Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

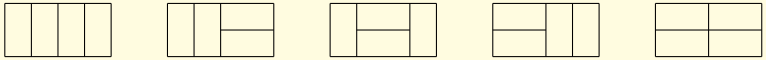
Teil III

Werte

10. Knobelaufgabe #3

Bob der Baumeister will eine 100 m breite und 2 m hohe Mauer bauen. Als Baumaterial hat er 2 m breite und 1 m hohe Quader zur Verfügung.

Wieviele Möglichkeiten gibt es die Mauer zu konstruieren?



Eine 4 m breite Mauer lässt sich zum Beispiel auf 5 Arten zusammensetzen.

Wie erhöht sich die Zahl der Möglichkeiten, wenn die Mauer statt 2 m sogar 3 m hoch werden soll?

10. Gliederung

11 Natürliche Zahlen

12 Boolesche Werte

13 Wertedefinitionen

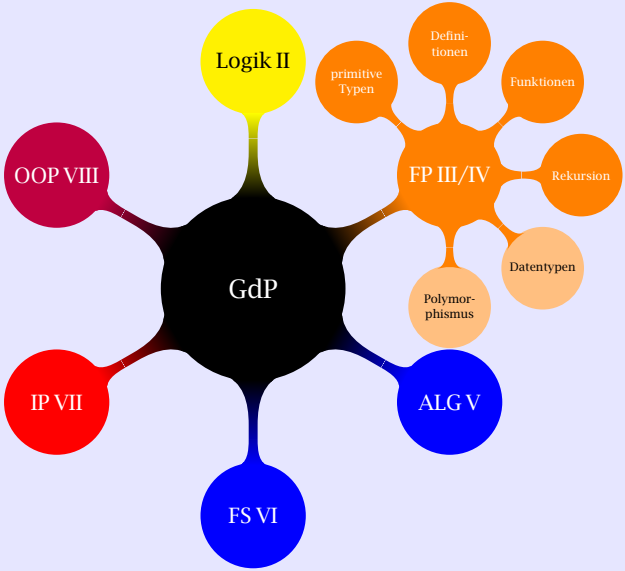
14 Funktionsdefinitionen

15 Funktionsausdrücke

16 Rekursive Funktionen

17 Entwurfsmuster

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster



10. Lernziele

Nach Durcharbeitung dieses Kapitels sollten Sie

- den Unterschied zwischen statischer und dynamischer Semantik kennen,
- Sinn und Zweck von Typsystemen verstanden haben,
- das Konzept von Bindungen verstanden haben,
- Funktionen und Rekursion verstanden haben,
- Entwurfsmuster kennen und anwenden können,
- Mini–F# Programme lesen und selbst schreiben können.

10. Semantik

Semantics is a strange kind of applied mathematics; it seeks profound definitions rather than difficult theorems. The mathematical concepts which are relevant are immediately relevant. Without any long chains of reasoning, the application of such concepts directly reveals regularity in linguistic behaviour, and strengthens and objectifies our intuitions of simplicity and uniformity.

— J.C. Reynolds (1980)

10. Growing a language …

I think you know what a man is. A woman is more or less like a man, but not of the same sex. (This may seem like a strange thing for me to start with, but soon you will see why.)

Next, I shall say that a person is a woman or a man (young or old).

To keep things short, when I say «he« I mean «he or she«, and when I say «his« I mean «his or her.«

A machine is a thing that can do a task with no help, or not much help, from a person.

— Growing a Language, Guy L. Steele Jr.

10. Struktur der folgenden Abschnitte

- Motivation
- Abstrakte Syntax

$e \in \mathsf{Expr} ::= \cdots$ ***Ausdrücke***

- Statische Semantik

$t \in \mathsf{Type} ::= \cdots$ ***Typen***

- Dynamische Semantik

$v \in \mathsf{Val} ::= \cdots$ ***Werte***

- Vertiefung
- Blick über den Tellerrand

10. Statische Semantik — Sinn und Zweck

- Ein Programm in Mini-F# ist ein ***Ausdruck***:
  - $4711 + 815$
  - `"Hello, world!"`
- Ausdrücke sind beliebig kombinierbar.
- Nicht alle Kombinationen machen jedoch Sinn:
  - `"Hello, world!"` $* 4711$
- Die ***statische Semantik*** fängt diese sinnlosen Ausdrücke ab.
- Zu diesem Zweck werden Ausdrücke mit Hilfe von ***Typen*** in Schubladen eingeteilt:
  - `"Hello, world!"` hat den Typ *String*
  - 4711 hat den Typ *Nat*
- Die Multiplikation arbeitet auf den natürlichen Zahlen: `"Hello, world!"` $* 4711$ ist nicht wohlgetypt.

