Programmierung 1

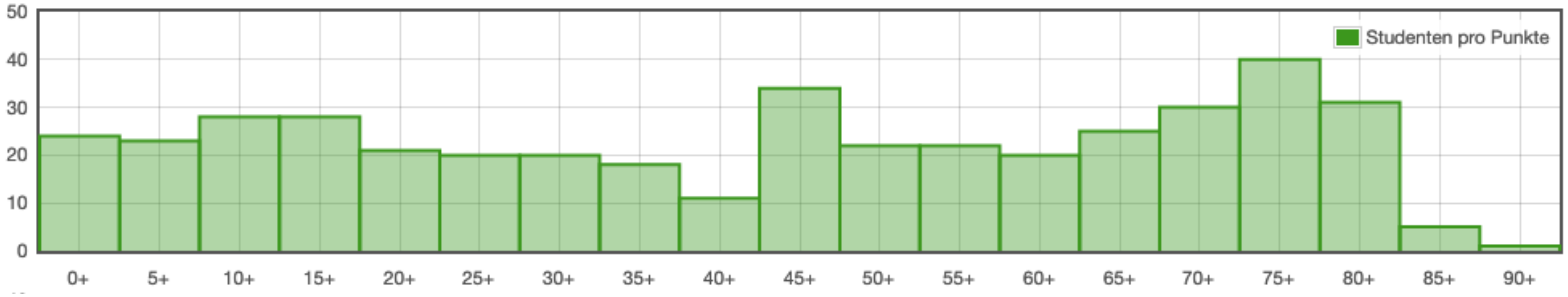
Vorlesung 13

Livestream beginnt um 14:15 Uhr

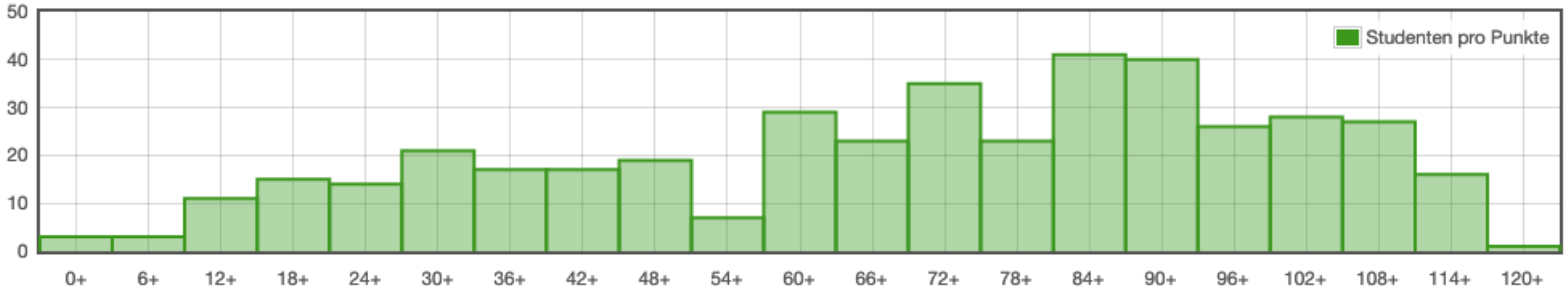
Mengenlehre
Mathematische Prozeduren

Programmierung 1

Mittelklausur (Haupttermin) WS 2020/2021



Mittelklausur (Haupttermin) WS 2017/2018



Kapitel 8
Mengenlehre

Axiomatische Beschreibung von Mengen

1. Eine **Menge** ist eine **Zusammenfassung von Mengen**. Die von einer Menge x zusammengefassten Mengen bezeichnen wir als die **Elemente** von x. Wir schreiben $\boldsymbol{x} \in \boldsymbol{y}$, um zu sagen, dass **die Menge** x ein **Element der Menge** y ist.

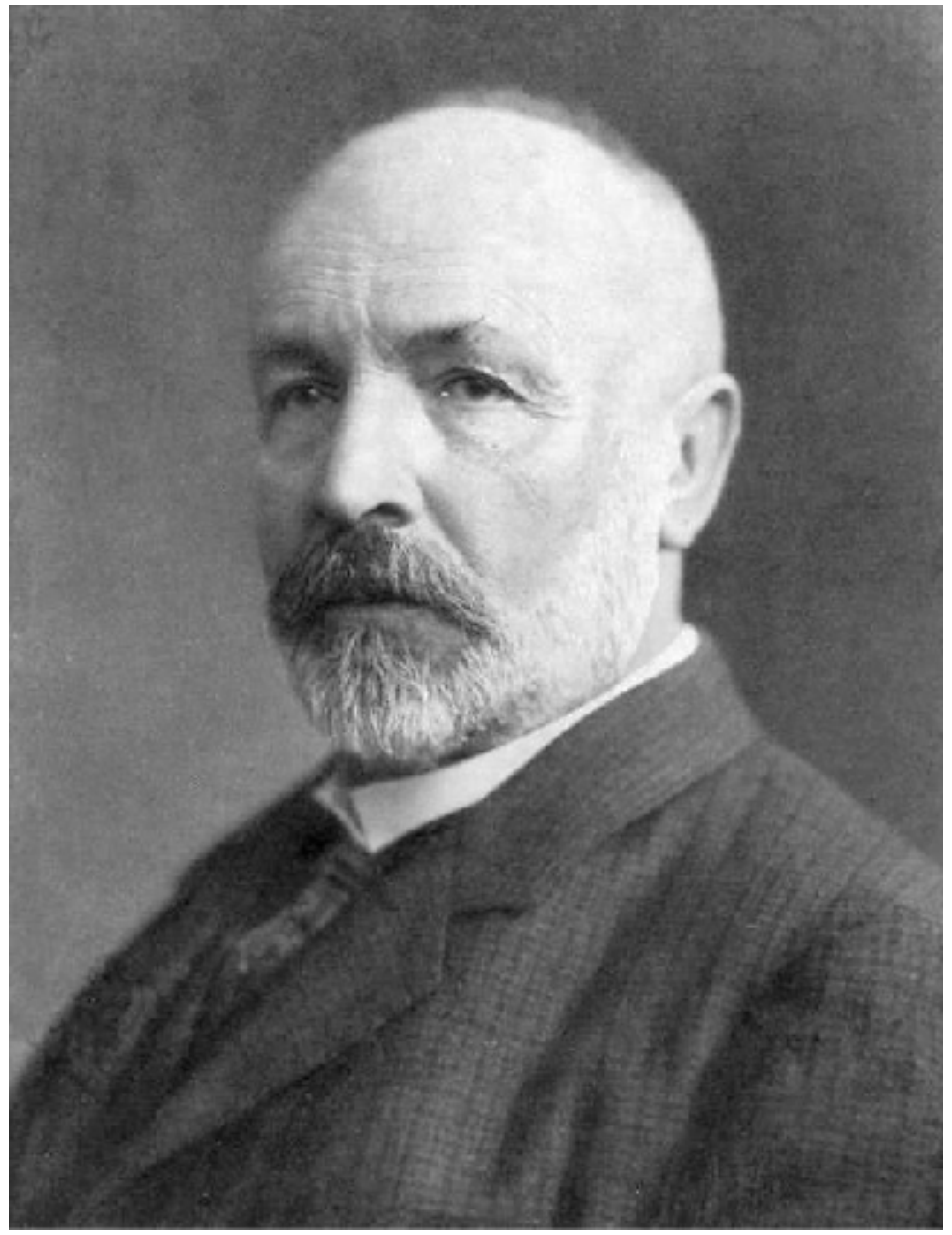
2. Es gibt genau eine Menge, die **keine Elemente** hat. Diese Menge heißt **leere Menge** und wird mit $\varnothing$ bezeichnet.

