der Relationen
   - $\to_{FM}$ Frosch schieben, $\to_{FJ}$ Frosch springen

- $\to_R \;=\; \to_{FM} \cup \to_{FJ}$ Optionen für Spieler R
- $\to \;=\; \to_R \cup \to_L$

2. entsprechend für Hackenbush (ohne/mit grün)
3. Vereinfachung von Spielen: Eigenschaften der Relationen
   - $\to_{DL}$ dominierte Option entfernen für L,
   - $\to_D \;=\; \to_{DL} \cup \to_{DR}$
   - $\to_{VL}$ reversible Option überspringen für L,
   - $\to_V \;=\; \to_{VL} \cup \to_{VR}, \to \;=\; \to_D \cup \to_V$

7 Beziehungen zw. ARS-Eigenschaften

Motivation, Plan

- eine Anwendung ist:
  - um $x \leftrightarrow^* y$ zu entscheiden:
  - bestimme $x \to^* x' \in$, $y \to^* y' \in$, teste $x' = y'$

  funktioniert, falls $\mathsf{UNC}(\to)$ und $\mathsf{SN}(\to)$
- Vorgehen:
  - WCR und SN abhängig vom konkreten $\to$ (später)
  - $\mathsf{WCR} \wedge \mathsf{SN} \Rightarrow \mathsf{CR}$ (Lemma von Newman)
  - $\mathsf{CR} \Rightarrow \mathsf{UNC}$
- CR ist auch bei $\neg\,\mathsf{SN}$ interessant, Bsp: Lambda-Kalkül, das ist auch der historische Hintergrund

Normalformen

- Satz: $\forall R : \mathsf{UNC}(R) \Rightarrow \mathsf{UNR}(R)$
- Beweis (einfach):
  Def. $\mathsf{UNC}(\to)$: $\ni y_1(\leftarrow \cup \to)^* y_2 \in\!\Rightarrow y_1 = y_2$
  Def. $\mathsf{UNR}(\to)$: $\ni y_1(\leftarrow^* \cdot \to^*) y_2 \in\!\Rightarrow y_1 = y_2$
  es gilt $(\leftarrow^* \cdot \to^*) \subseteq (\leftarrow \cup \to)^*$

- es gilt nicht $\forall R : \mathsf{UNR}(R) \Rightarrow \mathsf{UNC}(R)$, Gegenbeispiel: Hausaufgabe (Universum mit 5 Elementen)

CR und Konfluenz

- Def: CR: $(\leftarrow \cup \rightarrow)^* \subseteq (\rightarrow^* \cdot \leftarrow^*)$
- Def: Konfluenz: $(\leftarrow^* \cdot \rightarrow^*) \subseteq (\rightarrow^* \cdot \leftarrow^*)$
- Satz: diese Eigenschaften stimmen überein
  (deswegen nur eine gemeinsame Bezeichnung)
- Beweis: CR $\Rightarrow$ Konfluenz wie eben (einfache Inklusion)
- B: Konfluenz $\Rightarrow$ CR: Ind. nach k in $\rightarrow^* \cdot(\leftarrow^* \cdot \rightarrow^*)^k \cdot \leftarrow^*$:
  einen Berg $(\leftarrow^* \cdot \rightarrow^*)$ durch ein Tal $(\rightarrow^* \cdot \leftarrow^*)$ ersetzen:
  $\rightarrow^* \cdot(\leftarrow^* \cdot \rightarrow^*)^{k+1} \cdot \leftarrow^*$
  $= \quad \rightarrow^* \cdot(\leftarrow^* \cdot \rightarrow^*) \cdot (\leftarrow^* \cdot \rightarrow^*)^k \cdot \leftarrow^*$
  $\subseteq \quad \rightarrow^* \cdot(\rightarrow^* \cdot \leftarrow^*) \cdot (\leftarrow^* \cdot \rightarrow^*)^k \cdot \leftarrow^*$
  $\subseteq \quad \rightarrow^* \cdot(\leftarrow^* \cdot \rightarrow^*)^k \cdot \leftarrow^*$

Konfluenz und Normalformen

- Satz: $\forall R : \mathsf{CR}(R) \Rightarrow \mathsf{UNC}(R)$
- Gegenbeispiel für $\Leftarrow$ (3 Elemente, *keine* Normalformen)
- Beweis: falls $\ni y_1(\leftarrow \cup \rightarrow)^* y_2 \in$
  dann wegen CR: $\exists z, i, j : y_1 \rightarrow^i z \leftarrow^j y_2$
  aus $y_1 \in$ folgt $i = 0$, aus $y_2 \in$ folgt $j = 0$,
  also $y_1 \rightarrow^0 \cdot \leftarrow^0 y_2$, also $y_1 = y_2$

Lokale Konfluenz und Konfluenz

- Lemma von Newman: $\mathsf{SN} \wedge \mathsf{WCR} \implies \mathsf{CR}$
- Beweis (nach Henk Barendregt, zitiert in TeReSe)
  - wegen SN hat jedes $a \in U$ wenigstens eine Nf.

- wir zeigen: ... genau eine Nf. Daraus folgt CR.
  - Bezeichnung: a mehrdeutig ($M(a)$): hat ≥ 2 Nf.
  - zeigen: wenn $M(a)$, dann exist b mit $a \to b$ und $M(b)$.
    daraus folgt $\neg\exists a : M(a)$, sonst Widerspruch zu SN
  - Falls $M(a)$, dann $a \to b_1 \to^* n_1$ und $a \to b_2 \to^* n_2$ mit $n_1 \neq n_2$ Nf.
    WCR: existiert c mit $b_1 \to^* c \leftarrow^* b_2$ und $c \to^* n$ Nf.
    Dann $n_1 \neq n$ oder $n_2 \neq n$, also $M(b_1)$ oder $M(b_2)$.

Konfluenz ohne Termination

- Anwendung: $\mathsf{CR}(\to_\beta)$ für Lambda-Kalkül,

  Beispiel: $(\lambda y.yy)(I(\lambda z.zz))$ für $I = \lambda x.x$

- Satz: Falls $\to_1 \subseteq \to_2 \subseteq \to_1^*$ und $\mathsf{DP}(\to_2)$, dann $\mathsf{CR}(\to_1)$.

- Beweis (TeReSe Ex. 1.3.1)

  $\mathsf{DP}(\to_2) \Rightarrow \mathsf{DP}(\to_2^*) = \mathsf{CR}(\to_2)$, $\to_1^* = \to_2^*$, $\mathsf{CR}(\to_1)$.

- Anwendung: $s \to_2 t$ durch vollständige Reduktion einer Menge von markierten Redexes in s

  Markierungen werden verbraucht oder kopiert, neue entstehende Redexe *nicht* markiert

  deswegen ist $\to_2$ wohldefiniert (es gilt $\mathsf{SN}(\to_2)$)

- Bemerkung/Wiederholung: $\mathsf{WCR}(\to) \not\Rightarrow \mathsf{CR}(\to)$

Hausaufgaben

empfohlen: beide (b), 4, 5

1. TeReSe Ex. 1.3.2

   Def: $(\to_1, \to_2)$ kommutieren schwach: $\leftarrow_1 \cdot \to_2 \subseteq \to_2^* \cdot \leftarrow_1^*$

   Def: $(\to_1, \to_2)$ kommutieren: $\leftarrow_1^* \cdot \to_2^* \subseteq \to_2^* \cdot \leftarrow_1^*$

   (a) Satz: wenn $(\to_1, \to_2)$ schwach kommutieren und $\mathsf{SN}(\to_1 \cup \to_2)$, dann kommutieren $(\to_1, \to_2)$.

   (b) Gegenbeispiel für: ... und $\mathsf{SN}(\to_1) \wedge \mathsf{SN}(\to_2)$, dann ...

2. TeReSe Ex. 1.3.15, Geser 1990, di Cosmo, Piperno 1995

   (a) Wenn $\leftarrow_1 \cdot \rightarrow_2 \subseteq \rightarrow_2^+ \cdot \leftarrow_1^*$ und $\mathsf{SN}(\rightarrow_2)$, dann kommutieren $(\rightarrow_1, \rightarrow_2)$.

   (b) zeigen, daß $\mathsf{SN}(\rightarrow_2)$ notwendig ist.

3. TeReSe Ex. 1.3.3, Rosen 1973.

   Def (sub-commutative) $\mathsf{CR}^{\leq 1}(\rightarrow) :=\leftarrow \cdot \rightarrow \subseteq \rightarrow^{0,1} \cdot \leftarrow^{0,1}$

   Satz: aus $\rightarrow_1^* = \rightarrow_2^*$ und $\mathsf{CR}^{\leq 1}(\rightarrow_1)$ folgt $\mathsf{CR}(\rightarrow_2)$.

4. TeReSe Ex 1.3.22.ii

   ein R mit $\mathsf{WCR}(R) \wedge \mathsf{UNR}(R) \wedge \neg\, \mathsf{UNC}(R)$

5. alle Gegenbeispiele zu vorigen Aussagen könnte man

   - durch passende autotool-Aufgaben übeprüfen oder
   - durch passenden Carpa-Aufruf bestimmen `https://www.win.tue.nl/~hzantema/carpa.html`

   —aber nur, falls es Gegenbeispiele mit endlichem Universum gibt. Das ist nicht immer der Fall.

   Geben Sie eine Eigenschaft vom Typ `Prop` an `https://autotool.imn.htwk-leipzig.de/docs/autotool-collection-1.3/Rewriting-Abstract-Data.html#t:Prop`, die

   - erfüllbar ist,
   - aber nicht endlich erfüllbar (d.h., mit endlichen Universum erfüllbar)

6. Implementierung in Carpa (2018?) mit `https://gitlab.imn.htwk-leipzig.de/autotool/all0/-/blob/master/collection/src/Rewriting/Abstract/Solve.hs` (2015?) vergleichen.

   Gemeinsamkeit: aussagenlogische Codierung. Unterschied: SAT/BDD. Auswirkungen?