10. Statische Semantik — Typregeln

Die ***statische Semantik*** legt fest, dass die Multiplikation zwei natürliche Zahlen nimmt und eine natürliche Zahl zum Ergebnis hat.

$$\frac{e_1 : \mathit{Nat} \qquad e_2 : \mathit{Nat}}{e_1 * e_2 : \mathit{Nat}}$$

Für jedes neu eingeführte Konstrukt wird eine solche ***Typregel*** angegeben. Diese Beweisregeln spezifizieren die zweistellige Relation

$$e : t$$

zwischen Ausdrücken und Typen. *Lies:* »e hat den Typ t«.

10. Statische Semantik — Begriffe

- Die statische Semantik hat eine ordnende Funktion: fast alle Abschnitte führen *einen* neuen Typ ein, zusammen mit Sprachkonstrukten, die auf diesem Typ arbeiten.
- Ein Ausdruck e heißt ***wohlgetypt***, wenn es einen Typ t gibt, so dass sich $e : t$ mit den Regeln der statischen Semantik ableiten lässt.
- Wohlgetypte Ausdrücke können ausgerechnet werden.

10. Dynamische Semantik — Auswertungsregeln

Wie ein Ausdruck ausgerechnet wird, legt die ***dynamische Semantik*** fest. Für die Multiplikation: beide Argumente werden ausgerechnet, das Ergebnis ist das Produkt der Teilergebnisse.

$$\frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1 * e_2 \Downarrow n_1 \cdot n_2}$$

Für jedes Konstrukt wird mindestens eine ***Auswertungsregel*** angegeben. Diese Regeln spezifizieren die zweistellige Relation

$$e \Downarrow v$$

zwischen Ausdrücken und Werten. *Lies:* »e wertet zu v aus«.

10. Dynamische Semantik — Werte

- Was ist ein Wert?
- Ein Wert ist das Ergebnis eines Programms; der Wert eines arithmetischen Ausdrucks ist zum Beispiel eine natürliche Zahl.
- Mit jedem neu eingeführten Typ werden wir auch den Bereich der Werte erweitern.
- Ein Typ ist im Prinzip die Menge aller zugehörigen Werte.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

10. Dynamische Semantik — Beweisbäume

Ein Programm wird ausgerechnet, indem die Auswertungsregeln für die Teilausdrücke zu einem Beweisbaum kombiniert werden.

$$\dfrac{\dfrac{\overline{4711 \Downarrow 4711} \qquad \overline{2 \Downarrow 2}}{4711 * 2 \Downarrow 9422} \qquad \overline{815 \Downarrow 815}}{4711 * 2 + 815 \Downarrow 10237}$$

Konstanten wie 4711 oder 815 werten zu sich selbst aus — Konstanten *sind* Werte.

11. Motivation

Die ganze Zahl schuf der liebe Gott,
alles übrige ist Menschenwerk.
— Leopold Kronecker (1823–1891)

- Den Begriff Rechnen werden die meisten mit Zahlen in Verbindung bringen.
- In diesem Abschnitt führen wir einige elementare Konstrukte zum Rechnen mit *natürlichen* Zahlen ein.

Natürliche Zahlen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Boolesche Werte
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

11. Abstrakte Syntax

$n \in \mathbb{N}$	$::= 0 \mid 1 \mid 2 \mid \cdots$	***natürliche Zahlen***
$e \in \mathsf{Expr}$	$::=$	***Arithmetische Ausdrücke:***
	$\mid n$	natürliche Zahl
	$\mid e_1 + e_2$	Addition
	$\mid e_1 \dot{-} e_2$	natürliche Subtraktion (»monus«)
	$\mid e_1 * e_2$	Multiplikation
	$\mid e_1 \div e_2$	natürliche Division
	$\mid e_1 \,\%\, e_2$	Divisionsrest

11. Statische Semantik

$t \in \mathsf{Type} ::=$	***Typen:***
$\quad \mid \mathit{Nat}$	Typ der natürlichen Zahlen

Typregeln:

$$\frac{}{n : \mathit{Nat}}$$

$$\frac{e_1 : \mathit{Nat} \qquad e_2 : \mathit{Nat}}{e_1 + e_2 : \mathit{Nat}} \quad \text{usw.}$$

Natürliche Zahlen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Boolesche Werte
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

11. Dynamische Semantik

$v \in \mathsf{Val} ::=$	***Werte:***
$\quad \mid n$	natürliche Zahlen

Auswertungsregeln:

$$\frac{}{n \Downarrow n}$$

$$\frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1 + e_2 \Downarrow n_1 + n_2}$$

$$\frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1 * e_2 \Downarrow n_1 \cdot n_2}$$

Natürliche Zahlen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Boolesche Werte
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

11. Dynamische Semantik

Ist doch alles ziemlich offensichtlich, oder? Ich meine, es ist doch klar, dass '$*$' beide Argumente ausrechnet und die Ergebnisse dann multipliziert.