3. Jede endliche oder unendliche Sammlung von Mengen kann zu einer Menge zusammengefasst werden. Die Menge, die die Mengen $x_1,\ldots,x_n$ **zusammenfasst,** bezeichnen wir mit $\boldsymbol{\{ x_1,\ldots,x_n \}}$**.**

4. **Gleichheit:** Zwei Mengen x und y sind genau dann **gleich** $(x = y)$, wenn **jedes Element** von x ein Element von y ist und **jedes Element** von y ein Element von x ist.

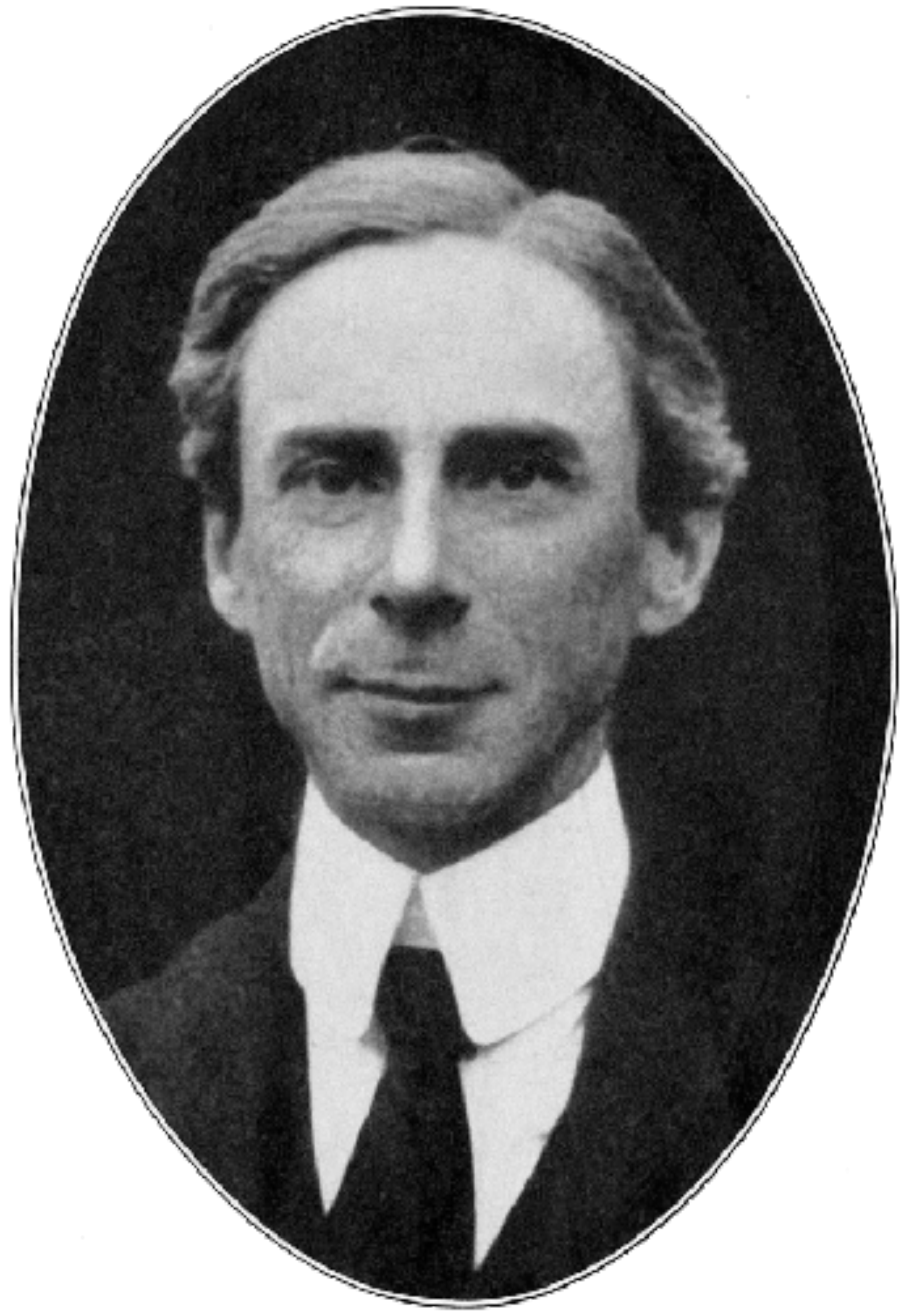
5. **Wohlfundiertheit:** Es gibt keine unendliche Folge $x_1, x_2, x_3, \ldots$ von Mengen, sodass $\ldots \in x_3 \in x_2 \in x_1$ gilt.

„Unter einer **Menge** verstehen wir
jede Zusammenfassung
von bestimmten wohlunterschiedenen Objekten
unserer Anschauung oder unseres Denkens
zu einem Ganzen."
(1895)



Georg Cantor
(1845-1918)

Russellsche Antinomie



Bertrand Russell
(1872-1970)

- **Russellsche Klasse:** $R = \{ x \mid x \notin x \}$
- **Enthält *R* sich selbst?**
- **Angenommen *R* enthält sich selbst**,
  dann gilt aufgrund der Definition,
  dass *R* sich **nicht** enthält,
  was der Annahme widerspricht.
- **Angenommen *R* enthält sich selbst nicht**,
  dann erfüllt *R* die Definition,
  so dass *R* sich **doch** selbst enthält,
  was wiederum der Annahme widerspricht.