8 Terme, Ersetzungs-Systeme (Grundl.)

Termbäume, Positionen

- Signatur Σ: Menge von Funktionssymbolen mit Stelligkeit, Bsp. $\{(s,1),(z,0),(p,2)\}$,
- (Σ) ist die kleinste Menge M mit ...
- (t): Menge der Positionen in $t \in (\Sigma)$: ...
- $t(p)$ Symbol an Position p in t
- $t\,|_p$ Teilterm an Position p in t (alternative Notation: $t[p]$)

Beispiele:

- Term $t = p(z, s(s(z)))$,
- Positionen $(t) = \{\epsilon, 0, 1, 10, 100\}$,
- Symbol $t(1) = s$, Teilterm $t[1] = s(s(z))$,

Baum-Bereiche

- Def: Menge $P \subseteq^*$ heißt Baum-Bereich (*tree domain*), wenn

  (... es einen Term t gibt, so daß $P = (t)$ — das soll aber ohne Benutzung des Term-Begriffs definiert werden)

  - $\epsilon \in P$
  - P abgeschlossen unter ...

  definiere und benutze Relationen $<_u$ (up) und $<_l$ (left).

- Def: ein Term t über Signatur Σ *ist* Abbildung von einem Baumbereich P nach Σ

  oder: partielle Abbildung $t :^* \hookrightarrow \Sigma$ mit $\mathsf{dom}(t)$ Baumbereich

Einsetzen an Position

- $t[p := s]$ in t an Position p den Term s einsetzen
- Übung: exakte Definition, für Term als Abbildung von Baumbereich

  dazu ist $t' :^* \hookrightarrow \Sigma$ zu implementieren

  für $t' = t[p := s]$ und wenn $t, s :^* \hookrightarrow \Sigma$ gegeben
- Übung: ergänzen: $t[p_1 := s_1][p_2 := s_2] =$
  - wenn ..., dann $= t[p_2 := s_2][p_1 := s_1]$
  - sonst = ... (2 Fälle)

Variablen, Substitutionen

- (Σ, V): Menge der Terme mit Symbolen aus Σ und Variablen aus V.
- Vereinbarung: Variablen am Ende des Alphabets $(x, y, \ldots)$, Funktionssymbole am Anfang $(f, g, a, b, \ldots)$
- Menge der in einem Term t vorkommende Variablen: (t)
- Substitution σ ist Abbildung $V \to (\Sigma, W)$
- eine Substitution σ angewendet auf einen Term t erzeugt Term $t\sigma$
  Beispiel $t = f(x)$, $(t) = \{x\}$, $\sigma : x \mapsto p(0,0)$, $t\sigma = f(p(0,0))$.

Regeln

- Regel (l, r), Schreibweise $(l \to r)$, mit $l, r \in (\Sigma, V)$
- Regel $(l \to r)$ an Position p in $t \in (\Sigma)$ anwenden, um s zu erhalten, Schreibweise $t \to_{(l \to r),p} s$
  $\exists \sigma : V \to (\Sigma) : t[p] = l\sigma \wedge t[p := r\sigma] = s$
- Beispiel: Regel $(l, r) = (f(x), g(x, x))$, Term $t = h(1, f(f(2)))$.
  Position $p = [1] \in (t)$, Teilterm $t[p] = f(f(2))$,
  Substitution $\sigma : x \mapsto f(2)$ mit $t[p] = l\sigma$,
  auf r anwenden: $r\sigma = g(f(2), f(2))$,
  in t einsetzen: $s = t[p := r\sigma] = h(1, r\sigma) = h(1, g(f(2), f(2)))$.

Regelsysteme

- R eine Menge von Regeln (TRS, term rewriting system)
  definiert Relation $\to_R$ auf (Σ)
  (einmalige Anwendung irgendeiner Regel irgendwo):
  $\to_R := \{(t, s) \mid \exists (l, r) \in R, p \in (t) : t \to_{(l,r),p} s\}$
- jedes solche $\to_R$ definiert ein ARS
- ist nichtdeterministisches Berechnungsmodell
  Resultat $= (\to_R)$-Normalform (kürzer: R-Normalform)

- Ausdrucksstärke stimmt mit Turing-Maschinen überein
  - Simulation einer TM M durch $\to_R^+$ für TRS R
  - Simulation von $\to_R$ durch TM

Termersetzung/Anwendungen

Termersetzung ist

- Grundlage für funktionale Programmierung
  (Auswertung, Kompilation, Verifikation)
- Grundlage für Symbolisches Rechnen
- (Grundlage für XML-Transformationen mit XSLT, ...)

Für Anwendungen wichtig sind die Eigenschaften

- Termination (SN) (keine unendlich langen Rechnungen)
- Konfluenz (CR,UNR) (eindeutige Ergebnisse)
- Ableitungskomplexität (Länge von Rechnungen als Funktion der Startgröße)

das ist aber alles unentscheidbar (wg. Turing-Vollst.)

Term-Ersetzung und Computeralgebra

- Regeln (Term-Ersetzungs-System) für das Differenzieren:

$$\begin{aligned} D_x(x) = 1, \text{ wenn } x \notin A : D_x(A) &= 0 \\ D_x(A + B) &= D_x(A) + D_x(B) \\ D_x(A \cdot B) = \ldots, D_x(A/B) &= \ldots \\ D_x(\log x) = 1/x, D_x(\sin x) &= \cos x, \ldots \end{aligned}$$

  Dazu braucht man aber noch Vereinfachungsregeln.
- Wie drückt man die Kettenregel $D_x(f(g(x)) = \ldots$ aus?
  Hier sind f und g Variablen, aber zweiter Ordnung (bezeichnen Funktionen).
  allgemein: Higher Order Rewriting (HOR) als Verallgemeinerung von TRS und Lambda-Kalkül

Wort- und Term-Ersetzung

- Wort (Folge) als Term (Baum, Pfad) auffassen
  Beispiel: $abaab$ bedeutet $a(b(a(a(b(\epsilon)))))$.
- Wortersetzungssystem (SRS):
  Menge von Regeln, Regel: Paar von Wörtern
  $ab \to ba$ bedeutet $a(b(x)) \to b(a(x))$
- Alle Aussagen über TRS gelten auch für SRS,
- SRS (als Berechnungsmodell) sind Turing-vollständig, denn eine TM ist (im wesentlichen) ein SRS
- für viele Anwendungen sind TRS natürlich (algebraische Beschreibung, Terme mit Symbolen der Stelligket > 1)
- zahlreiche Methoden für TRS wurden zuerst für SRS erfunden und untersucht (z.B. für Termination)

Hausaufgaben zu TRS, SRS

1. ergänzen: $t[p_1 := s_1][p_2 := s_2] = \ldots$
2. Normalformen, Termination, Konfluenz des TRS

$$\begin{aligned} \neg(1) &\to 0, \quad \neg(\neg(x)) \to x \\ 1 \wedge x &\to x, \quad 0 \wedge x \to 0 \\ x \vee y &\to \neg(\neg x \wedge \neg y) \end{aligned}$$

   über Signatur $\{(0,0),(1,0),(\neg,1),(\vee,2),(\wedge,2)\}$

   jeweils Beweis (soweit jetzt möglich, später systematisch) oder Gegenbeispiel (mit Reparaturvorschlag)
3. Die Zug-Relation in Hackenbush auf höchstens binären Bäumen als TRS darstellen. (Signatur, Regeln f. Blau)
4. (auf Papier) unendliche Ableitungen finden für
   (a) $\{f(0,y) \to f(y,S(0)), f(S(x),y) \to f(x,S(y))\}$
   (b) $\{f(0,S(x),y) \to g(f(0,x,S(y)), f(x,y,S(S(0))))$

(c) $A(A(F,x),y) \to A(A(x,A(F,y)),F)$

5. (experimentell) unendliche Ableitungen finden für:
   (a) $\{1000 \to 0001110\}$, allgemein $10^k \to 0^k1^k0$
   (b) $\{0000 \to 0101, 1001 \to 0100\}$
   (c) $\{0000 \to 0111, 1001 \to 0010\}$
   (d) $\{0000 \to 0111, 1011 \to 0110\}$
6. nachrechnen (Abschnitt 3.1, Alg. 3.2, Ex. 3.3: auf Zahlenfolgen, diese repräsentieren Terme) und beweisen:
   Ikebuchi, Nakano: *On Properties of B terms*, 2020 `https://doi.org/10.23638/LMCS-16(2:8)2020`

9 Gleichungsdefinierte Strukturen

Gleichungssysteme (Syntax)

- Def: Gleichungssystem E über Signatur Σ ist Menge von Paaren $(l,r) \in (\Sigma, V)^2$, geschrieben $l \approx r$
- Def: die Relation $\approx_E$ ist $(\to_E \cup \to_E^-)^*$. (alternativ: $\leftrightarrow_E^*$)
- Bsp: $E_1 = \{f^5(a) \approx a, f^3(a) \approx a\}$. Zeige $f(a) \approx_{E_1} a$.
  Bsp: $E_2 = \{f(f(x)) \approx g(x)\}$. Zeige $f(g(x)) \approx_{E_2} g(f(x))$.
  Bsp: Die Relation $\approx_{D_i}$ ist entscheidbar für
  - $D_1 = \{f(f(x,y),z) \approx f(x,f(y,z))\}$
  - $D_2 = \{f(f(x,y),z) \approx f(x,f(y,z)), f(x,y) \approx f(y,x)\}$

  $\approx_C$ ist *nicht entscheidbar* für $C = \{A(A(K,x),y) \approx x,\ A(A(A(S,x),y),z) \approx A(A(x,z),A(y,z))\}$

Algebren (Semantik)

- Def. eine *Algebra* A zur Signatur Σ ist:
  eine Menge (domain) D_A und
  für jedes k-stellige c aus Σ eine Funktion $[c]_A : D^k \to D$.

- Bsp. $\Sigma = \{P/2, S/1, Z/0\}, D_A =, [P]_A(x, y) = x + y, [S]_A(x) = x + 1, [Z]_A = 0.$
- Def: Wert eines Terms $t \in (\Sigma, V)$ in A

  unter Belegung $\alpha : (t) \to D$, Notation $[t, \alpha]_A$

  Bsp. $[P(S(Z), a), \{(a, 5)\}]_A = 6$
- beachte verschiedene Wortbedeutungen von „Algebra“:
  - “high school algebra”: Umformen von Ausdrücken.
  - lineare Algebra: untersucht Vektoren, Matrizen, ...
  - abstrakte Algebra: allgemein Mengen mit Operationen.

Modelle

- Def. eine Gleichung $l \approx r$ ist *gültig* in einer Σ-Algebra $A = (D_A, [\cdot]_A)$, geschrieben $A \models l \approx r$,

  falls für jede Belegung $\alpha : (l) \cup (r) \mapsto D_A$ gilt: $[l, \alpha]_A = [r, \alpha]_A$
- Def. eine Σ-Algebra $A = (D_A, [\cdot]_A)$ heißt *Modell*

  für ein Gls. E über Σ, wenn $\forall(l \approx r) \in E : A \models l \approx r$.
- Bsp: $D =, [P](x, y) = x+y, [S](x) = x+1, [Z] = 0$ ist Modell für $E = \{P(Z, y) \approx y, P(S(x), y) \approx S(P(x, y))\}$.