Was ist denn, wenn eins der Argumente 0 ist?

Na ja, dann ist das Ergebnis halt auch 0. Ich sehe nicht, worauf Du hinaus willst.

Wenn das erste Argument schon 0 ist, müssen wir das zweite Argument ja gar nicht mehr ausrechnen!

$$\frac{e_1 \Downarrow 0}{e_1 * e_2 \Downarrow 0}$$

Ok, ist vielleicht schneller, aber 0 kommt immer noch raus.

11. Dynamische Semantik — Subtraktion

Die Differenz zweier Ausdrücke ist 0, wenn das zweite Argument größer ist als das erste.

$$\frac{e_1 \Downarrow n+k \qquad e_2 \Downarrow n}{e_1 \dot{-} e_2 \Downarrow k} \qquad \frac{e_1 \Downarrow n \qquad e_2 \Downarrow n+k}{e_1 \dot{-} e_2 \Downarrow 0}$$

☞ Mini-F# arbeitet auf den natürlichen und nicht auf den ganzen Zahlen. (F# kennt von Haus aus keine natürlichen Zahlen, dafür aber ganze Zahlen und Fließkommazahlen. In Kürze mehr dazu.)

Der Operator $\dot{-}$ hört auch auf den Namen »monus«.

Natürliche Zahlen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Boolesche Werte
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

11. Dynamische Semantik — Division

Die Operatoren '÷' und '%' implementieren die Division mit Rest: $a \div b$ ist der Quotient von a und b und $a \,\%\, b$ ist der Divisionsrest.

$$\frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1 \div e_2 \Downarrow q} \qquad n_1 = q \cdot n_2 + r \text{ und } r < n_2$$

$$\frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{e_1 \,\%\, e_2 \Downarrow r} \qquad n_1 = q \cdot n_2 + r \text{ und } r < n_2$$

☞ Die Nebenbedingungen legen die Metavariablen $q, r \in \mathbb{N}$ eindeutig fest.

☞ Für $b > 0$ gilt stets

$$a = (a \div b) * b + (a \,\%\, b) \quad \text{und} \quad 0 \leqslant a \,\%\, b < b$$

☞ Jede Zahl a lässt sich eindeutig in einen Quotienten und in einen Rest zerlegen für ein festes $b > 0$.

11. Dynamische Semantik — Division

- Die Ausdrücke $e \div 0$ und $e \,\%\, 0$ sind undefiniert; die dynamische Semantik ordnet ihnen keinen Wert zu: es gibt kein r mit $r < 0$.
- Das ist unbefriedigend — die statische Semantik sollte ja gerade derartige Programme herausfiltern.
- Eine Lösung für dieses Problem stellen wir erst *sehr* viel später vor (in Teil VII).

Natürliche Zahlen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Boolesche Werte
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

11. Dynamische Semantik

Ich habe noch mal über die Multiplikation nachgedacht. Wenn wir die Regel

$$\frac{e_1 \Downarrow 0}{e_1 * e_2 \Downarrow 0}$$

hinzunehmen, dann können wir Programme auswerten, die sonst keinen Wert haben.

Ähem, wie meinst Du das? Die statische Semantik, oder wie das heißt, lässt sowas doch gar nicht zu.

Hast Du nicht aufgepasst ;-)? Der Ausdruck $0 \div 0$ hat zum Beispiel keinen Wert.

Ja ...

Na, dann hat $0 * (0 \div 0)$ auch keinen Wert. Aber *mit* meiner Regel können wir den Ausdruck zu 0 auswerten.

Natürliche Zahlen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Boolesche Werte
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

11. Demo

```
⋙ 4711 * 2 + 815
10237
⋙ 11 * 11
121
⋙ 111 * 111
12321
⋙ 111111111 * 111111111
12345678987654321
```

☞ Die Ergebnisse sind stets exakt. (Die Genauigkeit ist *nicht* auf die native Genauigkeit von Rechnern, 32 oder 64 Bit, eingeschränkt.)

