Programmierung und Modellierung

Zusammenfassung und Ausblick

Martin Wirsing

in Zusammenarbeit mit
Moritz Hammer

SS 2009

Inhalt

13.1 Thema der Vorlesung

- Einführung in grundlegende Prinzipien der **funktionalen**
  - **Programmierung und Datenmodellierung**
  - am Beispiel der Sprache **SML**

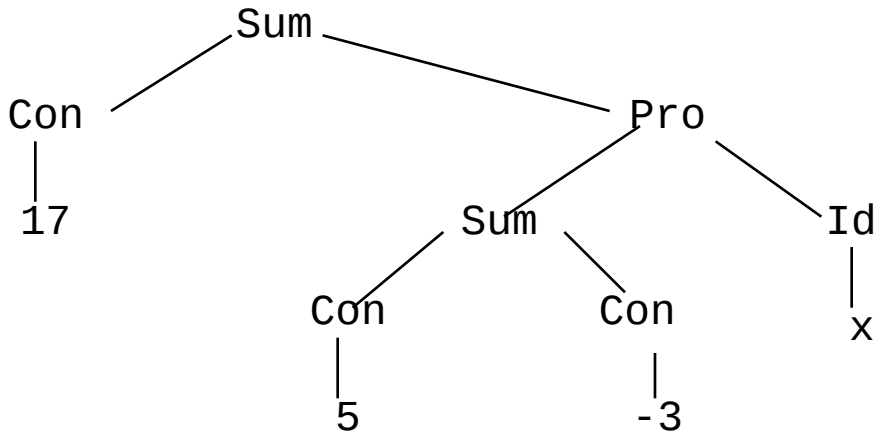
- Wesentlichen Themen waren:
  - Funktionen und Rekursion
  - Auswertung und Terminierung von Programmen
  - Rekursive Datentypen und Module
  - Syntaxanalyse, Fixpunktsatz zur denotationellen Semantik von BNF und Programmiersprachen.

13.2 Auswertung von Abstrakten Syntaxtermen

- **Abstrakte Syntax für arithmetische Terme**
  - `exp ::= num | id | e+e | e*e`
  - **In SML: Rekursiver Datentyp**

```
datatype exp = Con of int | Id of string
             | Sum of exp*exp | Pro of exp*exp;
```

- **Beispiel**
  - **17 + ((5-3) * x) als abstrakter Syntaxbaum:**



Auswertung von Abstrakten Syntaxtermen

- **Umgebung**
  - `type id = string;`
  - `type ‘a env = (id  -> ‘a);`
  - `exception UNDEF_ID;`
  - **Beispiel**

```
val v0 = fn “x” => 10
         | “y” => ~5
         | _   => raise UNDEF_ID;
```

Auswertung von Abstrakten Syntaxtermen

- **Auswertung arithmetischer Terme in einer Umgebung**
  - **Strukturelle Rekursion gemäß der Struktur des rekursiven Datentyps:**

```
fun eval (Con i) v =  i
|   eval (Id n)  v =  v(n)
|   eval (Sum (e1, e2)) v = (eval e1 v) + (eval e2 v)
|   eval (Pro (e1, e2)) v = (eval e1 v) * (eval e2 v);
```

  - **Beispiel**

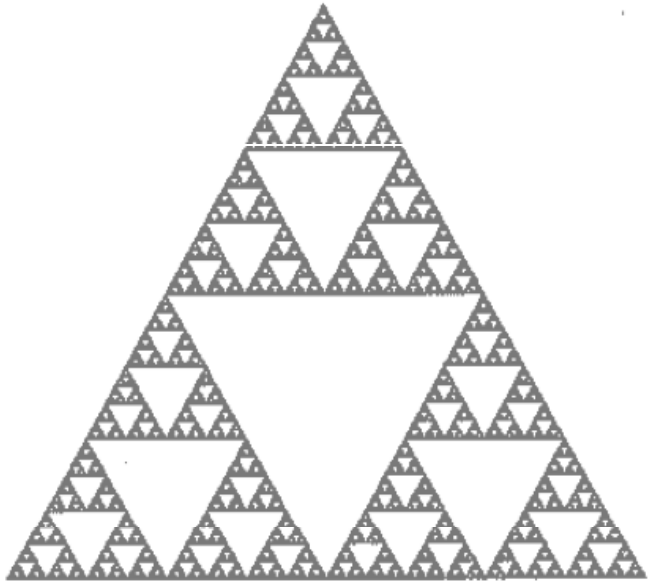
```
eval
  (Sum (Con 17, Pro (Sum (Con 5, Con ~3), Id “x”)))
  v0;
```