  Übung: gibt es andere Modelle dafür?

  (über ? Ja: $[P](x, y) = y, [S](x) = x, [Z] = 0$. über anderen Bereichen, z.B.: , Wörter, endlicher Bereich?)

Syntaktische u. semantische Äq. von Termen

- für Gleichungssystem E über Σ

  Bsp. $E = \{P(Z, y) \approx y, P(S(x), y) \approx S(P(x, y))\}$.
- bereits definiert: *syntaktische Äquivalenz*: $\approx_E$

  $(\leftarrow_E \cup \rightarrow_E)^*$, alternative Schreibweise: $E \vdash s \approx t$
- Def. $E \models s \approx t$: für jedes Modell A von E gilt $A \models s \approx t$.

  Bsp: $E \models P(S(S(Z)), y) \approx S(S(y))$.

  Vorsicht: $E \not\models P(x, y) \approx P(y, x)$, dazu später mehr

- Def: *semantische Äquivalenz*: $E \models s \approx t$
- Satz (Birkhoff): $E \vdash s \approx t \iff E \models s \approx t$. Anwend.:
  - syntaktische Beweise für Aussagen in (allen) Modellen
  - semantische Widerlegung von $s \leftrightarrow_E^* t$ (ein Nicht-Mod.)

Gleichungstheoreme und induktive Th.

- Bsp. $E = \{P(Z, y) \approx y, P(S(x), y) \approx S(P(x, y))\}$.
  Gilt $E \models P(x, y) \approx P(y, x)$?
- Nein. Betrachte Modell A für E über $\mathbb{N}$, $[Z]_A = 0$,

$$\begin{aligned} [S]_A(x) &= \text{if } 2|x \text{ then } 2 + x \text{ else } 1 \\ [P]_A(x, y) &= \text{if } 2|x \text{ then } x + y \text{ else } 1 \end{aligned}$$

  $A \not\models P(x, y) \approx P(y, x)$, betrachte $\alpha = \{(x, 2), (y, 1)\}$.
  Nach Satz von Birkhoff folgt $P(x, y) \not\approx_E P(y, x)$.
- Beachte: $\neg\exists t \in (\Sigma, \emptyset) : [t]_A = 1$.
  Für alle $s, t \in (\Sigma, \emptyset)$ gilt $P(s, t) \approx_E P(t, s)$.
  Sprechweise: $P(x, y) \approx P(x, y)$ ist ein *induktives Theorem* in E, aber kein Gleichungs-Theorem.

Aufgaben

1. Für $E = \{aba(x) \approx x\}$: gilt $abbbaa(x) \approx_E bba(x)$?
2. $F = \{f(x, f(y, z)) \approx f(f(x, y), z), f(e, x) \approx x, f(i(x), x) \approx e\}$. Zeige $f(x, e) \approx_F x$.
3. $G = \{f(x, f(y, z)) \approx f(f(x, y), z), f(f(x, y), x) \approx x\}$. Zeige $f(x, x) \approx_G x$ und $f(f(x, y), z) \approx_G f(x, z)$
4. $H = \{f(x, f(y, z)) \approx f(f(x, y), z), f(e, x) \approx x, f(x, i(x)) \approx e\}$. Zeige $f(x, e) \not\approx_H x$.

Jetzt ad-hoc, später teilw. systematisch/automatisiert
Quellen: TeReSe Kapitel 7.1; Baader/Nipkow Kapitel 3, 4.

10 Termination

Motivation

- gibt es unendliche Ableitungen für . . . ?
  - $a \to b$, $ba \to ab$, $ba \to aab$, $aa \to aba$, $100 \to 00110$
  - $A(A(D, x), y) \to A(x, A(x, y))$
  - Regeln zum symbolischen Differenzieren
  - zum Auswerten o. Vereinfachen Boolescher Ausdrücke
  - beliebiges Programm (TM (SRS), funktional (TRS))
- Anwendung (später): Vervollständigungs-Algorithmus

  (zum Entscheiden der syntaktischen Kongruenz $\approx_E$)

  Invariante: das (aktuelle) System terminiert (Ziel: . . . und ist konfluent)

Automatische Terminations-Analyse

- die Menge $\{R \mid \mathsf{SN}(\to_R)\}$ ist nicht entscheidbar,

  (Beweis: Reduktion vom Halteproblem: Simulation der Rechnung einer TM, mit zusätzlicher Betrachtung ungültiger Konfigurationskodierungen)
- wegen der großen praktischen Bedeutung in der Software-Verifikation wünscht man *korrekte partielle* Verfahren: diese antworten auf Frage $\mathsf{SN}(R)$

  mit: Ja, Nein (beides muß stimmen) oder Weißnicht.
- Wettbewerb für Implementierungen solcher Verfahren: Intl. Termination Competition (seit 2003),

  mit Termination Problem Data Base (TPDB) `http://termination-portal.org/`

Historische Quellen zur Termination

- Alan Turing: *Checking a large routine*, 1949, `http://www.turingarchive.org/browse.php/B/8`
- Zohar Manna and Steven Ness: *On the termination of Markov algorithms*, 1970. Dallas Lankford: *On Proving term rewriting systems are Noetherian*, 1979.

siehe http://perso.ens-lyon.fr/pierre.lescanne/not_accessible.html#termination

- Nachum Dershowitz: *33 Examples of Termination*, 1995. http://www.math.tau.ac.il/~nachumd/papers/printemp-print.pdf
- Intl. Workshops on Termination (seit 1993) http://termination-portal.org/wiki/WST

Beweisverfahren für Termination

- syntaktisch: Bsp. für $\{ba \to ab\}$:
  aus $u \to v$ folgt $|u| = |v|$ und $u >_{\text{lex}} v$ für $b > a$
  Vorsicht: für $\{ba \to aabb\}$ gilt auch $u >_{\text{lex}} v$
- semantisch: Bsp. für $\{ba \to ab\}$:
  Anzahl der Inversionen $f(w) := |\{(i,j) \mid i < j \wedge w_i > w_j\}|$
  ist eine *Schrittfunktion* (aus $u \to v$ folgt $f(u) > f(v)$)
- Transformation: Bsp. für $\{aa \to aba\}$:
  jeder Buchstabe markiert mit seinem rechten Nachbarn
  $\{a_a a_a \to a_b b_a a_a, a_a a_b \to a_b b_a a_b\}$, Anzahl d. a_a ist Schrittfkt.
- Modularität: $\mathsf{SN}(R_1) \wedge \mathsf{SN}(R_2) \wedge$ weitere Bedingungen $\implies$ $\mathsf{SN}(R_1 \cup R_2)$.
  Vorsicht: $R_1 = \{a \to b\}, R_2 = \{b \to a\}$

Wohlfundierte monotone Algebren

- Beispiel:
  - Regel $ba \to aab$, Abl. $bba \to baab \to aabab \to aaaabb$
  - Interpretation in : $[a](x) = x + 1, [b](x) = 3x, [\epsilon] = 0$
  - $[bba] = 9 > [baab] = 6 > [aabab] = \ldots > [aaaabb] = \ldots$
- Def: $(D, >, [\cdot])$ ist *wohlfundierte monotone* Algebra
  - wohlfundiert: $\mathsf{SN}(>)$, keine unendl. absteigende Kette
  - monoton: $\forall c \in \Sigma, x \in D, y \in D : x > y \Rightarrow [c](x) > [c](y)$
    allgemein: Monotonie in jedem Argument einzeln

- Def: monotone Algebra ist *kompatibel* mit R:
  - Kompatibilität: $\forall (l, r) \in R, x \in D : [l](x) > [r](x)$

  wg. Monotonie folgt daraus $u \to_R v \Rightarrow [u] > [v]$
- Satz: $\mathsf{SN}(\to_R) \iff \exists$ wfmA, die mit R kompatibel ist.

Wohlfundierte monotone Algebren (II)

- Satz: $\mathsf{SN}(\to_R) \iff \exists$ wfmA, die mit R kompatibel ist.
- Beweis ($\mathsf{SN}(\to_R) \Leftarrow \ldots$): indirekt:
  falls $x_0 \to_R x_1 \to_R x_2 \to_R \ldots$ unendlich,
  dann $[x_0]_A > [x_1]_A > [x_2]_A > \ldots$ unendlich,
  Widerspruch zu $\mathsf{SN}(>)$.
- Beweis ($\mathsf{SN}(\to_R) \Rightarrow \ldots$) zu konstruieren ist die wfmA:
  - Universum: (Σ), Interpretation: Symbol durch sich
  - $u > v :\Leftrightarrow f(u) > f(v)$ mit $f(t) := \max\{k \mid \exists t' : t \to_R^k t'\}$, ist erlaubt (max. über endliche Menge) wegen $\mathsf{SN}(\to_R)$.
- Das hilft aber nicht bei der Suche nach kompatibler wfmA
- Q: warum ist dieses f keine wfmA $(, >, [\cdot])$?
  A: es ist im allgemeinen keine Algebra (kein `fold`)

Beispiel wfmA

Signatur $\Sigma = \{P/2, S/1, Z/0\}$.
Regelmenge $R = \{P(Z, y) \to y, P(S(x), y) \to S(P(x, y))$

verschiedene Σ-Algebren über , überprüfe Eigenschaften.
Welche liefert einen Terminationsbeweis für R?

- $[P](x, y) = x + y, [S](x) = x + 1, [Z] = 0$
- $[P](x, y) = 2x, [S](x) = x + 1, [Z] = 1$
- $[P](x, y) = 2x + y + 1, [S](x) = x + 1, [Z] = 0$

Aufgabe: finde kompatible wfmA für
$R \cup \{M(Z, y) \to Z, M(S(x), y) \to P(M(x, y), y)\}$

Aufgabe: ... für $A(A(D,x),y) \to A(x,A(x,y))$

Merke: kompatibel: links $>$ rechts, Modell: links $=$ rechts,

Systematik wfmA

- bisher: Interpretation der Funktionssymbole durch multilineare Funktionen über . — Erweiterungen:
  - Polynome über
  - lineare Funktionen (Matrizen) über ^{d}
  - ... über anderen Halbringen, wie $(\{-\infty\}\cup, \max, +)$
- für jedes solche Klasse von Algebren möchte man lösen:
  - das *Verifikations-Problem*:
    gegeben: (endliche Beschreibung einer) Algebra A,
    gesucht: A ist monoton? A ist kompatibel mit R?
  - das *Synthese-Problem*:
    gegeben R, gesucht R-kompatible wfmA A

Matrix-Interpretationen (für SRS) als wfmA

- Universum: ^{d} (Spaltenvektoren) mit $x_d \geq 1$
  Ordnung: $x > y \iff x_1 > y_1 \wedge \bigwedge_{i>1} x_i \geq y_i$
  Interpretation $[c](x) = M_c \cdot x$ für Matrix $M_c \in^{d\times d}$
- $[a](x) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \cdot x, \quad [b](x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot x$
- das Verifikation-Problem: ist Algebra? wohlfundiert? monoton? kompatibel mit $aa \to aba$?
  - für dieses Beispiel,
  - allgemein (wie beweist man $\mathsf{SN}(a^2b^2 \to b^3a^3)$)?