Natürliche Zahlen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Boolesche Werte
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

11. Über den Tellerrand: F#

F# kennt eine Reihe verschiedener Zahlentypen (*Nat* ist leider nicht vordefiniert):

- *byte*: natürliche Zahlen von 0 bis $2^8 - 1 = 255$;
- *sbyte*: ganze Zahlen von $-2^7 = -128$ bis $2^7 - 1 = 127$ (signed byte);
- *int16*: ganze Zahlen von $-2^{15} = -32768$ bis $2^{15} - 1 = 32767$;
- *uint16*: natürliche Zahlen von 0 bis $2^{16} - 1 = 65535$ (unsigned int16);
- *int* und *uint32*: dito mit 32 Bit;
- *int64* und *uint64*: dito mit 64 Bit;
- *float32*: 32-bit Fließkommazahlen;
- *float*: 64-bit Fließkommazahlen;
- *bigint*: ganze Zahlen.

☞ Rechnen mit beschränkter Genauigkeit hat seine Tücken: $255uy + 1uy = 0uy$ und $2147483647 + 1 = -2147483648$.

12. Motivation

Nicht-triviale Programme treffen viele Entscheidungen. Im einfachsten Fall wird geprüft, ob ein bestimmter Sachverhalt wahr oder falsch ist:

- Ist das Konto überzogen?
- Ist Florian größer als Lisa?

Das Ergebnis einer solchen Überprüfung repräsentieren wir durch einen Wahrheitswert: *true* oder *false*.

Natürliche Zahlen
Boolesche Werte
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

12. Motivation

In Abhängigkeit von einem Wahrheitswert kann die Rechnung dann einen bestimmten Verlauf nehmen.

if e_1 ***then*** e_2 ***else*** e_3

Wertet der Ausdruck e_1, die Bedingung, zu *true* aus, dann wird mit der Auswertung von e_2 fortgefahren; wertet e_1 zu *false* aus, dann wird e_3 ausgewertet.

Natürliche Zahlen

Boolesche Werte

Motivation

Abstrakte Syntax

Statische Semantik

Dynamische Semantik

Vertiefung

Werte/-definitionen

Funktions/-definitionen

Funktions-ausdrücke

Rekursion

Entwurfsmuster

12. Abstrakte Syntax

$e ::= \cdots$	***Boolesche Ausdrücke:***
$\mid$ *false*	falsch
$\mid$ *true*	wahr
$\mid$ ***if*** e_1 ***then*** e_2 ***else*** e_3	Alternative
$\mid$ $e_1 < e_2$	kleiner
$\mid$ $e_1 \leqslant e_2$	kleiner gleich
$\mid$ $e_1 = e_2$	gleich
$\mid$ $e_1 <> e_2$	ungleich
$\mid$ $e_1 \geqslant e_2$	größer gleich
$\mid$ $e_1 > e_2$	größer

☞ Die Notation $e ::= \cdots$ soll andeuten, dass wir die Kategorie der Ausdrücke um Boolesche Ausdrücke und Vergleichsoperatoren *erweitern*.
Der Teilausdruck e_1 der Alternative heißt ***Bedingung***; die Teilausdrücke e_2 und e_3 heißen ***Zweige*** der Alternative.

12. Statische Semantik

$t ::= \cdots$	***Typen:***
$\mid$ *Bool*	Typ der Booleschen Werte

Typregeln:

$$\frac{}{\mathit{false} : \mathit{Bool}} \qquad \frac{}{\mathit{true} : \mathit{Bool}}$$

$$\frac{e_1 : \mathit{Bool} \qquad e_2 : t \qquad e_3 : t}{\textbf{\textit{if}}\ e_1\ \textbf{\textit{then}}\ e_2\ \textbf{\textit{else}}\ e_3 : t}$$

$$\frac{e_1 : \mathit{Nat} \qquad e_2 : \mathit{Nat}}{e_1 < e_2 : \mathit{Bool}} \qquad \text{usw.}$$

☞ Die Bedingung muss vom Typ *Bool* sein; die Zweige der Alternative können einen beliebigen Typ besitzen; dieser ist auch der Typ des gesamten Ausdrucks.

12. Dynamische Semantik

$v ::= \cdots$	***Boolesche Werte:***
$\mid$ *false*	falsch
$\mid$ *true*	wahr

Auswertungsregeln:

$$\frac{}{\mathit{false} \Downarrow \mathit{false}} \qquad \frac{}{\mathit{true} \Downarrow \mathit{true}}$$

$$\frac{e_1 \Downarrow \mathit{true} \qquad e_2 \Downarrow v}{\textbf{\textit{if}}\ e_1\ \textbf{\textit{then}}\ e_2\ \textbf{\textit{else}}\ e_3 \Downarrow v} \qquad \frac{e_1 \Downarrow \mathit{false} \qquad e_3 \Downarrow v}{\textbf{\textit{if}}\ e_1\ \textbf{\textit{then}}\ e_2\ \textbf{\textit{else}}\ e_3 \Downarrow v}$$