13.3 Fraktale und Mandelbrotmengen

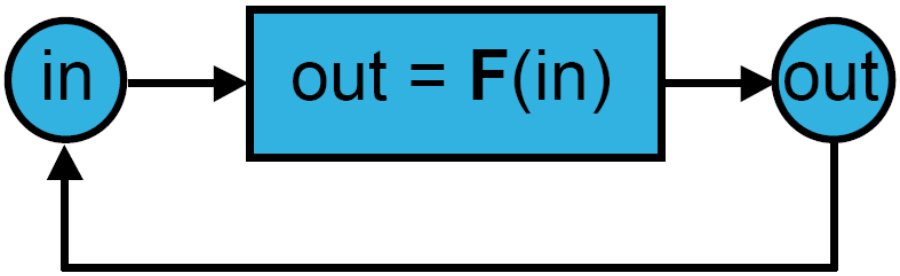
- Fixpunkte
  - Berechnung von Fixpunkten wird in der Informatik in unterschiedlichen Situationen benötigt:
    - Der **Satz von Knaster und Tarski** besagt, dass jede **monotone** Funktion f : $X \to X$ einen kleinsten Fixpunkt besitzt, falls X ein ω-cpo mit kleinstem Element ist.
    - Ist f stetig, so läßt sich der kleinste Fixpunkt nach dem **Fixpunktsatz von Kleene iterativ** als Supremum einer Kette von Approximationen berechnen.
    - Der Satz von Kleene liefert eine denotationelle Semantik für **BNF-Grammatiken** sowie für **rekursive Funktionen** und damit von SML.
  - (Fixpunkt-) Iteration spielt eine große Rolle bei Fraktalen

Fraktale

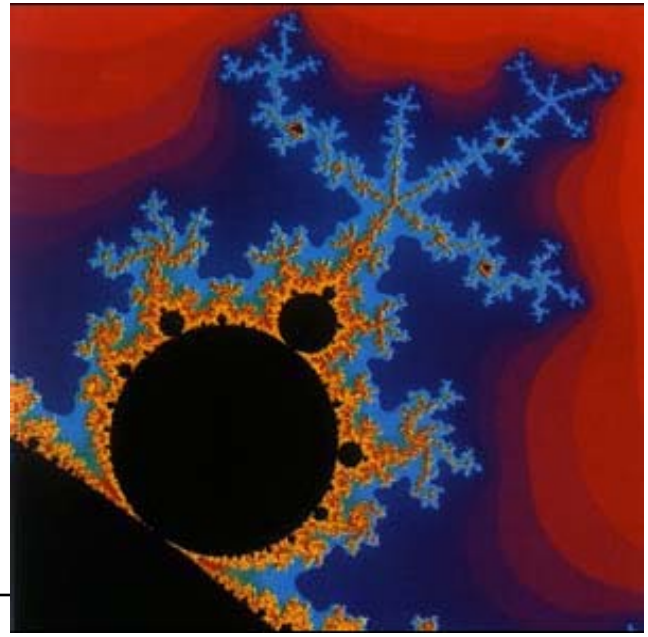
- Ein **Fraktal** ist ein natürliches oder künstliches Gebilde, das einen hohen Grad von Selbstähnlichkeit aufweist.
  - Fraktale entstehen durch einen Feedback Prozess („Dynamisches System“), bei dem eine Funktion F iteriert wird.