Wohlfundiertheit

- **Wohlfundiertheit:**
  Es gibt **keine unendliche Folge $x_1, x_2, x_3, \ldots$** von Mengen, sodass $\ldots \in x_3 \in x_2 \in x_1$ gilt.

John von Neumann
(1903-1957)

- In der **Baumdarstellung** einer Menge kann man **nicht unendlich oft absteigen.**
- Nach **endlich vielen Abstiegen** erreicht man stets die leere Menge.
- Eine **Menge** kann erst gebildet werden **"nachdem"** ihre **Elemente** gebildet sind.

Frage

Welche der folgenden Aussagen sind korrekt?

- Für jede Menge X existiert eine Menge Y so dass X eine echte Obermenge von Y ist. ✗
- Für jede Menge X existiert eine Menge Y so dass Y eine echte Obermenge von X ist. ✓
- Es gibt eine Menge, die sich selbst enthält. ✗
- Es gibt eine Menge, die alle Mengen enthält. ✗

Mathematische Objekte als reine Mengen

- **Boolesche Werte:**
  false: $\emptyset$
  true: $\{\emptyset\}$

- **Natürliche Zahlen:**

0: $\emptyset$	2: $\{\{\emptyset\}\}$	4: $\{\{\{\{\emptyset\}\}\}\}$
1: $\{\emptyset\}$	3: $\{\{\{\emptyset\}\}\}$	$\ldots$

- **Zahlenmengen:** $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{R}$

$\mathbb{B} := \{0, 1\}$	**Boolesche Werte**
$\mathbb{N} := \{0, 1, 2, \ldots\}$	**Natürliche Zahlen**
$\mathbb{N}_+ := \{1, 2, 3, \ldots\}$	**Positive natürliche Zahlen**
$\mathbb{Z} := \{\ldots, -2, -1, 0, 1, 2, \ldots\}$	**Ganze Zahlen**
$\mathbb{R} :=$ Menge der reellen Zahlen	**Reelle Zahlen**

- **Paare:**
  (x,y): $\{\{x\},\{x,y\}\}$

Tupel

$$\langle x_1,\dots,x_n\rangle := \{(1,x_1),\dots,(n,x_n)\} \qquad \text{für } n \geq 0$$

- **Beispiele:**

$$\langle\rangle = \emptyset \qquad \textbf{leeres Tupel}$$

$$\langle x\rangle = \{(1,x)\}$$

$$\langle x,y\rangle = \{(1,x),\ (2,y)\}$$

$$\langle x,y,z\rangle = \{(1,x),\ (2,y),\ (3,z)\}$$

- **Länge, Komponenten, Positionen:**

$$|\langle x_1,\dots,x_n\rangle| := n \qquad \textbf{Länge}$$

$$\mathit{Com}\,\langle x_1,\dots,x_n\rangle := \{x_1,\dots,x_n\} \qquad \textbf{Komponenten}$$

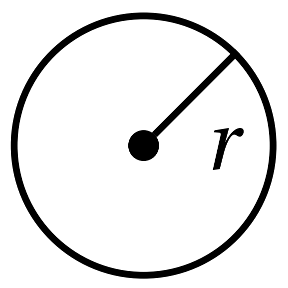
$$\mathit{Pos}\,\langle x_1,\dots,x_n\rangle := \{1,\dots,n\} \qquad \textbf{Positionen}$$

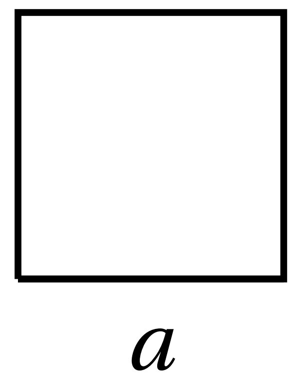
- **Menge aller Tupel:**

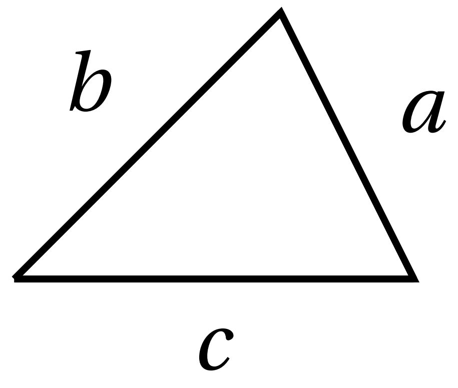
$$X^* := \{\, t \mid t \text{ Tupel mit } \mathit{Com}\, t \subseteq X \,\}$$

Interne Darstellung von Konstruktortypen

- **Interne Darstellung** als **Paare:**








```
Circle r        Square a        Triangle (a,b,c)
 (1,r)           (2,a)           (3, (a,b,c))
```

- Die erste Komponente der Paare ist die **Variantennummer**, die zweite das **Datum**.

Konstruktortypen, Listen, Bäume

- **Konstruktortypen: Summe von Mengen**

$$+\langle X_1,\dots,X_n\rangle \;:=\; \{\langle i,x\rangle \mid i\in\{1,\dots,n\} \text{ und } x\in X_i\} \qquad \text{für } n\geq 0$$

$$X_1+\dots+X_n \;:=\; +\langle X_1,\dots,X_n\rangle \qquad \text{für } n\geq 2$$

Beispiel:

$$\{3,4\}+\{5,6\}=\{\langle 1,3\rangle,\ \langle 1,4\rangle,\ \langle 2,5\rangle,\ \langle 2,6\rangle\}$$