Natürliche Zahlen
Boolesche Werte
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

12. Dynamische Semantik — Vergleichsoperatoren

Für jeden Vergleichsoperator gibt es zwei Auswertungsregeln.

$$\frac{e_1 \Downarrow n+k \qquad e_2 \Downarrow n}{e_1 < e_2 \Downarrow \mathit{false}} \qquad \frac{e_1 \Downarrow n \qquad e_2 \Downarrow n+1+k}{e_1 < e_2 \Downarrow \mathit{true}} \qquad \text{usw.}$$

Natürliche Zahlen
Boolesche Werte
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

12. Beispielrechnung

$$\dfrac{\dfrac{\dfrac{\dfrac{}{4711 \Downarrow 4711} \quad \dfrac{}{815 \Downarrow 815}}{4711 < 815 \Downarrow \mathit{false}} \quad \dfrac{}{\mathit{true} \Downarrow \mathit{true}}}{\textbf{\textit{if}}\ 4711 < 815\ \textbf{\textit{then}}\ \mathit{false}\ \textbf{\textit{else}}\ \mathit{true} \Downarrow \mathit{true}} \quad \dfrac{}{\texttt{"yes"} \Downarrow \texttt{"yes"}}}{\textbf{\textit{if}}\ (\textbf{\textit{if}}\ 4711 < 815\ \textbf{\textit{then}}\ \mathit{false}\ \textbf{\textit{else}}\ \mathit{true})\ \textbf{\textit{then}}\ \texttt{"yes"}\ \textbf{\textit{else}}\ \texttt{"no"} \Downarrow \texttt{"yes"}}$$

☞ Alternativen dürfen beliebig geschachtelt werden: die Bedingung der äußeren Alternative ist wiederum eine Alternative.

12. Demo

```
⋙ false
false
⋙ true
true
⋙ if false then "yes" else "no"
"no"
⋙ if true then "yes" else "no"
"yes"
⋙ if 4711 < 815 then "yes" else "no"
"no"
⋙ if (if 4711 < 815 then false else true) then "yes" else "no"
"yes"
⋙ if 4711 < 815 then true else 7 > 1
true
⋙ if 4711 < 815 then 7 > 1 else false
false
```


Natürliche Zahlen

Boolesche Werte

Motivation

Abstrakte Syntax

Statische Semantik

Dynamische Semantik

Vertiefung

Werte/-definitionen

Funktions/-definitionen

Funktions-ausdrücke

Rekursion

Entwurfsmuster

12. Vertiefung

Ein Boolescher Ausdruck modelliert einen Sachverhalt oder eine Aussage.

Wir sind gewohnt, einfache Aussagen zu komplexen Aussagen zusammenzusetzen:

- ***Negation***: Der Kunde ist *nicht* kreditwürdig.
- ***Konjunktion***: Das Konto ist überzogen *und* der Kunde ist nicht kreditwürdig.
- ***Disjunktion***: Das Netzteil ist defekt *oder* die Leitung ist unterbrochen.

Natürliche Zahlen
Boolesche Werte
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

12. Vertiefung

- ***Negation*** von e:

  if e ***then*** *false* ***else*** *true*

- ***Konjunktion*** von e_1 und e_2:

  if e_1 ***then*** e_2 ***else*** *false*

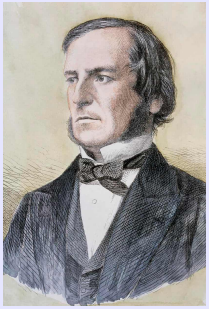
- ***Disjunktion*** von e_1 und e_2:

  if e_1 ***then*** *true* ***else*** e_2

☞ Wir kürzen die Negation von e mit *not* e, die Konjunktion von e_1 und e_2 mit e_1 && e_2 und die Disjunktion mit e_1 || e_2 ab.

12. George Boole (1815–1864)

Der englische Mathematiker George Boole entwickelte in seiner Schrift »The Mathematical Analysis of Logic« von 1847 den ersten algebraischen Logikkalkül und begründete damit die moderne mathematische Logik.



Boole stellte die Wahrheitswerte durch die Zahlen 0 und 1 dar und drückte die logischen Operationen entsprechend durch arithmetische Operationen aus.

56 OF HYPOTHETICALS.

1st. Disjunctive Syllogism.