Matrix-Interpretationen (für TRS) als wfmA

- Universum (Spalten)Vektoren ^{d}
- Ordnung: $x > y \iff x_1 > y_1 \wedge \bigwedge_{i>1} x_i \geq y_i$
- Interpretation jedes k-stelligen Symbols f
  durch k-stellige multi-lineare Funktion
  $[f](x_1, \ldots, x_k) = M_1 \cdot x_1 + \cdots + M_k x_k + m_0$
  mit Vektor m_0, Matrizen $M_1, \ldots, M_k$,
  angewendet auf Vektoren $x_1, \ldots, x_k$
- die vorige Interp. für SRS: $[a](x) = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \cdot x + \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

```
( a , Multilinear   { absolute = column[0,1]
   , coefficients = [ matrix[[1,1],[0,0]] ]   } )
```

Synthese von Gewichtsfunktionen (für SRS)

- Gewicht ist Abbildung $g : \Sigma \to$
  forgesetzt zu $g^* : \Sigma^* \to$: $w \mapsto \sum_i g(w_i)$
  Bsp: $g = \{a \mapsto 2, b \mapsto 5\}$, $g(bab) = 12$
- das ist wfmA $(, >, [\cdot])$ mit $[\epsilon] = 0$, $[c](x) = g(c) + x$
- ist kompatibel mit R, falls $\forall (l, r) \in R : g^*(l) > g^*(r)$
  Bsp (Verifikation) $R = \{bab \to aaba, a^3 \to b\}$
- Synthese: gegeben R, kompatible g durch lin. Ungls.
  $(1-3)g_a + (2-1)g_b > 0 \wedge (3-0)g_a + (0-1)g_b > 0$
  Lösbarkeit ist effizient entscheidbar
- als Terminations-Verfahren erscheint das sehr einfach, kann aber nach einer Transformation, welche die Signatur vergrößert, sehr wirksam sein.

Modulare SN-Beweise, relative Termination

- Def: $\to_1 \,/\, \to_2$ ist die Relation $\to_1 \cdot \to_2^*$

- Satz: $\mathsf{SN}(\to_1 \,/\, \to_2) \wedge \mathsf{SN}(\to_2) \implies \mathsf{SN}(\to_1 \cup \to_2)$.

  Sprechweise $\mathsf{SN}(\to_1 \,/\, \to_2)$: $\to_1$ *terminiert relativ zu* $\to_2$.

- Beweis (indirekt): jede gemischte Ableitung enthält nur endlich viele $\to_1$, nach dem letzten nur endlich viele $\to_2$.

- Def: wfmA $(D, >, [\cdot])$ *schwach kompatibel* mit S: $\forall (l, r) \in S, x \in D : [l](x) \geq [r](x)$

- Satz: Wenn ex. wfmA D: D ist kompatibel mit R und D ist schwach kompatibel mit S, dann $\mathsf{SN}(\to_R \,/\, \to_S)$.

- Bsp: $\mathsf{SN}(\{c \to aabb, ab \to ba\})$, D zählt die c.

Aufgaben

1. Folie Matrix-Interpretationen: das Verifikations-Problem für das Beispiel.

   Beschreiben Sie möglichst genau die Komponenten des Vektors $[w](0,0,1)^T$ für $w \in \{a, b\}^*$ (Beispiele, allgemein)

   Für Rechnungen mit Matrizen z.B. maxima verwenden.

2. Wenn

   - $A = (D_A, >_A, [\cdot]_A)$ eine wfmA kompatibel mit R und schwach kompatibel mit S,
   - und $B = (D_B, >_B, [\cdot]_B)$ eine wfmA kompatibel mit S:

   konstruieren Sie eine wfmA mit Universum $D_A \times D_B$, die kompatibel mit $R \cup S$ ist.

   Lösungsplan:

   - Operationen: komponentenweise. $[f](x, y) = ([f]_A(x), [f]_B(y))$
   - Ordnung auf dem Universum angeben (unter welchem Namen kennen Sie diese?),
   - Monotonie und Kompatibilität nachweisen.

3. Für A, B, R, S wie in voriger Aufgabe:

   wenn A und B durch Gewichtsfunktionen g_A, g_B definiert sind, dann gibt es eine Gewichtsfunktion g, die mit $R \cup S$ kompatibel ist.

(also eine kompatible wfmA auf . Nach voriger Aufgabe gibt es eine solche auf ^{2}.)

Anwenden auf $R = \{bab \to aaba\}, S = \{a^3 \to b\}$, $g_A = \{a \mapsto 1, b \mapsto 3\}, g_B = \{a \mapsto 1, b \mapsto 0\}$

4. durch Beispiele illustrieren: wenn $\mathsf{SN}(\to_1)$ und $\mathsf{SN}(\to_2)$ und $(\to_1 \cup \to_2)$ transitiv, dann $\mathsf{SN}(\to_1 \cup \to_2)$.

   Beweis ist wohl schwierig

5. (von Hand, evtl. autotool) bestimme wfmAs für

   - $\{aa \to bbb, bb \to a\}$
   - Peano-Multiplikation (Variante A: ohne Additionsregeln, B: mit)
   - $\{p \wedge (q_1 \vee q_2) \to (p \wedge q_1) \vee (p \wedge q_2), (p_1 \vee p_2) \wedge q \to (p_1 \wedge q) \vee (p_2 \wedge q)\}$.

   Hinweis: Dershowitz: 33 examples.

6. Termination des TRS zur Booleschen Auswertung (vorige Woche)

7. Familien von langen Ableitungen und Terminationsbeweis für die SRS

   - $R_1 = \{ba \to acb, bc \to abb\}$
   - $R_2 = \{ba \to acb, bc \to cbb\}$
   - $R_3 = \{ba \to aab, bc \to cbb\}$

   Hinweis: eines hat doppelt exponentielle Ableitungslängen, eines mehrfach exponentielle, eines ist nichtterminierend.

   Beispiel: einfach exponentielle Ableitungslängen für $ba \to aab$: $\forall k : ba^k \to^* a^{2k}b$, $\forall k : b^k a \to^* a^{2^k}b^k$.

   Hinweis: jede Regelanwendung verlängert um 1, deswegen $u \to^k v \implies k = |v| - |u|$, man muß also die Schritte nicht zählen, sondern nur die Wortlängen vergleichen.

8. Geben Sie ein *längen-erhaltendes* terminierendes SRS mit exponentiellen Ableitungslängen an. Hinweis: Binärzähler.

   Alphabet und Regelanzahl beliebig. Das SRS möglichst kurz (Summe aller Längen aller Regelseiten)

   (Mit einer Regel geht das nicht. Wenn die terminiert, sind die Ableitungslängen höchstens quadratisch.)

9. für alle o.g. Aufgaben u. die restlichen von voriger Woche

   verwenden Sie die Terminationsbeweiser: AProVE (RWTH Aachen), TTT2 (U Innsbruck), Matchbox `https://gitlab.imn.htwk-leipzig.de/waldmann/pure-matchbox`. Selbst kompilieren/installieren oder Web-Oberfläche.

11 Konfluenz und Vervollständigung

Motivation, Beispiel

- gegeben E, gesucht: Entscheidungsverfahren für $s \approx_E t$

  Bsp: $E = \{aba \to \epsilon\}$, $s = abbbaa, t = bba$.

  Plan: Vergleich der E-Normalf. von s, t. Aber: $\neg(\mathsf{UN}(E))$,

- Lösung: Vergleich der $\to_F$-Normalformen für ein F mit

  - Hinzufügen von Regeln $E = E_0 \subseteq E_1 \subseteq \cdots \subseteq E_n = F$
  - Invariante: $E_i \subseteq (\approx_E)$ und $\mathsf{SN}(E_i)$ (dann $(\approx_{E_i}) = (\approx_E)$)
  - Ziel: $\mathsf{WCR}(F)$ (dann gilt $\mathsf{CR}(F)$ und $\mathsf{UNC}(F)$)

- Bsp: $ba \leftarrow_E ababa \to_E ab$, also $ba \approx_E ab$,

  $E_1 = E \cup \{ba \to ab\}$, $\mathsf{SN}(E_1)$ wg. length-lex-order

  E_1 nicht lokal konfluent wegen ..., neue Regel ...,

Kritische Paare

Def: Für ein Termersetzungssystem R über Σ: falls

- $(l_1 \to r_1)$ und $(l_2 \to r_2)$ sind Regeln in R ohne gemeinsame Variable (ggf. vorher umbenennen!)
- es gibt $p \in (l_1)$, so daß $l_1[p] \notin$ und

  $l_1[p]$ und l_2 sind ***unifizierbar*** mit ***mgu*** σ, d.h., $l_1[p]\sigma = l_2\sigma$
- falls $(l_1 \to r_1) = (l_2 \to r_2)$ (vor Umbenennung), dann $p \neq \epsilon$
- dann ist $(r_1\sigma, (l_1\sigma)[p := r_2\sigma])$ ***kritisches Paar*** (CP) von R.

Bsp: $\{f(f(x,y),z) \to_1 f(x,f(y,z)), f(i(x),x) \to_2 e()\}$

- CP aus Überlappung von 1 und 2 bei Position $[0]$ (in 1)
- CP aus Überlappung von 1 mit sich

Kritische Paare und (lokale) Konfluenz

- Def: CP (s, t) *zusammenführbar*, falls $\exists u : s \to_R^* u \wedge t \to_R^* u$.
- Wenn $\mathsf{SN}(R)$, dann ist Zusammenführbarkeit entscheidbar
- Wenn R endlich, dann hat R endlich viele CP.
- $\mathsf{WCR}(R)$ gdw. jedes kritische Paar zusammenführbar.
- Satz: Für endliche terminierende R ist Konfluenz entscheidbar.