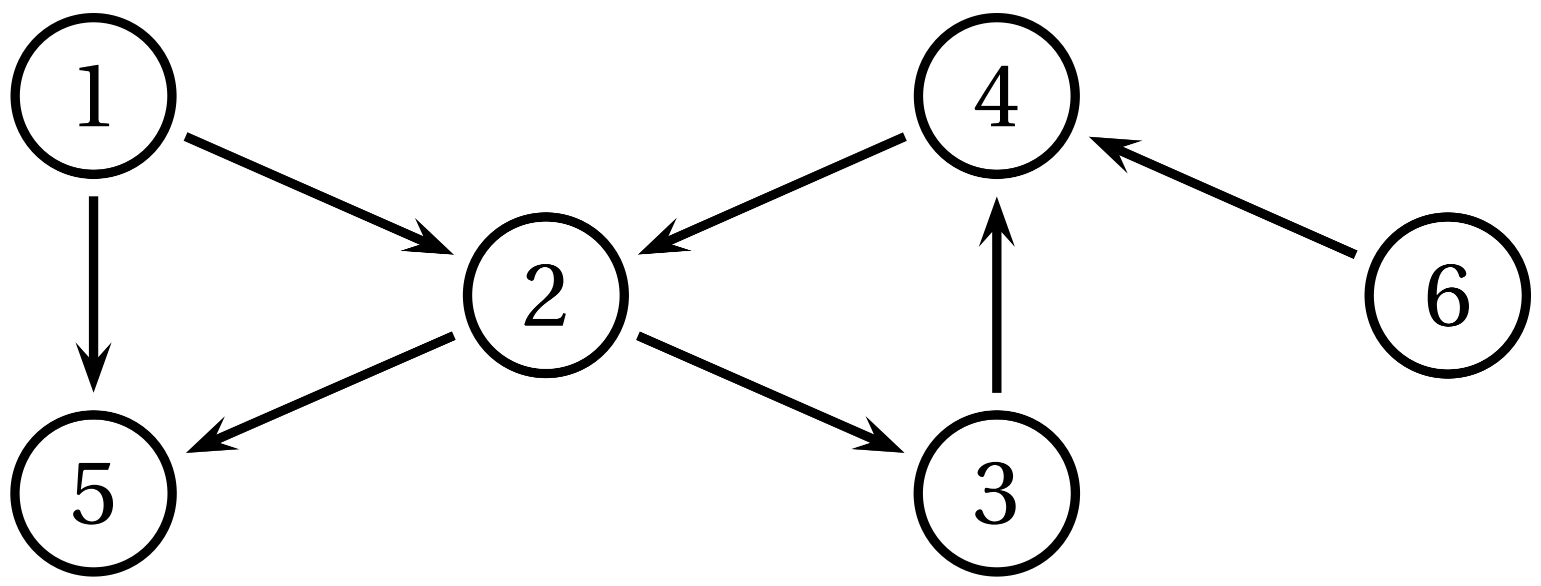
- **Listen** über *X*:

$$\mathscr{L}(X) \;:=\; \{\langle\rangle\}\cup(X\times\mathscr{L}(X))$$

- **Bäume** über *X*:

$$\mathscr{T}(X) \;:=\; X\times\mathscr{L}(\mathscr{T}(X))$$

Gerichtete Graphen



$$V = \{1, 2, 3, 4, 5, 6\}$$

$$E = \{(1,2),\ (1,5),\ (2,3),\ (2,5),\ (3,4),\ (4,2),\ (6,4)\}$$

Ein **(gerichteter) Graph** ist ein Paar (V,E) bestehend aus

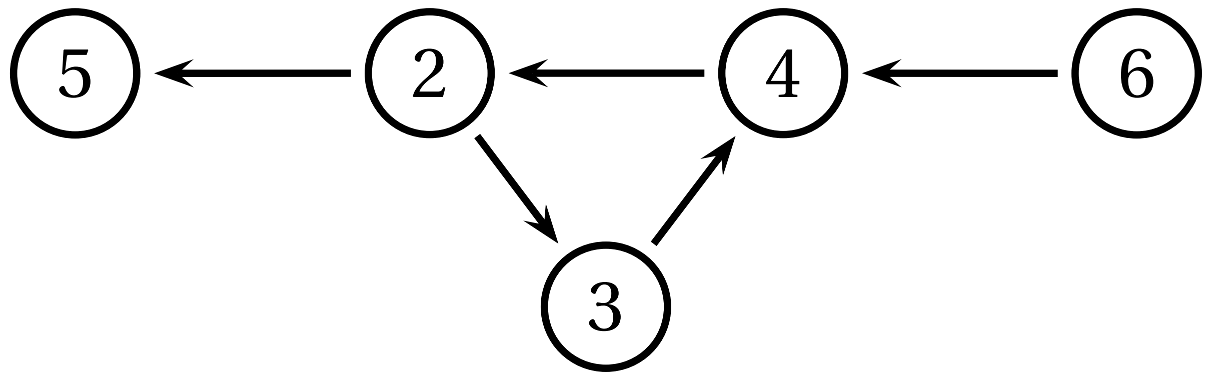
- einer Menge V von **Knoten** (engl. **vertices**) und
- einer Menge $E \subseteq V \times V$ von **Kanten** (engl. **edges**)

Binäre Relationen

- Eine **(binäre) Relation** ist eine Menge von Paaren.
- Für eine Relation *R* definieren wir den **Definitionsbereich** (engl. **domain**), **Wertebereich** (engl. **range**) und die **Knoten** (engl. **vertices**) wie folgt:

$$\begin{aligned} \mathit{Dom}\, R &:= \{x \mid \exists y\colon (x,y) \in R\} && \textbf{Definitionsbereich} \\ \mathit{Ran}\, R &:= \{y \mid \exists x\colon (x,y) \in R\} && \textbf{Wertebereich} \\ \mathit{Ver}\, R &:= \mathit{Dom}\, R \cup \mathit{Ran}\, R && \textbf{Knotenmenge} \end{aligned}$$

- Der Graph (*Ver R, R*) ist der **zu *R* gehörige Graph.**



$R = \{(2,3),\ (2,5),\ (3,4),\ (4,2),\ (6,4)\}$

$\mathit{Dom}\, R = \{2, 3, 4, 6\}$ $\quad$ $\mathit{Ran}\, R = \{2, 3, 4, 5\}$ $\quad$ $\mathit{Ver}\, R = \{2, 3, 4, 5, 6\}$

Funktionen

- Eine Relation *R* heißt **funktional**
  wenn es zu jedem $x \in$ *Dom R* **genau ein** $y \in$ *Ran R* gibt mit $(x,y) \in R$.