Either X is true, or Y is true (exclusive),	$x + y - 2xy = 1$
But X is true,	$x = 1$
Therefore Y is not true,	$\therefore y = 0$
Either X is true, or Y is true (not exclusive),	$x + y - xy = 1$
But X is not true,	$x = 0$
Therefore Y is true,	$\therefore y = 1$

Natürliche Zahlen
Boolesche Werte
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Werte/-definitionen
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Knobelaufgabe #4

Ein Ausdruck besteht aus einer Folge von Zeichen; ein Ausdruck vom Typ *String* wertet zu einer Zeichenfolge aus. Schreiben Sie ein Programm, das zu seinem eigenen Programmtext auswertet.

Easy!

```
⋙ "done"
"done"
```

Nicht ganz Harry: das Programm `"done"` besteht aus 6 Zeichen, sein Wert aus 4.

??? … Ach so, die Anführungsstriche!!! Schnell repariert:

```
⋙ show "done"
"\"done\""
```

Ich sehe, Du hast die Funktion *show* entdeckt, die einen Wert in seine externe Darstellung überführt. Jetzt besteht Dein Programm aus 11 Zeichen, der Wert aus 6.
Ein Tipp: wenn Du nach der Auswertung des Ausdrucks *putline it* eintippst, musst Du Zeichen für Zeichen den Programmtext erhalten.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Motivation

- Rechnungen sind in der Regel nicht linear, sie enthalten Zwischen- oder Hilfsrechnungen.
- Beispiel: Berechnung des Flächeninhalts eines Quadrats:
  - zunächst: Seitenlänge ausrechnen,
  - dann: Ergebnis der Zwischenrechnung mit sich selbst multiplizieren.
- Um das Ergebnis einer Zwischenrechnung gegebenenfalls mehrfach verwenden zu können, geben wir ihm einen Namen :

$$\textit{let}\ s = 4711 + 815$$

Das Konstrukt ist eine ***Wertedefinition***: der Bezeichner links wird an den Wert des Ausdrucks rechts gebunden. *Lies:* sei s gleich $4711 + 815$.

- Ein Bezeichner ist ein Ausdruck und somit auch:

$$s * s$$

Der obige Ausdruck berechnet den gewünschten Flächeninhalt.

13. Motivation

- Welcher Bezeichner kann wo verwendet werden?
- Wie werden Wertedefinitionen und Ausdrücke verknüpft?
- Der Zusammenhang wird durch einen ***in***-Ausdruck hergestellt.

$$\textbf{\textit{let}}\ s = 4711 + 815\ \textbf{\textit{in}}\ s * s$$

  - Vor ***in*** steht eine Wertedefinition.
  - Nach ***in*** steht ein Ausdruck.
  - Das gesamte Konstrukt ist wiederum ein Ausdruck.
  - Der Bezeichner s ist nur in $s * s$ ***sichtbar***.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Beispiel: Ausflug in den Zoo

Problem:

Die Klasse 2c macht einen Ausflug in den Zoo. Es fahren 27 Schülerinnen und Schüler und 3 Lehrer mit. Die Fahrtkosten betragen 2€ für Kinder und 3€ für Erwachsene. Kinder zahlen 5€ Eintritt und Erwachsene 10€. Wie teuer ist der Ausflug?

Berechnung der Kosten:

$$
\begin{array}{lll}
\textit{let}\ \textit{Schüler} & = 27 & \textit{in} \\
\textit{let}\ \textit{Lehrer} & = 3 & \textit{in} \\
\textit{let}\ \textit{Fahrtkosten} & = 2 * \textit{Schüler} + 3 * \textit{Lehrer} & \textit{in} \\
\textit{let}\ \textit{Eintritt} & = 5 * \textit{Schüler} + 10 * \textit{Lehrer} & \textit{in} \\
\multicolumn{3}{l}{\textit{Fahrtkosten} + \textit{Eintritt}}
\end{array}
$$

☞ Benennung von Teilrechnungen.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Wahl der Bezeichner

- Der Bezeichner in einer Wertedefinition kann frei gewählt werden:

  let $s = 4711 + 815$ ***in*** $s * s$

  ist gleichwertig zu

  let *size* $= 4711 + 815$ ***in*** *size* $*$ *size*

- Für das Ausrechnen spielen Namen keine Rolle, wohl aber für den menschlichen Betrachter eines Programms. *Deshalb:* möglichst aussagekräftige Bezeichner vergeben.
- *Allgemein gilt:* je größer der Sichtbarkeitsbereich eines Namens, desto mehr Sorgfalt sollte man bei der Namenswahl walten lassen.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Abstrakte Syntax — Definitionen

Wir führen eine neue syntaktische Kategorie ein: ***Deklarationen***.