Sierpinsky Menge



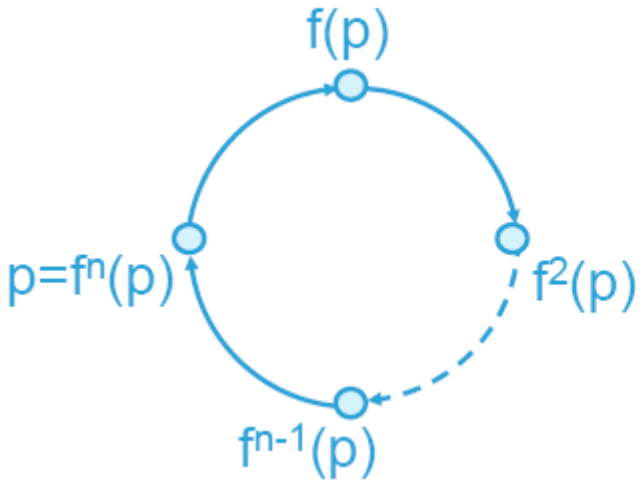
[Christph Traxler, Vorlesung Fraktale, TU Wien, 2008]



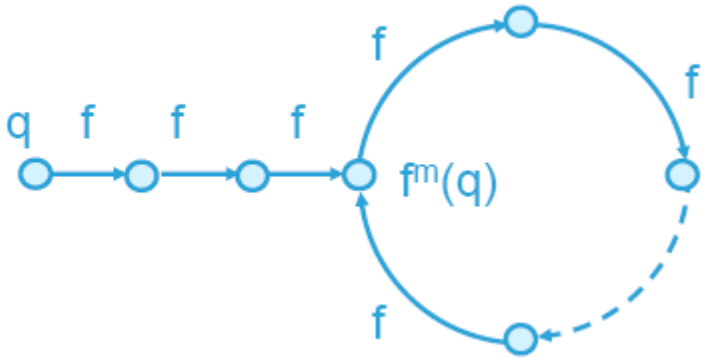
Mandelbrot Menge

Dynamisches System

- Sei f: X→X eine stetige Funktion, X ein metrischer Raum.
  - Dann heißt {X, f} **dynamisches System**.
  - Ein Punkt p ∈ X heißt **periodisch**, falls es ein n > 0 gibt, so dass $f^n(p) = p$ (d.h. p ist Fixpunkt von f).
  - {p, f(p),...,$f^n(p)$} heißt **Zyklus** von p und n die **Periode** von p
  - Ein Punkt p ∈ X heißt **quasi-periodisch**, falls es ein m > 0 gibt, so dass $f^m(q)$ periodisch ist.