Bsp. Kritische Paare für SRS

- $E_0 = \{aba \to \epsilon\}$, als TRS: $\{a(b(a(x))) \to x\}$
- Überlappung der Regel mit sich an Position $[0, 0]$
  $l_1 = aba(x), r_1 = x, l_2 = aba(x'), r_2 = x', p = [0, 0]$
  $l_1[p] = a(x)$, $\mathsf{mgu}(l_1[p], l_2) = \{x \mapsto ba(x')\} = \sigma$
  das aus dieser Überlappung konstruierte CP ist
  $(r_1\sigma, (l_1\sigma)[p := r_2\sigma]) = (x\sigma, (aba(x)\sigma)[p := x'\sigma]) = (ba(x'), ababa(y)[p := x']) = (ba(x'), ab(x'))$
- Ü: alle CP für $E_1 = E_0 \cup \{ba(x') \to ab(x')\}$
- Ü: alle CP für $\{aba(x) \to b(x)\}$

Bsp. kritische Paare für TRS

- CP für $\{f(x, x) \to a, f(x, g(x)) \to b\}$
- CP für $\{0 + y \to_1 y, s(x) + y \to_2 s(x + y), x + 0 \to_3 x, x + s(y) \to_4 s(x, y)\}$

Orthogonale Systeme

- ein TRS R über Σ heißt *nichtüberlappend*, wenn R keine kritischen Paare besitzt.
  (dann gilt $\mathsf{WCR}(R)$ trivialerweise)
- ein TRS R heißt linkslinear, wenn in keiner linken Regelseite eine Variable mehrfach vorkommt.
- ein TRS R heißt *orthogonal*, falls es linkslinear und nichtüberlappend ist.
- Satz: jedes orthogonale System ist konfluent.
  beachte: das gilt auch für nicht-terminierende Systeme
- braucht man *linkslinear* wirklich? Ja:
  $\{f(x,x) \to a, f(x,g(x)) \to b, c \to g(c)\}$

Vervollständigung (basic)
. . . ist Algorithmus mit dieser Spezifikation

- Eingabe: Gleichungssystem E
- Ausgabe: Ersetzungssystem R mit Eigenschaften:
  - R ist *konvergent* (:= terminierend und konfluent)
  - $\approx_R \;=\; \approx_E$

Implementierung: zusätzliche Eingabe: eine wfmA A

- ordne Paare aus E bzgl. $>_A$, Resultat R_0
- wiederhole bis $R_{i+1} = R_i$:
  $R_{i+1} := R_i \cup$ die folgenden Paare: $\forall (s,t) \in (R_i)$:
  - wähle $s' \in_{R_i} (s), t' \in_{R_i} (t)$
  - falls $s' \neq t'$, ordne (s', t') bzgl. $>_A$.

Korrektheit? Termination? beachte: 3 Fehlermöglichkeiten

Vervollständigung (improved)
bei einfacher Vervollständigung:

- jedes neue kritische Paar muß sofort orientiert werden.
- manche sind evtl. nicht orientierbar

alternativer Ansatz:

- das Orientieren aufschieben,
- bis neue Regeln vorliegen, die zur Umformung des Paares benutzt werden können
- so daß das umgeformte Paar orientierbar ist

(nichtdeterministischer) Vervollständigungsalgorithmus wird formuliert mittels Inferenzsystem

Inferenzsystem für Vervollständigung

(Leo Bachmair 1991, zitiert in Baader/Nipkow)

- Grundbereich: Paare (E, R) , Start: $(E_0, \emptyset)$, Ziel: $(\emptyset, R_n)$
- Invariante: $\rightarrow_{R_k} \subseteq >_A$ und $\approx_{E_0} = \approx_{E_k \cup R_k}$
- Fairness: $(R_n) \subseteq \bigcup_k E_k$

Regeln (weiter evtl. in Übung)

- deduce: $(s \leftarrow_R u \rightarrow_R t) \Rightarrow (E, R) \vdash (E \cup \{s \approx t\}, R)$
- orient: $(s >_A t) \Rightarrow (E \cup \{s \approx t\}, R) \vdash (E, R \cup \{s \rightarrow t\})$
- delete: $(E \cup \{s \approx s\}, R) \vdash (E, R)$
- simplify: $(s \rightarrow_R^+ u) \Rightarrow (E \cup \{s \approx t\}, R) \vdash (E \cup \{u \approx t\}, R)$.

Korrektheit?

Warum ist einfache Vervollst. ein Spezialfall?

Aufgabe zu Vervollständigung in autotool

- Gleichung: z.B. $(f(1, f(2, 3)), f(f(1, 2), 3))$,

  Variablen als Zahlen geschrieben

- wfmA: Argumente sind $x_1, x_2, \ldots$, z.B.

```
Interpretation { dim = 1
, values = Polynomial_Interpretation (listToFM
 [ ( f , 2 * x1 + x2 + 1 ) , ( a, 0) ]) }
```

- Schritte: Deduce, Orient, Delete, Simplify, z.B.

```
, steps = [ Orient { s = f (a , a ) , t = a } ]
```

- Rechnung ist Folge $\ldots (E_i, R_i) \rightarrow (E_{i+1}, R_{i+1}) \ldots$

  wie auf voriger Folie

Vervollständigung: Knuth/Bendix, 1970

- Donald Knuth and Peter Bendix: *Word Problems in Universal Algebras* 1970
- E sind Gruppen-Axiome, Signatur $\{e/0, i/1, m/2\}$

  1. $m(e(), x) \to x$
  2. $m(i(x), x) \to e()$
  3. $m(m(x, y), z) \to m(x, m(y, z))$

  abgeleitet z.B.: 8. $m(x, e()) \to x$
- kritische Paare wie hier definiert, orientiert nach der (später so benannten) Knuth-Bendix-Ordnung (KBO):
  - abnehmende Summe von Buchstabengewichten $\in$
  - bei gleicher Summe: lexikogr. Abstieg bzgl. $>$ auf Σ
  - Vorsicht bei Gewichten 0, damit $\mathsf{SN}(>_{\mathsf{KBO}})$

Wortprobleme für Gruppen, ab 1900

- das war eine Motivation für Knuth und Bendix
- Monoid-*Darstellung* (für Gruppe ähnlich): verwendet
  - endliche Menge von Generatoren (Buchstaben)
  - endliche Menge von Gleichungen (Wortpaaren)

  $\langle a, b \mid ab = ba \rangle$ für $\{a, b\}^* / \approx_E$ mit $E = \{ab \to ba\}$
- Kombinatorische Gruppentheorie (Walter Dyck, 1881) `https://zenodo.org/record/1942346/preview/article.pdf`
- $\approx_E$ ist das *Wortproblem* (Max Dehn 1911)
- ist einer der Ursprünge für die spätere Entwicklung der Begriffe *Algorithmus* und *Berechenbarkeit*.
- es gibt E, für die $\approx_E$ nicht entscheidbar ist (Novikov 1955)

Wiederholung: Unifikation

- ein *Unifikator* von zwei Termen $s, t \in (\Sigma, V)$
  ist eine Substitution $\sigma : V \to (\Sigma, V)$ mit $t\sigma = s\sigma$
- zwei Terme s, t können keinen, einen oder mehrere Unifikatoren haben
- Substitutionen nacheinander ausführen: $\sigma \circ \tau$
- Substitutionen ordnen durch $\sigma_1 \leq \sigma_2$ (σ_1 ist allgemeiner als σ_2) falls $\exists \tau : \sigma_2 = \sigma_1 \circ \tau$
- Satz: Wenn s, t unifizierbar, dann gibt es einen allgemeinsten Unifikator (most general unifier) σ von s und t: für jeden Unifikator σ_2 von s und t gilt $\sigma \leq \sigma_2$.
  $\mathsf{mgu}(s, t)$ ist bis auf Umbennenung eindeutig.

Wiederholung: Bestimmung des mgu

- falls $s = t$, dann return identische Abbildung
- falls s Variable
  - falls s in t vorkommt, dann fail
  - sonst return $(s \mapsto t)$
- falls t Variable entsprechend
- falls $s = f(x_1, \ldots)$ und $t = g(y_1, \ldots)$, dann
  - falls $f \neq g$, dann fail, sonst ...
  - bestimme $\sigma = \mathsf{mgu}(x_1, y_1)$
  - return $\sigma \cup \mathsf{mgu}((x_2\sigma, \ldots), (y_2\sigma, \ldots))$

Dieser Algorithmus ist korrekt, aber nicht effizient.

Hausaufgaben
empfohlen: jeder wenigstens eine konkrete Rechen-Aufgabe und eine Literatur-Aufgabe.

1. zu Folie Kritische Paare für TRS: kritische Paare ausrechnen.
   Passende wfmA angeben und vervollständigen (soweit möglich).

2. zu Folie Orthogonale Systeme: für das Beispiel nachrechnen: WCR, $\neg$ CR.

   Das durch diese Regelmenge erzeugte ARS mit früheren Beispielen für WCR und $\neg$ CR vergleichen.

3. Zeigen Sie Konfluenz von $\{1 \vee 1 \to 1, x \vee y \to y \vee x\}$. Welche der in der VL genannten Sätze und Methoden sind hier anwendbar?

4. Beispiele und Beweis: die Dyck-Sprache ist die $\approx_D$-Äquivalenzklasse von ϵ für $D = \{ab \to \epsilon\}$.

   Weil D konvergent ist (Beweis?) ist jede $\approx_D$-Äquivalenzklasse durch eine D-Normalform charakterisiert.

   Diese D-Normaformen sind $b^p a^q$. Die D-Normalisierung ist also eine Abbildung $f : \{a, b\}^* \to^2$.

   Für $f(w_1) = (p_1, q_1)$ und $f(w_2) = (p_2, q_2)$, geben Sie $f(w_1 \cdot w_2) = (p, q)$ als Funktion von p_1, q_1, p_2, q_2 an.

   (Auf Deutsch: f ist ein Monoid-Homomorphismus von $(\{a, b\}^*, \epsilon, \cdot)$ nach $(^2, (0, 0,), ?)$ und das Fragezeichen ist gesucht. Recherche-Aufgabe: wie heißt dieses Ziel-Monoid?)