$x \in \mathsf{Ident}$	***Bezeichner***
$d \in \mathsf{Decl} ::=$	***Deklarationen:***
$\quad \mid$ ***let*** $x = e$	Wertedefinition

☞ Der Bereich der Bezeichner Ident wird in Teil VI genau festgelegt.

Für's erste: ein Bezeichner fängt mit einem Buchstaben an. Danach können weitere Buchstaben, Ziffern, und Sonderzeichen wie ein Unterstrich oder ein Apostroph folgen.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Abstrakte Syntax — Ausdrücke

Ein ***in***-Ausdruck verknüpft eine Definition mit einem Ausdruck.

$e ::= \cdots$	***lokale Definitionen:***
$\mid x$	Bezeichner
$\mid d$ ***in*** e	lokale Definition

Der Teilausdruck e heißt Rumpf des ***in***-Ausdrucks.

13. Statische Semantik

- Wie werden ***in***-Ausdrücke typisiert?

  $$\textbf{\textit{let}}\ s = 4711 + 815\ \textbf{\textit{in}}\ s * s$$

  Wenn wir den Teilausdruck $s * s$ typisieren, woher kennen wir den Typ von s?
- Ein Bezeichner kann in unterschiedlichen Kontexten einen unterschiedlichen Typ besitzen.

  $$(\textbf{\textit{let}}\ s = \mathit{true}\ \textbf{\textit{in}}\ s)\ \&\&\ (\textbf{\textit{let}}\ s = 815\ \textbf{\textit{in}}\ s * s > 4711)$$

13. Statische Semantik — Signatur

- Wir merken uns die Typen von Bezeichnern mit Hilfe einer sogenannten Signatur.

$$\Sigma \in \mathsf{Sig} = \mathsf{Ident} \to_{\mathrm{fin}} \mathsf{Type} \qquad \textit{Signatur}$$

Eine Signatur ist eine endliche Abbildung von Bezeichnern auf Typen.

- Wir erweitern die Typregeln um Signaturen. Die Regeln spezifizieren die nunmehr *dreistellige* Relation

$$\Sigma \vdash e : t$$

zwischen Signaturen, Ausdrücken und Typen. *Lies:* »bezüglich der Signatur Σ hat e den Typ t«.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Statische Semantik — Definitionen

Die statische Semantik ordnet

- einem Ausdruck einen Typ und
- einer Definition eine Signatur

zu.

Typregel:

$$\frac{\Sigma \vdash e : t}{\Sigma \vdash (\textbf{\textit{let}}\ x = e) : \{x \mapsto t\}}$$

☞ Eine Signatur repräsentiert — ähnlich wie ein Typ — das, was wir über eine Definition statisch wissen.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Statische Semantik — Ausdrücke

Typregeln:

$$\frac{}{\Sigma \vdash x : \Sigma(x)} \qquad x \in dom\ \Sigma$$

☞ Zu jedem Bezeichner muss eine definierende Bindung existieren.

$$\frac{\Sigma \vdash d : \Sigma' \qquad \Sigma, \Sigma' \vdash e : t}{\Sigma \vdash (d\ \textbf{\textit{in}}\ e) : t}$$

☞ Der Teilausdruck e wird bezüglich der Signatur Σ, Σ' typisiert. Der Kommaoperator erweitert Σ um Σ' und regelt Überschneidungen.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Statische Semantik — Beispiel

Beispiel für einen wohlgetypten Ausdruck:

$$\dfrac{\dfrac{}{\varnothing \vdash 47 : \mathit{Nat}}}{\varnothing \vdash (\textbf{\textit{let}}\ s = 47) : \{s \mapsto \mathit{Nat}\}} \quad \dfrac{\dfrac{}{\{s \mapsto \mathit{Nat}\} \vdash s : \mathit{Nat}} \quad \dfrac{}{\{s \mapsto \mathit{Nat}\} \vdash s : \mathit{Nat}}}{\{s \mapsto \mathit{Nat}\} \vdash s * s : \mathit{Nat}}$$
$$\varnothing \vdash (\textbf{\textit{let}}\ s = 47\ \textbf{\textit{in}}\ s * s) : \mathit{Nat}$$

Beispiel für einen *nicht* wohlgetypten Ausdruck:

$$\dfrac{\dfrac{}{\varnothing \vdash 47 : \mathit{Nat}}}{\varnothing \vdash (\textbf{\textit{let}}\ s = 47) : \{s \mapsto \mathit{Nat}\}} \quad \dfrac{\dfrac{}{\{s \mapsto \mathit{Nat}\} \vdash s : \mathit{Nat}} \quad \{s \mapsto \mathit{Nat}\} \vdash a : \mathit{Nat}?}{\{s \mapsto \mathit{Nat}\} \vdash s * a : \mathit{Nat}}$$
$$\varnothing \vdash (\textbf{\textit{let}}\ s = 47\ \textbf{\textit{in}}\ s * a) : \mathit{Nat}$$