- Eine **Funktion** ist eine **funktionale Relation.**

- **Beispiel:**

$$f = \{(1,2),\ (3,4),\ (5,6),\ (7,8)\}$$

$$1 \longrightarrow 2 \qquad 3 \longrightarrow 4 \qquad 5 \longrightarrow 6 \qquad 7 \longrightarrow 8$$

Funktionsmengen

$X \rightharpoonup Y \;:=\; \{\, f \mid f \text{ Funktion mit } \mathit{Dom}\, f \subseteq X \text{ und } \mathit{Ran}\, f \subseteq Y \,\}$

Funktionen von X nach Y

$X \rightarrow Y \;:=\; \{\, f \mid f \text{ Funktion mit } \mathit{Dom}\, f = X \text{ und } \mathit{Ran}\, f \subseteq Y \,\}$

Totale Funktionen von X nach Y

$X \overset{\text{fin}}{\rightharpoonup} Y \;:=\; \{\, f \mid f \text{ endliche Funktion mit } \mathit{Dom}\, f \subseteq X \text{ und } \mathit{Ran}\, f \subseteq Y \,\}$

Endliche Funktionen von X nach Y

- **Beispiele:**

$$\mathbb{B} \rightarrow \mathbb{B} \;=\; \{\, \{\langle 0,0\rangle, \langle 1,0\rangle\},\; \{\langle 0,1\rangle, \langle 1,1\rangle\},\; \{\langle 0,1\rangle, \langle 1,0\rangle\},\; \{\langle 0,0\rangle, \langle 1,1\rangle\} \,\}$$

$$\mathbb{B} \rightharpoonup \mathbb{B} \;=\; (\mathbb{B} \rightarrow \mathbb{B}) \cup \{\, \emptyset,\; \{\langle 0,0\rangle\},\; \{\langle 0,1\rangle\},\; \{\langle 1,0\rangle\},\; \{\langle 1,1\rangle\}\}$$

$$\mathbb{B} \overset{\text{fin}}{\rightharpoonup} \mathbb{B} \;=\; \mathbb{B} \rightharpoonup \mathbb{B}$$

Darstellungen

- Eine **Darstellung** für eine Menge X ist eine Funktion f mit $Ran\ f = X$.
- Die Elemente von $Dom\ f$ sind **die Darstellungen für** die Elemente von X.
- **Injektive Darstellungen** heißen auch **eindeutige Darstellungen**.
  (**injektiv:** zu jedem $y \in Ran\ R$ gibt es **genau ein** $x \in Dom\ R$ mit $(x,y) \in R$.)
- **Beispiele:**

$B \in \mathscr{L}(X) \to X^*$	$\mathrm{Set} \in \mathscr{L}(X) \to \mathscr{P}(X)$
$B[x_1, \ldots, x_n] = \langle x_1, \ldots, x_n \rangle$	$\mathrm{Set}[x_1, \ldots, x_n] = \{x_1, \ldots, x_n\}$
eindeutige Darstellung	nicht eindeutige Darstellung

Frage

Welche der folgenden Relationen sind eindeutige Darstellungen für die Menge der Booleschen Werte {0, 1}?

- { (0, 5), (1, 6) } ✗
- { (5, 0), (6, 1) } ✓
- { (5, 0), (5, 1), (6, 1) } ✗
- { (5, 0), (6, 1), (7, 1) } ✗

Lambda-Notation

- **Lambda-Notation:** Beschreibung einer **Funktion** durch eine **Vorschrift**, die zu jedem Argument das gewünschte Ergebnis liefert.

- Entspricht **Abstraktionen** in ML.

$$\lambda\, n \in \mathbb{N}.\, n = \{(n, n) \mid n \in \mathbb{N}\} = Id(\mathbb{N})$$

$$\lambda\, n \in \mathbb{N}.\, 2n + n^3 = \{(n,\ 2n + n^3) \mid n \in \mathbb{N}\}$$

$$\lambda\, (x, y) \in \mathbb{Z}^2.\, x + y = \{((x, y),\ x + y) \mid x \in \mathbb{Z},\ y \in \mathbb{Z}\}$$

Alonzo Church
(1903-1995)

Kapitel 9
Mathematische Prozeduren

Mathematische Prozeduren

- Eine **(mathematische) Prozedur** ist eine Berechnungsvorschrift, die bei der Anwendung auf ein Argument ein Ergebnis liefert.
- **Mathematische Prozeduren sind mathematische Objekte, die unabhängig von einer Programmiersprache existieren.**
- Eine mathematische Prozedur ist gegeben durch
  - den **Namen** der Prozedur,
  - den **Argumentbereich,**
  - den **Ergebnisbereich,** und
  - eine oder mehrere, eventuell rekursive, **definierende Gleichungen.**

Beispiele

$$abs : \mathbb{Z} \rightarrow \mathbb{N}$$

$$abs\ x = \text{if } x < 0 \text{ then } -x \text{ else } x$$

$$fac : \mathbb{N} \rightarrow \mathbb{N}$$
$$fac\ 0 = 1$$
$$fac\ n = n \cdot fac(n-1) \quad \text{für } n > 0$$

$$fac' : \mathbb{Z} \rightarrow \mathbb{Z}$$
$$fac'\ x = \text{if } x = 0 \text{ then } 1 \text{ else } x \cdot fac'(x-1)$$

$$fib : \mathbb{N} \rightarrow \mathbb{N}$$
$$fib\ n = \text{if } n < 2 \text{ then } n \text{ else } fib(n-1) + fib(n-2)$$

Wohlgeformtheitsbedingungen

1. **Funktionen** werden nur auf Elemente ihres **Definitionsbereichs** angewendet.
2. **Rekursive Anwendungen** der Prozedur erfolgen nur auf Elemente des **Argumentbereichs** der Prozedur.
3. Es werden nur **Ergebnisse im Ergebnisbereich** der Prozedur geliefert.
4. Die definierenden Gleichungen sind **disjunkt** und **erschöpfend**

Wohlgeformtheit garantiert: Für jeden Wert aus dem Argumentbereich einer Prozedur gibt es genau eine anwendbare Gleichung.