5. weitere Regeln für Inferenzsystem für Vervollständigung

   Siehe Winkler et al. (JAR 2013) `https://www.uibk.ac.at/informatik/computational-logic/research/publications/publications-2013/pdfs/13jar2.pdf`, Fig. 1, zitiert aus: Bachmair (TCS 1991).

   entsprechende Erweiterung der Autotool-Aufgabe diskutieren (. . . und realisieren als Masterarbeit)

   Vergleichen mit der Formulierung des Knuth-Bendix-Algorithmus in Huet (1980) `https://hal.inria.fr/inria-00076536/file/RR-0025.pdf`

6. Gruppendarstellungen und Anwendung von Knuth-Bendix: siehe Havas et al.: *Some Challenging Group Presentations* (J. Austr. Math. Soc., 1999) `https://staff.itee.uq.edu.au/havas/1999hhkr.pdf`

7. mit Original-Artikel von Knuth u. Bendix:

   - wie werden Terme notiert?
   - wo steht die Definition von KBO?
   - Vervollständigungsschritte nachrechnen (das Entstehen des CP sowie seine passende Orientierung)

zur Geschichte von Knuth/Bendix siehe auch Donald Knuth: *All Questions Answered*, Notices Vol 49(3), `https://www.ams.org/notices/200203/fea-knuth.pdf`

8. (als Wiederholung und Ergänzung zu Termination) in KBO: welche Symbole dürfen Gewicht 0 haben? Warum die anderen nicht? (Gegen-Beispiele für $\mathsf{SN}(>)$ angeben)

   TTT2 (Weboberfläche) kann auch KBO.

12 Polynome

Motivation (I): Polynom-Interpretationen

- Anwendung: wfmA $A = (, >, [\cdot])$, jedes $[f]_A$ ist Polynom.

  Bsp: $[f](x, y) = x^2 \cdot y$ (Vorsicht), $[g](x, y) = x^2 + y$

  Bsp: kompatibel mit $g(g(x, y), z) \to g(x, g(y, z))$?

- es müssen Aussagen der Form $\forall x, y, z : P(x, y, z) > 0$ (automatisch) nachgewiesen werden

- welche Approximation in der autotool-Aufgabe dazu?

- Zeige $x, y \geq 0 \Rightarrow (x + y)/2 \geq \sqrt[2]{xy}$

  Ü: Zeige $x, y, z \geq 0 \Rightarrow (x + y + z)/3 \geq \sqrt[3]{xyz}$

  Ü: Hilberts 17. Problem (nichtneg. P, das kein SOS ist)

Motivation (II): Algebraische Zahlen

Bsp: gesucht ist Minimalpolynom für $y = \sqrt{2} + \sqrt[3]{3}$

	1	$\sqrt[3]{3}$	$\sqrt{2}$	$\sqrt{2}\sqrt[3]{3}$	$\sqrt[3]{3^2}$	$\sqrt{2}\sqrt[3]{3^2}$
y^0	1	0	0	0	0	0
y^1	0	1	1	0	0	0
y^2	2	0	0	2	1	0
y^3	3	6	2	0	0	3
y^4	4	3	12	8	12	0
y^5	60	20	4	15	3	20
y^6	17	90	120	24	60	18

die letzte Zeile ist linear abhängig von den vorigen (Dimension des Vektorraumes ist ≤ 6), ergibt Darstellung von y^6 als rationale Linearkombination von $y^0, \ldots, y^5$.

Motivation (III): Geometrische Örter

- Satz: in jedem Dreieck liegen
  - die Seitenmitten
  - die Höhenfußpunkte
  - die Mitten der oberen Höhenabschnitte

  auf einem Kreis (Feuerbach-Kreis, 9-Punkte-Kreis).
- Beweis durch symbolisches Rechnen:
  - Koordinaten der Ecken $(A_1, A_2), (B_1, B_2), (C_1, C_2)$
  - ex. Polynom P, so daß: (X_1, X_2) liegt auf Umkreis der Seitenmitten gdw $P(A_1, A_1, B_1, B_2, C_1, C_2, X_1, X_2) = 0$
  - (X_1, X_2) ist ein Höhenfußpunkt: $Q(\ldots) = 0$
  - zu zeigen ist $Q(\ldots) = 0 \Rightarrow P(\ldots) = 0$
    das ist (später) eine Aussage über Polynom-Ideale

Semantik und Syntax von Polynomen

- Polynom (semantisch) als Funktion: $\lambda x.x^3 - 3x^2 + 3x - 1$
- Polynom (syntaktisch)

  als Menge von Monomen $\{1x^3, -3x^2, 3x^1, -1x^0\}$,

  als Folge von Koeffizienten $[1, -3, 3, -1]$

- symbolisches Rechnen mit Polynomen (elementare Arithmetik)
- als Grundlage für (später) Algorithmus von Buchberger zur Bestimmung von Gröberbasen
- Wdhlg. (Zahlen)bereiche (Gruppen, Ringe, Körper),
  Anwendung: Polynome in mehreren Variablen

Gruppen, Ringe, Körper

- Definitionen (Wiederholung)
  - Monoid: Addition, Null
  - Gruppe: wie Monoid, und Subtraktion
  - Ring: wie Gruppe und zusätzlich Multiplikation, Eins
  - Körper: wie Ring und zusätzlich Division
  - Vektorraum (ü. Körper): Gruppe, skalare Multiplikation
- Bsp: , , $/6$, $/7$, , , ^{2} , , $[X]$
- zusätzliche Eigenschaften (Beispiele)
  - Ring heißt nullteilerfrei: $\forall a, b : a \neq 0 \wedge b \neq 0 \Rightarrow (ab) \neq 0$
  - Ring heißt euklidisch (bzgl. Funktion $|\cdot| : R \rightarrow$),
    wenn $\forall a, b \neq 0 : \exists q, r : a = q \cdot b + r$ mit $|r| < |b|$

Polynome in einer Variablen

- Repräsentation:
  - dicht: Folge (Data.Sequence) der Koeffizienten
  - dünn: endliche Abbildung (Data.Map)
    Schlüssel: Grad, Wert: Koeffizient
    normalisierte Darstellung: nur Koeffizienten $\neq 0$ werden repräsentiert (dann einfacher Test auf $P = 0$)
- Auswertung: mit Horner-Schema

- Addition
  - in dichter Darstellung: `Data.List.zipWith (+)`
  - in dünner Darstellung: `Data.Map.unionWith (+)`
- Multiplikation: naiv: quadratisch, mit FFT: $n \log n$

Polynome in mehreren Variablen

- das wird in den allermeisten Anwendungen gebraucht
- mögliche Repräsentationen für $[X_1, X_2, \ldots]$
  - geschachtelt: $= ([X_1])[X_2, \ldots]$
    als 1-Var-Polynom, Koeffizienten sind $(n-1)$-Var-Pol.
  - flach: als endliche Abbildung
    Schlüssel: Monom (Bsp: $X_2^8 X_4^5$), Wert: Koeffizient
    Monom: als endliche Abbildung
    Schlüssel: Variable (Bsp: X_2, X_4) Wert: Exponent
- die geschachtelte Darstellung ist elegant (und man muß nichts neu programmieren)
  kann aber unpraktisch sein (wenn man „von oben“ eine innere Variable benutzt)

Hausaufgaben

1. Bezeichnungen:
   arithmetisches Mittel $A(x_1, \ldots, x_n) = (\sum x_i)/n$, geometrisches Mittel $G(x_1, \ldots, x_n) = \sqrt[n]{\prod x_i)}$.
   Aufgabe: Stelle $A(x_1, \ldots, x_4)^4 - G(x_1, \ldots, x_4)^4$ als Summe von Quadraten (SOS) von Polynomen dar.
   Hinweis: $A(x_1, x_2)^2 - G(x_1, x_2)^2 = (x_1 - x_2)^2/4$ und
   $A(x_1, x_2, x_3, x_4) = A(A(x_1, x_2), A(x_3, x_4)) \geq A(G(x_1, x_2), G(x_3, x_4)) \geq G(G(x_1, x_2), G(x_3, x_4)) = G(x_1, x_2, x_3, x_4)$.
   Zusatz: $A(x_1, x_2, x_3)^3 - G(x_1, x_2, x_3)^3$ ist kein SOS, kann aber als Bruch mit SOS im Zähler und $(x+y+z)$ im Nenner geschrieben werden. Dazu die SOS-Darstellung der A-G-Ungleichung für die 4 Werte $x_1, x_2, x_3, t = A(x_1, x_2, x_3)$ benutzen, denn $A(x_1, x_2, x_3, t) = t$ und $G(x_1, x_2, x_3, t)^4 = G(x_1, x_2, x_3)^3 \cdot t$.

Hintergrund: Bruce Reznick: *Some Concrete Aspects of Hilbert's 17th Problem*, 2000, `https://faculty.math.illinois.edu/~reznick/`.

2. das Minimalpolynom für $\sqrt{2} + \sqrt[3]{3}$ nach angegebenem Verfahren ausrechnen und überprüfen.

   Ähnlich für $\sqrt{3} + \sqrt{5}$ oder (z.B.) $\sqrt[5]{3} + \sqrt[5]{5}$

3. Das Polynom P für „liegt auf Umkreis der Seitenmitten" angeben (oder ähnlicher geometrischer Ort im Dreieck).

   o.B.d.A $A_1 = A_2 = B_1 = 0$ annehmen.

   Warum wäre zusätzlich $C_2 = 0$ doch eine B.d.A.?

4. für die in VL angegebene Implementierung von Polynomen:

   eine anderen Koeffizientenbereich (als `Rational`) benutzen, z.B. `Complex Rational`

   nichttriviale Rechnungen durchführen (z.B. Division von $X^{pq} - 1$ durch $X^p - 1$), Ergebnis prüfen,

   Laufzeit messen, Ausführung profilieren, teure Funktionen feststellen, ggf. verbessern

   Vergleichen mit derselben Rechnung in einem richtigen Computer-Algebra-System (maxima, fricas). (Nicht irgendwo online, sondern lokal, damit man messen und vergleichen kann.)

13 Gröbnerbasen

Begriff, Motivation

- Gröbnerbasis: endliche Darstellung für ein Polynom-Ideal (eine Menge mit Abschlußeigenschenschaften)
- mit der man entscheiden kann, ob ein gegebenes Polynom zum Ideal gehört.
- Beispiel-Anwendung: Beweis geometrischer Aussagen, die sich algebraisch formulieren lassen.
- Erfinder: Bruno Buchberger 1965

  benannt nach seinem Doktorvater Wolfgang Gröbner,

(https://genealogy.math.ndsu.nodak.edu/id.php?id=18956, C. F. Gauß $\rightarrow^*$ F. Klein $\rightarrow^* \ldots$)

Literatur

- B. Buchberger: Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems (1970) https://www3.risc.jku.at/people/buchberg/refereed_publications.html

  (in 1970-00-00-A: Verweis auf Implementierung im „Formelcode der ZUSE Z 23“)

- F. Baader und T. Nipkow: Term Rewriting and all that (Kapitel 8), Cambridge Univ. Press 1998.