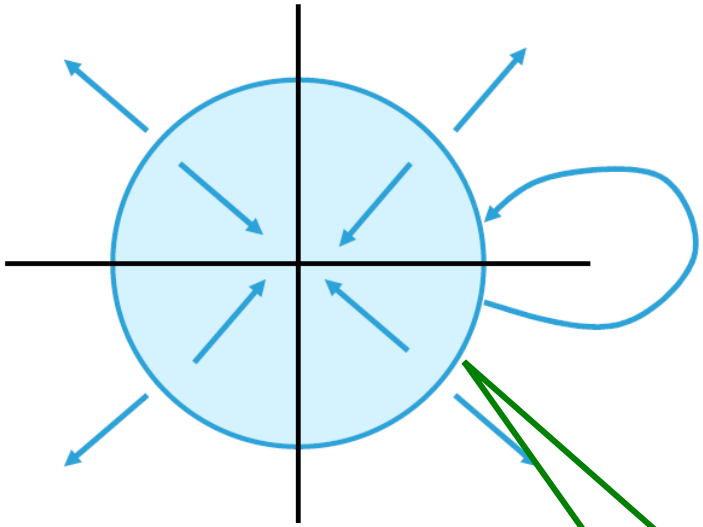
Periodisch



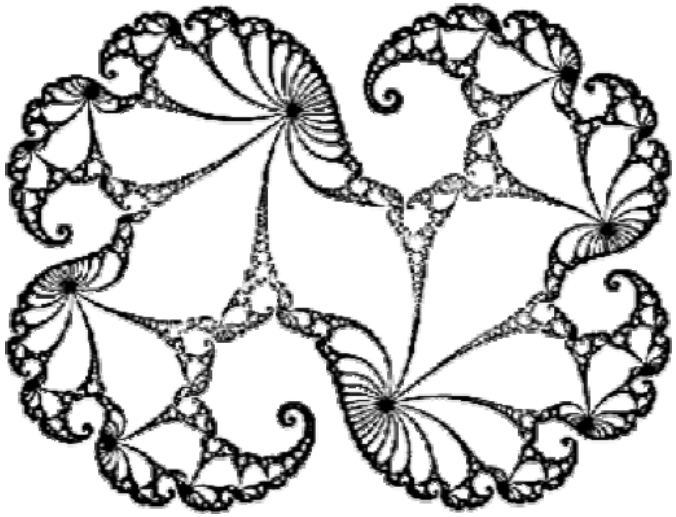
Quasi-periodisch

Julia-Mengen

- Iteration von Polynomen wie f(z) = z*z
- Partition der komplexen Ebene in 3 Mengen:
  - $|z|<1$: $(f^n(z))_{n\in N}$ konvergiert gegen 0
  - $|z|=1$: $(f^n(z))_{n\in N}$ bleibt auf dem Einheitskreis
  - $|z|>1$: $(f^n(z))_{n\in N}$ divergiert gegen ∞
- Der Rand zwischen divergierenden und konvergierenden Folgen heißt **Julia-Menge.**

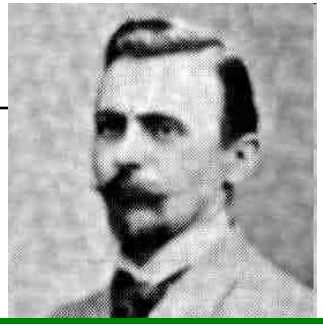


Iteration von z*z



Julia-Menge von z*z +c

Julia-Menge von z*z



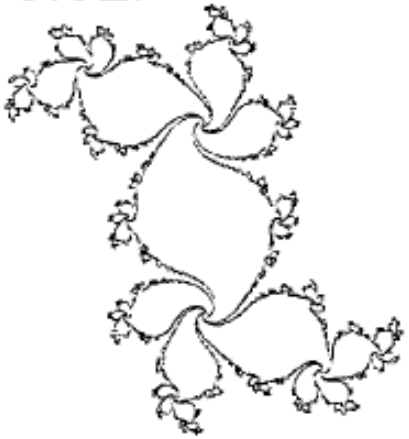
Pierre Fatou
1878 - 1929
franz. Mathematiker
Komplexe dynamische Systeme (mit Polynomen)

Gaston Julia
1893 - 1978
franz. Mathematiker
Zusammenarbeit mit Fatou
Julia-Mengen: Iteration komplexer rationaler Funktionen