Ausführung

- Durch Einsetzen eines Wertes in eine anwendbare **definierende Gleichung** (und anschließendes Vereinfachen) erhält man eine **Anwendungsgleichung.**

- **Beispiele:**

$$\begin{aligned} fac\, 5 &= 5 \cdot fac\, 4 \\ fac'(-3) &= -3 \cdot fac'(-4) \end{aligned}$$

$$\begin{aligned} &fac : \mathbb{N} \to \mathbb{N} \\ &fac\, 0 = 1 \\ &fac\, n = n \cdot fac(n-1) \quad \text{für } n > 0 \end{aligned}$$

$$\begin{aligned} &fac' : \mathbb{Z} \to \mathbb{Z} \\ &fac'\ x = \text{if } x = 0 \text{ then } 1 \text{ else } x \cdot fac'(x-1) \end{aligned}$$

Ausführungsprotokoll

- Durch wiederholtes Einsetzen der neuen Argumente, und Ersetzen der Prozeduranwendungen durch die entstehenden rechten Seiten, erhält man ein **Ausführungsprotokoll.**

- **Beispiel:**

$$\begin{aligned} \mathit{fib}\,3 &= \mathit{fib}\,2 + \mathit{fib}\,1 \\ &= (\mathit{fib}\,1 + \mathit{fib}\,0) + \mathit{fib}\,1 \\ &= (1 + \mathit{fib}\,0) + \mathit{fib}\,1 \\ &= (1 + 0) + \mathit{fib}\,1 \\ &= 1 + \mathit{fib}\,1 \\ &= 1 + 1 \\ &= 2 \end{aligned}$$

$$\begin{aligned} &\mathit{fib} : \mathbb{N} \to \mathbb{N} \\ &\mathit{fib}\ n = \text{if } n < 2 \text{ then } n \text{ else } \mathit{fib}(n-1) + \mathit{fib}(n-2) \end{aligned}$$

- Ein Ausführungsprotokoll heißt **terminierend,** wenn es mit einem **Wert endet.**

Terminierung

Sei p eine Prozedur $X \to Y$ und $x \in X$.
Wir sagen, dass die **Anwendung** $p\,x$

- **mit dem Ergebnis y terminiert** (auch: **für x mit y terminiert** oder **für x das Ergebnis y liefert**), wenn es ein Ausführungsprotokoll gibt, das mit $p\,x$ beginnt und mit y endet.

- **divergiert,** wenn es **kein terminierendes** Ausführungsprotokoll gibt, das mit $p\,x$ beginnt.

Der **Definitionsbereich** einer Prozedur p ist die Menge der **terminierenden Argumente:**

$$Dom\,p := \{x \in X \mid p \text{ terminiert für } x\}$$

Frage

Geben Sie den Argument- und den Definitionsbereich der folgenden Prozedur p an:

p: $\mathbb{N} \to \mathbb{N}$, p n = p n

- {}, {} ✗
- $\mathbb{N}$, {} ✓
- {}, $\mathbb{N}$ ✗
- $\mathbb{N}$, $\mathbb{N}$ ✗

Rekursionsfunktionen

- Die **Folgeargumente** einer **Prozedur** *p* für ein **Argument** *x* sind die Argumente $x_1,\ldots,x_n$, für die **rekursive Aufrufe** von *p* erforderlich sind, um das Ergebnis für *x* zu bestimmen.

- **Beispiele:**

Bei *fac* hat 4 das Folgeargument 3.

$$
\begin{aligned}
& \mathit{fac} : \mathbb{N} \to \mathbb{N} \\
& \mathit{fac}\, 0 = 1 \\
& \mathit{fac}\, n = n \cdot \mathit{fac}(n-1) \quad \text{für } n > 0
\end{aligned}
$$

Bei *fib* hat 4 die Folgeargumente 3 und 2.

$$
\begin{aligned}
& \mathit{fib} : \mathbb{N} \to \mathbb{N} \\
& \mathit{fib}\, n = \text{if } n < 2 \text{ then } n \text{ else } \mathit{fib}(n-1) + \mathit{fib}(n-2)
\end{aligned}
$$

Rekursionsfunktionen

- Die **Rekursionsfunktion** einer **Prozedur** $p: X \to Y$ ist eine **totale Funktion** $r: X \to X^*$, die jedes **Argument** auf das Tupel seiner **Folgeargumente** abbildet.
- **Beispiele:**

$$
\begin{aligned}
abs &: \quad \lambda n \in \mathbb{Z}. \langle\rangle \\
fac &: \quad \lambda n \in \mathbb{N}.\ \text{if } n = 0 \text{ then } \langle\rangle \text{ else } \langle n-1 \rangle \\
fib &: \quad \lambda n \in \mathbb{N}.\ \text{if } n < 2 \text{ then } \langle\rangle \text{ else } \langle n-1,\ n-2 \rangle
\end{aligned}
$$

$$
\begin{aligned}
&abs : \mathbb{Z} \to \mathbb{N} \\
&abs\ x = \text{if } x < 0 \text{ then } -x \text{ else } x
\end{aligned}
$$

$$
\begin{aligned}
&fac : \mathbb{N} \to \mathbb{N} \\
&fac\ 0 = 1 \\
&fac\ n = n \cdot fac(n-1) \quad \text{für } n > 0
\end{aligned}
$$

$$
\begin{aligned}
&fib : \mathbb{N} \to \mathbb{N} \\
&fib\ n = \text{if } n < 2 \text{ then } n \text{ else } fib(n-1) + fib(n-2)
\end{aligned}
$$

Rekursionsarten

- Eine Prozedur $p: X \to Y$ heißt **rekursiv,** wenn es **mindestens** ein Argument $x \in X$ gibt, für das $r\,x$ **nicht leer** ist.
- p heißt **linear-rekursiv**, wenn p **rekursiv** ist und $r\,x$ für **alle** Argumente $x \in X$ **höchstens eine** Position hat.
- p heißt **baumrekursiv**, wenn
  es **mindestens ein** Argument $x \in X$ gibt, für das
  $r\,x$ **zwei oder mehr** Positionen hat.