- H.-G. Gräbe: Gröbnerbasen und Anwendungen (Skript, Uni Leipzig); http://www.informatik.uni-leipzig.de/~graebe/skripte/

Anwendung

- Beispiel (Feuerbachkreis): in jedem Dreieck gilt: die Höhenfußpunkte und die Seitenmitten liegen auf einem Kreis (es gilt noch viel mehr).

- algebraische Behandlung:

  - für jeden Punkt $P_i = (x_i, y_i)$ zwei Variablen,
  - für jede gegebene/konstruierte Bedingung eine Gleichung,
    ergibt Gleichungssystem
    (von Gleichungen der Form „Polynom $= 0$“).

- Die Frage ist, ob behauptete Aussagen daraus folgen,

  d. h., ob entsprechende Polynome dann auch $= 0$ sind.

Ideale

- Ring $R = (M, 0, +, 1, \cdot)$, (Bsp:)
- *Ideal* I (Bsp: $4 = \{4z \mid z \in\}$) ist Menge $\subseteq M$ mit

  - abgeschlossen unter Summe (mit sich) $\forall x \in I, y \in I : (x + y) \in I$.

  - abgeschlossen bzgl. Multiplikation mit M (mit allen) $\forall x \in I, y \in M : (xy) \in I$.

- von $B \subseteq M$ ***erzeugtes*** Ideal: $\mathrm{Ideal}(B) = \langle B \rangle = \{\sum c_i b_i \mid c_i \in M, b_i \in B\}$
  Bsp: $\mathrm{Ideal}(\{8, 12\}) = \langle 8, 12 \rangle = ?$
- von Ideal I erzeugte Kongruenz: $f \equiv_I g \iff (f - g) \in I$
- gesucht: Implementierung für $\equiv_{\langle B \rangle}$ für Polynome

Motivation für Polynom-Reduktion

- Ansatz für Entscheidungsverfahren für $g_1 \equiv_B g_2$
  - Normalformen bestimmen $g_1 \to_B^* g_1', g_2 \to_B^* g_2'$
  - und vergleichen $g_1' = g_2' \iff g_1 \equiv_B g_2$

  dazu muß $\to_B$ terminieren und konfluent sein
- $g \to_B h$ falls $\exists f \in B$, so daß $h = g - c \cdot f$
  (nach Ideal-Definition folgt daraus $g \equiv_B h$)
- Termination: geeignete Ordnung, so daß $g > h$
- (lokale) Konfluenz: Zusammenführen kritischer Paare durch Ergänzung der Basis

Ordnung auf Monomen

- totale und wohlfundierte Ordnung auf Monomen mit
  - schwach kompatibel mit Teilbarkeit:
    wenn $m_1 \mid m_2$, dann $m_1 \leq m_2$
  - monoton bzgl. Multiplikation: $m_1 < m_2$, dann $mm_1 < mm_2$
- Beispiele:
  - lexikografisch
  - erst nach Exponentensumme, dann lexikografisch
    (diese verwenden wir, wenn nichts anderes gesagt wird)

- Nicht-Beispiele:
  - Ordnung nach Exponent von X_1
  - Ordnung nach Exponentensumme

Ordnung auf Mengen von Monomen

- Ordnung $>$ auf Monomen $\Rightarrow$ Ordnung $\gg$ auf Polynomen:
  $P \gg Q$ falls $\mathsf{Sort}_>(\mathsf{Mon}(P)) >_{\mathsf{lex}} \mathsf{Sort}_>(\mathsf{Mon}(Q))$
- $\mathsf{Mon}(P)$: die Menge der Monome von P (ohne Koeffz.)
  $\mathsf{Mon}(3X^2Y - 5XZ^3 + 4XYZ) = \{X^2Y, XZ^3, XYZ\}$
- $\mathsf{Sort}_>(M)$: absteigend geordnet bzgl. $>$
  $\mathsf{Sort}_>(\{X^2Y, XZ^3\}) = [XZ^3, X^2Y, XYZ]$
- Satz: für jede zulässige Ordnung $>$ und endliche Menge V von Variablen: die Ordnung $\gg$ terminiert auf der Menge der Polynome über V

Reduktion von Polynomen

- Bezeichnungen: für $p \neq 0$ (Bsp: $p = 3X^2Y - 5XZ^2$)
  $H_>(p)$, der Kopf von p: das größte (bzgl. $>$) Monom (mit Koeffz.) (Bsp: $H_>(p) = 3X^2Y$)
  $R_>(p)$, der Rest von p, so daß $p = H(p) + R(p)$
- $p \to_f q$, falls p ein Monom $a \cdot m$ enthält, das durch $H(f)$ teilbar ist (exist. m' mit $m = H(f)m'$) und $q = p - am'f$.
- für endliche Menge F von Polynomen: $(\to_F) = \bigcup_{f \in F} (\to_f)$
- Satz: $\to_F$ terminiert. Beweis-Idee: $p \to_f q \implies p \gg q$
  das Monom $H(f)$ in q wird durch eventuell mehrere, aber jedenfalls kleinere ersetzt. (Details: Übung)

Gröbnerbasen

- die eben definierte Relation $\to_F$ ist nicht notwendig konfluent. Beispiel (8.2.10 aus Baader/Nipkow)

  $F = \{f_1, f_2\}$ mit $f_1 = x^2y - x^2, f_2 = xy^2 - y^2$. Ordnung $>$ auf Monomen: erst nach Exponentensumme, dann lexikografisch.

  $x^2y^2 \to_{f_1} x^2y^2 - y \cdot f_1 = x^2y \to_{f_1} x^2y - 1 \cdot f_1 = x^2$ und $x^2y^2 \to_{f_2} ? \to_{f_2} y^2$.

  Beides sind Normalformen, also $\to_F$ nicht konfluent.

- Definition: eine endliche Menge G von Polynomen heißt *Gröbnerbasis* für ein Polynomideal I, falls $\mathrm{Ideal}(G) = I$ und $\to_G$ konfluent.

- Nicht-Beispiel (Fortsetzung) F ist keine Gröbnerbasis.

S-Polynome

- Man erhält eine Gröberbasis für F durch Vervollständigung (Hinzufügen von Polynomen $s \in \mathrm{Ideal}(F)$)

- Def: $S(f_1, f_2) := f_1 \cdot m/H(f_1) - f_2 \cdot m/H(f_2)$.

  mit $m = \mathrm{lcm}(H(f_1), H(f_2))$ (kleinstes gemeinsames Vielf.)

  Satz: $S(f_1, f_2) \in \mathrm{Ideal}(F)$.

- Beispiel $F = \{f_1, f_2\}$ mit $f_1 = x^2y - x^2, f_2 = xy^2 - y^2$

  $S(f_1, f_2) = f_1y - f_2x = x^2y - xy^2$

- Satz: Wenn für alle $f_1, f_2 \in F : S(f_1, f_2) \to_F 0$, dann ist F eine Gröbnerbasis.

- Beispiel: $x^2y - xy^2 \to_F x^2 - y^2 \not\to_F 0$.

Der Buchberger-Algorithmus

- Eingabe: endliche Menge F von Polynomen und Ordnung $>$ auf Monomen. Ausgabe: eine Gröbnerbasis G für $\mathrm{Ideal}(F)$. Start: $G_0 = G$.

  1. wähle $f_1, f_2 \in G_i$,
     bestimme eine $\to_{G_i}$-Normalform s von $S(f_1, f_2)$.
  2. falls $s \neq 0$, dann $G_{i+1} = G_i \cup \{s\}$

- Satz: Dieser Algorithmus hält nach endliche vielen Schritten und G_i ist Gröbnerbasis für F
- Den Test (wähle f_1, f_2, so daß) geeignet implementieren (nicht immer alle Paare behandeln)

Buchberger-Alg., Beispiel

- (Fortsetzung, $F = \{f_1, f_2\}$ wie oben.)
- $G_0 = F$, dann $S(f_1, f_2) \to_F x^2 - y^2 = f_3$, also $G_1 = \{f_1, f_2, f_3\}$.
- Neue Paare $S(f_1, f_3) = f_1 - yf_3 = y^3 - y^2 =: f_4$ ist Normalform,
- $S(f_2, f_3) = xf_2 - y^2 f_3 = -xy^2 + y^4 \to_{f_2} y^4 - y^2 \to_{f_4} y^3 - y^2 \to_{f_4} 0$.
- mit Mupad:

```
groebner::gbasis([x^2*y-x^2,x*y^2-y^2],DegreeOrder)
```

Der Buchberger-Algorithmus (II)

- Wielange das dauert und welche Basis man erhält, hängt von der gewählten Ordnung auf Monomen ab.
- Die Schrittzahl kann doppelt exponentiell sein.
- Mit manchen Ordnungen geht es „meistens“ schneller, da gibt es aber nur Erfahrungswerte.
- Anwendung: Mit einer Gröbnerbasis kann man das Ideal-Membership-Problem entscheiden: $f \in \mathrm{Ideal}(G) \iff f \to_G 0$.

Hausaufgaben

empfohlen: wenigstens 5,3,4,2 mit Papier und maxima.

1. zur Ordnung auf Monomen:

   Variable und ihre Ordnung:

   ```
   newtype V = V Natural deriving Eq
   instance Ord V where compare (V i) (V j) = compare j i
   ```

dann ist $V_0 > V_1 > V_2 > \ldots$.

Monome und ihre Ordnung:

```
type D = Natural
newtype Mono v = Mono (M.Map v D) deriving Eq
instance Ord v => Ord (Mono v) where
  compare (Mono f) (Mono g) =
    compare (M.toDescList f) (M.toDescList g)
```

Überprüfen und begründen Sie, daß das eine korrekte Implementierung der lexikografischen Ordnung ist.

Ist folgende Implementierung äquivalent?

```
    compare (M.toAscList g) (M.toAscList f)
```

2. zur Ordnung auf Polynomen:

   für Polynome in Variablen $X > Y > Z$: geben Sie möglichst lange $\gg$-Ketten an, die bei $X^3 + Y^2Z$ beginnen

   - bzgl. der grad-lexikografische Ordnung auf Monomen
   - bzgl. der lexikografische Ordnung auf Monomen

3. bestimmen Sie nach dem Buchberger-Algorithmus eine Gröbner-Basis B für $I = \langle X^2Y - 1, XY^2 - 1\rangle$.