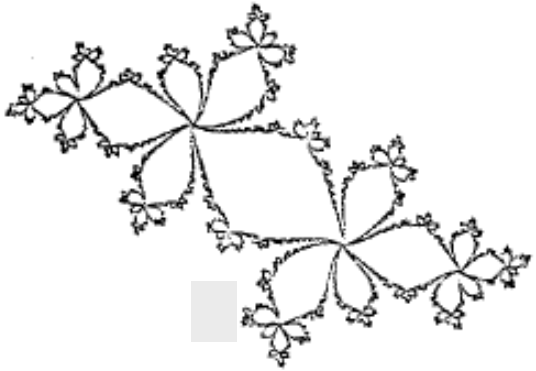
Julia-Mengen

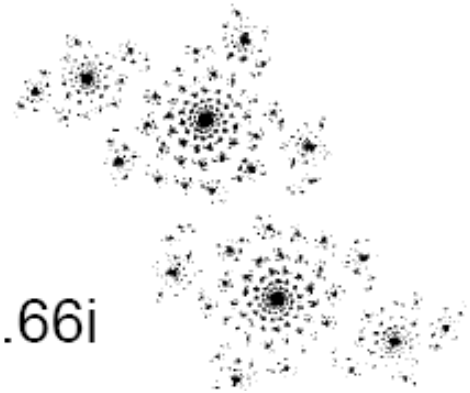
- Iteration von f(z) = z*z + c

c = 0.25+0.52i

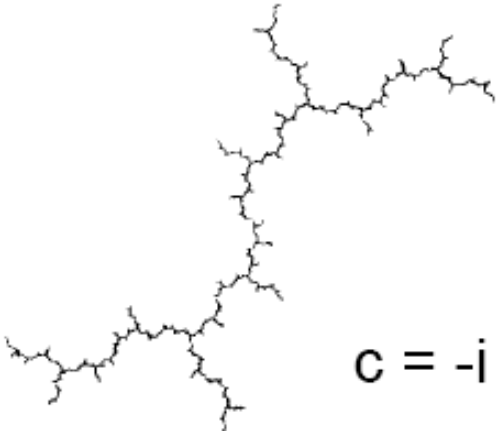


c = -0.5+0.55i





c = 0.66i

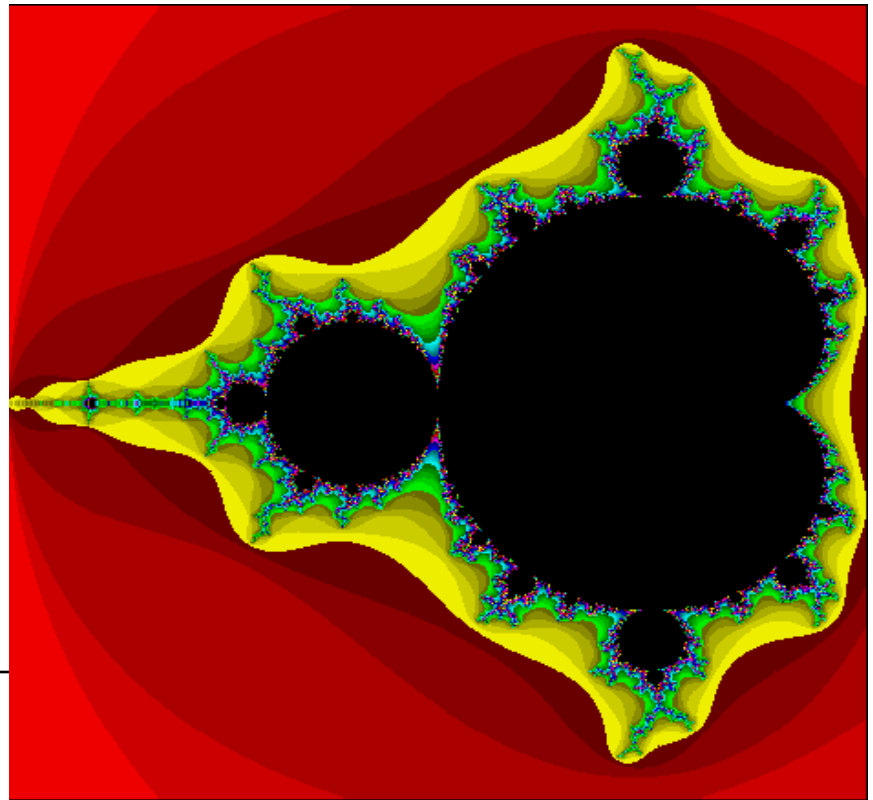


c = -i

Mandelbrot Menge

- Mandelbrot (1979): Konstruiere ein Bild der komplexen Ebene, das Informationen über die Julia-Menge J_c für jede komplexe Zahl zeigt, für die J_c zusammenhängend ist
- Idee: Betrachte Folgen $0 \rightarrow c \rightarrow c^2+c \rightarrow ...$, die in ihrer Größe beschränkt sind:
  - Sei $z_0 = (0,0)$ der Startpunkt
  - Iteriere $f(z) = z^2+c$

Benoît Mandelbrot
* 1924
franz. Mathematiker
IBM Fellow, Prof. Yale
Mitbegründer
Chaostheorie