- **Beispiele:**
  abs ist nicht rekursiv
  fac ist linear-rekursiv

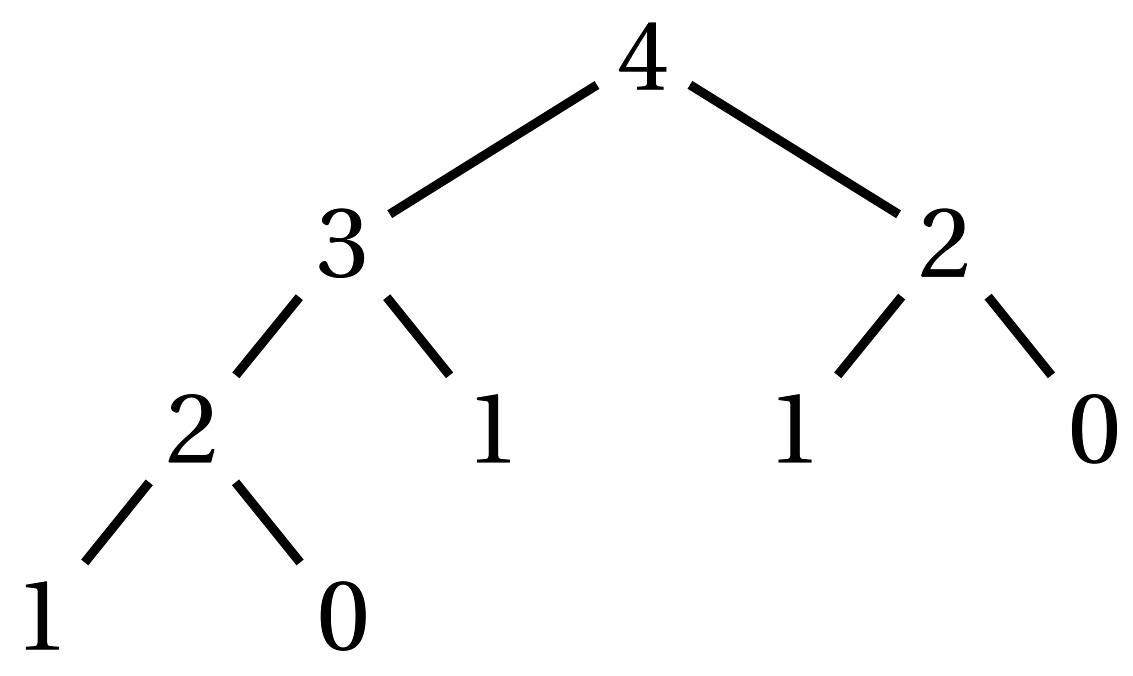
Rekursionsbaum

- Der **Rekursionsbaum** einer Prozedur für ein Argument x
  hat den **Kopf** x
  und die Rekursionsbäume der Folgeargumente als **Unterbäume.**

- $rb\colon X \to \mathscr{T}(X)$
  $rb\;x = (x, [rb\;x_1, \ldots, rb\;x_n])$ für $Com(r\;x) = \{x_1, \ldots, x_n\}$

- **Beispiel:**

  $fib : \mathbb{N} \to \mathbb{N}$
  $fib\;n = \text{if } n < 2 \text{ then } n \text{ else } fib(n-1) + fib(n-2)$



- Die **Rekursionstiefe** eines terminierenden Arguments x
  einer Prozedur ist die **Tiefe des Rekursionsbaums** für x.

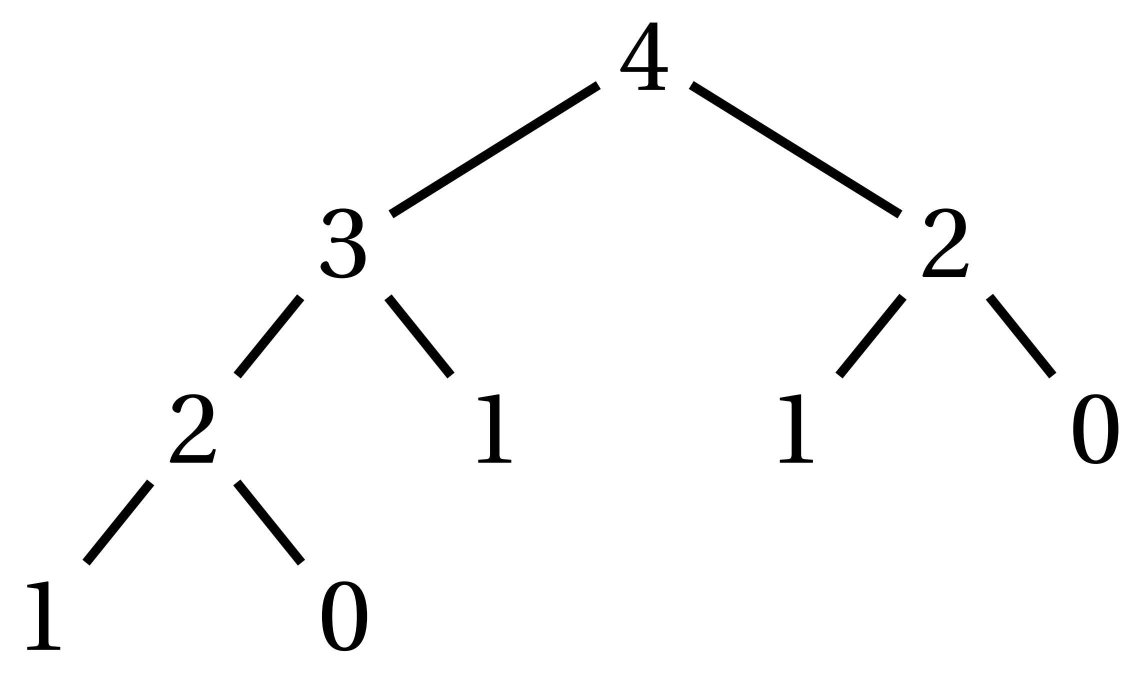
Rekursionsrelation

- Ein **Rekursionsschritt** einer Prozedur ist ein Paar (*x*,*x′*) das aus einem **Argument** *x* und einem **Folgeargument** *x′* of *x* besteht.

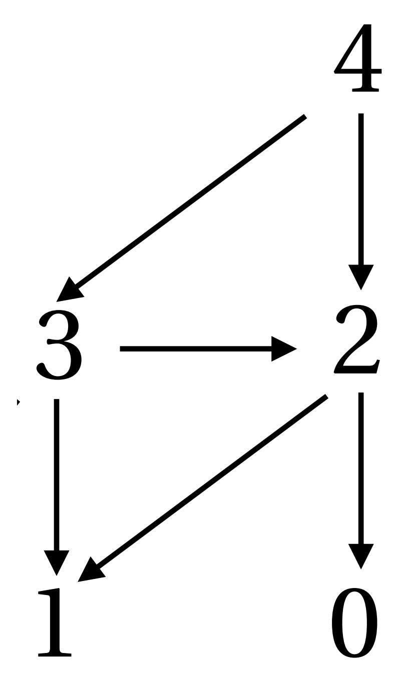
- Die Menge aller Rekursionsschritte einer Prozedur ist die **Rekursionsrelation** der Prozedur.