   - auf Papier
   - mit maxima (`load(grobner);poly_buchberger(...);`) oder fricas
   - mit Eigenbau-Implementierung (siehe Repo)

4. Benutzen Sie dieses B (oder B', siehe unten), um zu entscheiden, ob $X^5 \equiv_I Y^2$.

   - auf Papier
   - mit maxima (`poly_grobner_member`) oder fricas
   - mit Eigenbau-Implementierung

5. Geben Sie Polynome c_1, c_2 mit $X^5 - Y^2 = c_1(X^2Y - 1) + c_2(XY^2 - 1)$ an.

   Erweitern Sie den Buchberger-Algorithmus und das Reduktionsverfahren $\rightarrow_B$, um diese Darstellung zu erhalten.

   (zum Vergleich: Euklid bestimmt $\gcd(a, b)$, erweiterter Euklid bestimmt c, d mit $ac + bd = \gcd(a, b)$.)

6. Für eine Gröbner-Basis B definieren wir

$$B' = \{b \in B \mid \neg\exists c : b \rightarrow_{B\setminus\{b\}} c\}$$

   (alle b, die nicht durch die jeweils anderen Basis-Elemente reduziert werden können)

   Bestimmen Sie B' für voriges Beispiel.

   Ergänzen Sie die Eigenbau-Implementierung.

   (Es gilt $\langle B'\rangle = \langle B\rangle$. Jedes solche B' heißt *reduziert*. Die reduzierte Gröbner-Basis eines Polynom-Ideals ist im wesentlichen eindeutig.)

7. Arjen Cohen: *Gröbner Bases, an Introduction* (in: *Some Tapas of Computer Algebra*, Springer, 1999):

   > it is very hard to provide a good explicit upper bound on [die Laufzeit des Buchberger-Algorithmus], bad examples are known.

   Finden Sie solche *bad examples*

   - selbst ausdenken
   - mit Suchmaschine
   - brute force

   und messen Sie den Ressourcenverbrauch (maxima, fricas, Eigenbau)

   Erklärung: brute force mit unserer Implementierung und Leancheck-Generatoren für Monome und Polynome. Vorsicht: Generatoren sollen keine Exponenten oder Koeffizienten 0 erzeugen. Hinweis: Anzahl der Variablen fixieren, z.B. `data V = X | Y`. Deswegen ist der Code polymorph im Variablentyp `v`.

14 Gröberbasen in der Geometrie

Beispiel

- Quadrat $ABCD$, Punkt P auf Parallele zu BD durch C mit $BP = BD$. Schnittpunkt von CD und BP heißt Q. Zeige $DP = DQ$.
- algebraische Formulierung: Koordinaten: $A = (0,0), B = (1,0), C = (1,1), D = (0,1), P = (x,y), Q = (z,1)$.

  Lage: $CP \| BD$: $(P_x - C_y)(D_y - C_y) = (D_x - B_x)(P_y - C_y)$,

  Q auf CD: $CQ \| CD$, gleiche Länge: $|DP|^2 = |DQ|^2$.
- Quelle: Hans-Gert Gräbe: *Geometrie mit dem Computer (5. Geometrische Sätze vom Gleichungstyp)* 2018, `http://www.informatik.uni-leipzig.de/~graebe/skripte/`

Methode

- geometrische Lage von Punkten $[(x_1, y_1), \ldots, (x_k, y_k)]$

  beschrieben durch Aussage $p = 0$ für Polynom p
- geometrische Sätze beschrieben durch Implikation

  für alle $x_1, y_1, \cdots : (p_1 = 0 \wedge \cdots \wedge p_n = 0) \Rightarrow (q = 0)$
- das folgt aus $q \in \mathrm{Ideal}(p_1, \ldots, p_n)$.

  Beweis: Idealmitgliedschaft: $q = \sum c_i p_i$,

  dann $q(x_1, y_1, \ldots) = \sum c_i p_i(x_1, y_1, \ldots) = \sum c_i \cdot 0 = 0$
- Implementierung (einfache Variante): $q \in \mathrm{Ideal}(F)$,

  gdw. $q \xrightarrow{?}_B 0$ bzgl. einer Gröberbasis B für F.

  Ü: für das Beispiel ausrechnen

Methode (Variante)

- $p_1(x_1,\ldots) = 0 \wedge \cdots \wedge p_n(x_1,\ldots) = 0 \Rightarrow q(x_1,\ldots) = 0$.
- äquivalent: es gibt kein $x = (x_1,\ldots)$
  mit $p_1(x) = 0 \wedge \cdots \wedge p_n(x) = 0 \wedge q(x) \neq 0$.
- (der Trick von Rabinovitch) äquivalent: neue Var. f und
  $\neg\exists x$ mit $p_1(x) = 0 \wedge \cdots \wedge p_n(x) = 0 \wedge 1 - f \cdot q(x) = 0$.
- äquivalent: $\mathrm{Ideal}(p_1,\ldots,p_n, 1 - fq) = \mathrm{Ideal}(1)$
  Gröbnerbasis dafür kann man evtl. leichter berechnen
- Vgl. Resolution in der Logik: Falls $(A_1 \wedge \ldots \wedge A_n \wedge \neg B)$ widersprüchlich, dann $(A_1 \wedge \ldots \wedge A_n \implies B)$;
  aus widersprüchlicher Formel(menge) folgt jede Formel.

Anwendungen

- diese Notation für Mupad, Ü: übertrage auf andere CAS
- Seitenhalb. schneiden sich in einem Punkt:

```
kollinear:=(x,y,z)->((z.2-y.2)*(y.1-x.1)-(y.2-x.2)*(z.1-y.1))
mittelpunkt:=(a,m,b)->( (a.1-m.1)-(m.1-b.1),  (a.2-m.2)-(m.2-b.2) )

groebner::gbasis([
 mittelpunkt(a,f,b), mittelpunkt(b,d,c), mittelpunkt(c,e,a),
 kollinear(a,h,d), kollinear(b,h,e),
 1 - y * kollinear(c,h,f)
])
```

- Thaleskreis:

```
qdist:=(a,b)->((a.1-b.1)^2+(a.2-b.2)^2)
gleichweit:=(a,b,c,d)-> (qdist(a,b)-qdist(c,d))
senkrecht:=(a,b,c,d)-> ((a.1-b.1)*(c.1-d.1)+(a.2-b.2)*(c.2-d.2))

groebner::gbasis([
```

```
        senkrecht(a,c,b,c), mittelpunkt(a,m,b),
        1 - y*gleichweit(a,m,c,m)
 ])
```

- Satz von Napoleon (kann so gar nicht gehen, da die Dreiecke „nach außen“ zu errichten sind)

```
schwerpunkt:=(a,b,c,s)->(a.1+b.1+c.1-3*s.1,a.2+b.2+c.2-3*s.2)
gleichseitig:=(a,b,c)->(gleichweit(a,b,b,c),gleichweit(b,c,c,a))

groebner::gbasis([
        a.1,a.2,b.2,
        gleichseitig(a,b,p), schwerpunkt(a,b,p,u),
        gleichseitig(b,c,q), schwerpunkt(b,c,q,v),
        gleichseitig(c,a,r), schwerpunkt(c,a,r,w),
        1 - y * gleichseitig(u,v,w)
])
```

- im Projekt *Symbolic Data* haben H.-G. Gräbe et al. solche Beispiele erzeugt und gesammelt `https://web.archive.org/web/20120106071657/http://www.symbolicdata.org:80/`

  siehe auch `https://symbolicdata.github.io/`, `https://github.com/symbolicdata/data/tree/master/XMLResources/GeoProofSchemes`

Hausaufgaben

1. Beispiele für geometrische Theoreme aus Skript Gräbe 2018 (Simsonsche Gerade, Miquelscher Punkt, . . .)
   - mit Geogebra zeichnen, Theorem überprüfen (durch Betrachtung der Lage bei Verschieben von Punkten)
   - algebraisieren (automatisieren!)
   - nach beschriebenen Verfahren lösen (Gröberbasis mit verschiedenen CAS ausrechnen)
2. ein falsches Theorem beweisen (um zu sehen, wie die Methode die Falsch-Aussage erkennt)

3. noch offene Aufgaben aus vorigen Wochen, insbesondere zur Eigenimplementierung von Polynomen (Division!) und Buchberger-Algorithmus

4. *bad examples* (für Buchberger-Algorithmus) in verschiedenen CAS (viele naive Algebraisierungen von geometrischen Sätzen ergeben aufwendige Rechnungen, aber es gibt auch kleinere schwere Testfälle)

   Beispiel (Verallgemeinerung? Quelle?)

```
cyclic4 =
  let [a,b,c,d] = map (var . V) [1 .. 4]
  in  [ a*b*c*d-1
      , a*b*c+b*c*d+c*d*a+d*a*b
      , a*b+b*c+c*d+d*a
      , a+b+c+d
      ]
```

5. den Rabinovitch-Trick anwenden für den Satz über die Simson-Gerade in allgemeiner Lage (nur ein Punkt ist fixiert)

```
fsimson = prove $ do
  ...
  return $ colinear a' b' c'
```

   die letzte Zeile ersetzen durch

```
  z0 <- number
  assert $ ...
  return $ ...
```

   und Laufzeiten (der naiven Implementierung sowie anderer CAS) vergleichen

6. die Behandlung von Degenerations-Bedingungen,

   Beispiel: der Satz von Pappus

```
pappus = prove $ do
  a  <- point ; b  <- point; c  <- point_on a  b
  a' <- point ; b' <- point; c' <- point_on a' b'
  p <- intersection b c' b' c
```

```
q <- intersection a c' a' c
r <- intersection a b' a' b
return $ -- degenerate cases:
         parallel (b-c') (b'-c)
       * parallel (a-c') (a'-c)
       * parallel (a-b') (a'-b)
         -- actual conclusion:
       * colinear p q r
```

7. bessere Implementierungen des Buchberger-Verfahrens: nicht jedesmal *alle* S-Polynome bestimmen und bezüglich *aller* bisherigen reduzieren.

   Zum Code in `poly/Grobner.hs`:

   - die Invarianten beweisen
   - die Termination begründen
     - für jede einzelne Reduktion eines (S-)Polynoms
     - für das gesamte Buchberger-Verfahren