Mandelbrot-Mengen in SML: Komplexe Zahlen

[Andrew Cumming: A gentle introduction to ML; Napier University, 1998]

```
val zero = (0.0, 0.0);
fun plus (x1,y1) (x2,y2) =
  (x1 + x2, y1 + y2) : real*real;
fun minus (x1,y1) (x2,y2) =
  (x1-x2, y1-y2) : real*real;
fun times (x1,y1) (x2,y2) =
  (x1*x2 - y1*y2, x1*y2 + x2*y1) : real*real;
fun square(x,y) = (x*x - y*y, 2.0*x*y) ;
fun scalar (s:real) (x,y) = (s*x, s*y);
fun dist p q =
  let fun r(x,y)=Math.sqrt(x*x + y*y)
   in r(minus p q) end;
```

Mandelbrot-Mengen in SML: Iteration

- **Iteration**

```
fun id x = x;
fun twice f = f o f;
fun iterate f n =
  if n=0 then id
  else if n = 1 then f
       else if n mod 2 = 0 then twice (iterate f (n div 2))
            else f o twice (iterate f (n div 2));
fun thrice f = iterate f 3;
fun quice f  = iterate f 4;
fun t256 f   = iterate f 256;
```

- **Mandelbrot-Polynom**

```
fun checkFloat(a,b) =
  (Real.checkFloat a, Real.checkFloat b);
fun man c z = checkFloat(plus (square z) c);
```

Mandelbrot-Mengen in SML: Beispiele für c

- Einige komplexe Zahlen

```
val unbound = (0.5, 0.5);
val quasiFix = (0.2, 0.2);
val quasiTwo = (~1.0, 0.0);
val quasiThree = (~0.1, 0.7);
val quasiFour = (~1.3, 0.0);

val bound = (0.15625, 0.5625);
```

- Iterationen

```
t256 (man fast) zero;
man fast it;
```

Mandelbrot-Mengen in SML: Test auf Periodizität

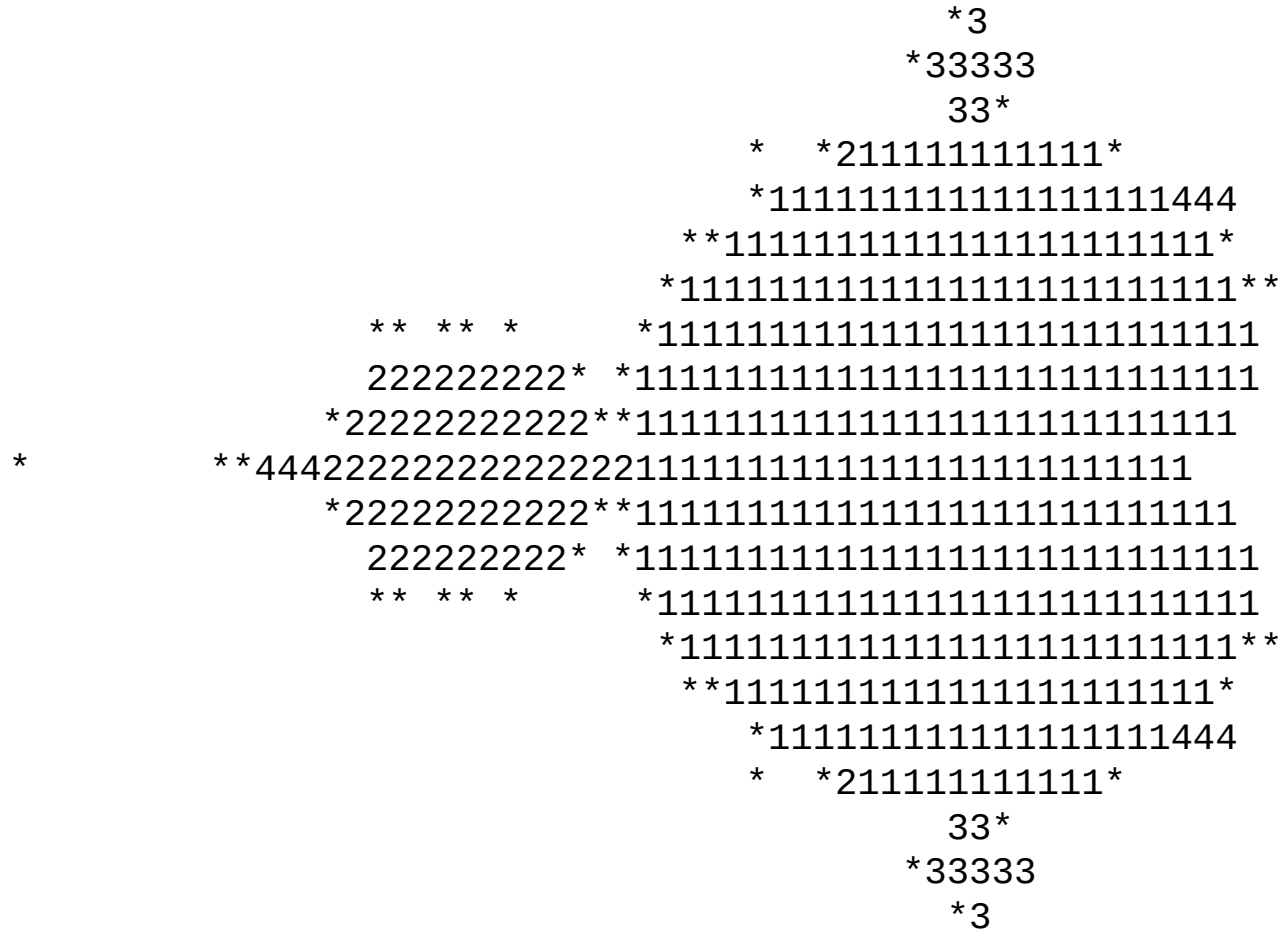
- Teste, ob Konstante c quasi-periodisch ist:
  - Iteriere man 256-mal und teste dann auf Periodizität:

```
fun cat p =
  let val small = 0.001;
      val a= t256 (man p) zero;
  in
  if dist a (man p a)<small then "1"
  else if dist a (twice (man p) a)<small then "2"
              else if dist a (thrice (man p) a)<small
                then "3"
            else if dist a (quice (man p) a)<small
                              then "4"
                  else "*" end
handle (Overflow | Div) => " ";
```