- Die Rekursionsrelation ergibt sich aus der **Rekursionsfunktion**:

$$R = \{\, (x, x') \mid x \in \mathit{Dom}\, r \;\wedge\; x' \in \mathit{Com}\, (r\, x) \,\}$$



Rekursionsbaum



Rekursionsrelation

Beispiele

$$
\begin{aligned}
abs &: \emptyset \\
fac &: \{(n, n-1) \mid n \in \mathbb{N},\ n > 0\} \\
fib &: \{(n, n') \in \mathbb{N}^2 \mid 0 \le n-2 \le n' \le n-1\}
\end{aligned}
$$

$$abs : \mathbb{Z} \to \mathbb{N}$$

$$abs\ x = \text{if } x < 0 \text{ then } -x \text{ else } x$$

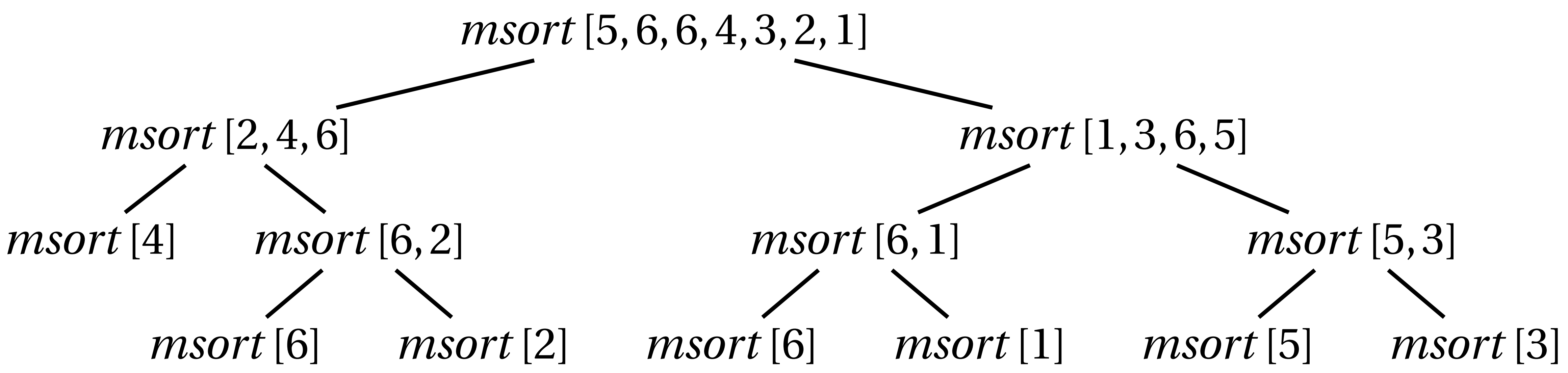
$$
\begin{aligned}
&fac : \mathbb{N} \to \mathbb{N} \\
&fac\,0 = 1 \\
&fac\ n = n \cdot fac(n-1) \quad \text{für } n > 0
\end{aligned}
$$

$$
\begin{aligned}
&fib : \mathbb{N} \to \mathbb{N} \\
&fib\ n = \text{if } n < 2 \text{ then } n \text{ else } fib(n-1) + fib(n-2)
\end{aligned}
$$

Terminierung

$$nil @ ys = ys$$
$$(x :: xr) @ ys = x :: (xr @ ys)$$

bestimmen. Die durch die zweite Gleichung eingeführte Rekursion terminiert, da die Länge der linken Liste jeweils um eins reduziert wird. Die Umsetzung der Gleichungen



Die Rekursion von *msort* terminiert, da die von *split* gelieferten Teillisten kürzer als die Ausgangslisten sind. Für die gute Laufzeit von *msort* ist es wesentlich, dass die von *split*

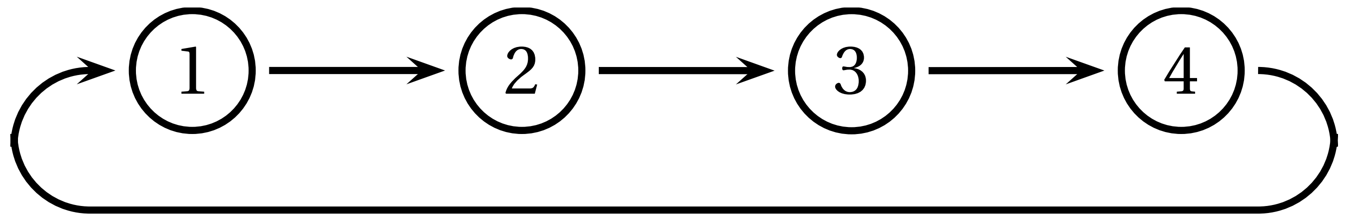
```
fun subexps e = e::
    (case e of
          A(e1,e2) => subexps e1 @ subexps e2
        | M(e1,e2) => subexps e1 @ subexps e2
        |    _     => nil)
```

val subexps : exp → exp list

Beachten Sie, dass die Prozedur *subexps* binär rekursiv ist. Die Terminierung von *subexps* ergibt sich aus der Tatsache, dass die rekursiven Anwendungen auf die Komponenten des aktuellen Ausdrucks erfolgen. Also werden mit fortschreitender Rekursion immer kleinere Ausdrücke behandelt. Machen Sie sich klar, dass der Rekursionsbaum eines

Terminierende Relationen

- Eine Relation R heißt **fortschreitend,** wenn sie **nicht leer** ist und es für jeden Knoten $x \in Ver\ R$ einen R-Nachfolger gibt.



fortschreitend

- Eine Relation heißt **terminierend** (auch: die Relation **terminiert**), wenn sie **keine fortschreitende** Relation enthält.

- Eine Relation R **terminiert für ein Objekt** x, wenn es keine fortschreitende Relation $R' \subseteq R$ mit $x \in Ver\ R'$ gibt.

Terminierende Relationen

- Eine **fortschreitende Relation** terminiert für **keinen** ihrer Knoten.

- Eine Relation **terminiert** genau dann,
  wenn sie **für jeden ihrer Knoten terminiert.**

- Eine **endliche** Relation **terminiert** genau dann,
  wenn sie **azyklisch** ist.
  không xoay vòng

- **Beispiel** für eine unendliche terminierende Relation:

$$Ter := \{(x, y) \in \mathbb{N}^2 \mid x > y\}$$

www.prog1.saarland