Mandelbrot-Mengen in SML: Formatierung und Druck

```
fun for (r1:real) r2 d f =
  if (0.0 > (r2-r1)*d) then ""
  else (f r1)^(for (r1+d) r2 d f);

fun line x1 x2 y =
  (for x1 x2 ((x2-x1)/78.0) (fn i=> cat(i,y)))^"\n";

fun box((x1,y1),(x2,y2))=
  for y2 y1 ((y1-y2)/24.0) (fn i=> line x1 x2 i);

fun K a b = b;

val ibox=( (~2.0,~1.0),(1.0,1.0));

fun doit x = K (print(box x)) x;

doit ibox;
```

Mandelbrot-Mengen in SML: Ausdruck



Mandelbrot-Mengen in SML: Weitere Formatierung

```
(* Box shifting stuff *)
fun zoom(p, q) =
  (add (scalar 0.75 p) (scalar 0.25 q),
   add (scalar 0.25 p) (scalar 0.75 q));
fun shift v (p,q) =
  let val w = dot v (sub q p) in (add w p, add w q)
  end;
val right = shift (0.5,0.0);
val left = shift (~0.5,0.0);
val up = shift (0.0,0.5);
val down = shift (0.0,~0.5);

doit (zoom ibox);
doit (zoom it);
doit (up(left it));
```

Ausblick: Software-Entwicklung

- Tätigkeiten bei der Software-Entwicklung
  - Anforderungsanalyse & Fachliche Konzeption
  - Technische Konzeption (Design, Entwurf)
  - Realisierung
  - Integration & Test
  - Wartung
- In dieser Vorlesung haben wir gelernt, deklarative funktionale Modelle von Softwaresystemen zu bilden.
- Im nächsten Semester werden alle Tätigkeiten der Software-Entwicklung gelehrt und geübt:
  - Vorlesung Objekt-orientierte Software-Entwicklung
  - Software-Entwicklungspraktikum

Vielen Dank für Ihre Aufmerksamkeit und Mitarbeit!

Moritz Hammer, alle Tutoren und ich wünschen Ihnen

viel Erfolg

in der Prüfung und

eine **angenehme**

vorlesungsfreie Zeit!