☞ Die Formel $\{s \mapsto \mathit{Nat}\} \vdash a : \mathit{Nat}$ lässt sich nicht ableiten.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Dynamische Semantik

- Wie werden ***in***-Ausdrücke ausgerechnet?

  let $s = 4711 + 815$ ***in*** $s * s$

  Wenn wir den Teilausdruck $s * s$ ausrechnen, woher kennen wir den Wert von s?
- Ein Bezeichner kann in unterschiedlichen Kontexten einen unterschiedlichen Wert besitzen.

  $(\textbf{\textit{let}}\ s = \textit{true}\ \textbf{\textit{in}}\ s)\ \&\&\ (\textbf{\textit{let}}\ s = 815\ \textbf{\textit{in}}\ s * s > 4711)$

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Dynamische Semantik — Umgebung

- Wir merken uns die Werte von Bezeichnern mit Hilfe einer sogenannten Umgebung.

$$\delta \in \mathsf{Env} = \mathsf{Ident} \to_{\text{fin}} \mathsf{Val} \qquad \textit{Umgebung}$$

Wie eine Signatur ist eine Umgebung eine endliche Abbildung; im Unterschied zur Signatur bildet sie Bezeichner auf *Werte* ab.

- Wir erweitern die Auswertungsregeln um Umgebungen. Die Regeln spezifizieren die nunmehr *dreistellige* Relation

$$\delta \vdash e \Downarrow v$$

zwischen Umgebungen, Ausdrücken und Werten. *Lies:* »bezüglich der Umgebung δ wertet e zu v aus«.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Dynamische Semantik — Definitionen

Die dynamische Semantik ordnet

- einem Ausdruck einen Wert und
- einer Definition eine Umgebung

zu.

Auswertungsregel:

$$\frac{\delta \vdash e \Downarrow v}{\delta \vdash (\textbf{\textit{let}}\ x = e) \Downarrow \{x \mapsto v\}}$$

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Dynamische Semantik — Ausdrücke

Auswertungsregeln:

$$\frac{}{\delta \vdash x \Downarrow \delta(x)} \qquad x \in dom\ \delta$$

☞ Zu jedem Bezeichner muss eine definierende Bindung existieren.

$$\frac{\delta \vdash d \Downarrow \delta' \qquad \delta, \delta' \vdash e \Downarrow v}{\delta \vdash (d\ \textbf{\textit{in}}\ e) \Downarrow v}$$

☞ Der Teilausdruck e wird bezüglich der Umgebung δ, δ' ausgewertet. Der Kommaoperator erweitert δ um δ' und regelt Überschneidungen.

13. Dynamische Semantik — Beispiel

$$\dfrac{\dfrac{\dfrac{}{\varnothing \vdash 4711 + 815 \Downarrow 5526}}{\varnothing \vdash \textbf{\textit{let}}\ s = 4711 + 815 \Downarrow \delta} \qquad \dfrac{\dfrac{}{\delta \vdash s \Downarrow 5526} \quad \dfrac{}{\delta \vdash s \Downarrow 5526}}{\delta \vdash s * s \Downarrow 30536676}}{\varnothing \vdash \textbf{\textit{let}}\ s = 4711 + 815\ \textbf{\textit{in}}\ s * s \Downarrow 30536676}$$

wobei $\delta = \{s \mapsto 5526\}$

☞ Der Rumpf $s * s$ wird bezüglich der Umgebung $\{s \mapsto 5526\}$ ausgerechnet.

☞ Aus Gründen der Übersichtlichkeit kürzen wir einfache Teilrechnungen ab.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Demo

```
⋙ let s = 4711 + 815 in s * s
30536676
⋙ let size = 4711 + 815 in size * size
30536676
⋙ (let s = 4711 + 815 in s * s) + 1
30536677
⋙ (let s = 4711 in s * s) + (let s = 815 in s * s)
22857746
⋙ let s = 4711 in let a = s * s in a + s
22198232
```


Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Freie Bezeichner

Ein Bezeichner, der nicht im Geltungsbereich einer Definition liegt, heißt ***frei***.

Ausdruck	freie Bezeichner
$s * s$	$\{s\}$
$s * a$	$\{a, s\}$
let $s = 4711$ ***in*** $s * s$	$\emptyset$
let $s = 4711$ ***in*** $s * a$	$\{a\}$

☞ In dem Ausdruck d ***in*** e werden die in d definierten Bezeichner von den freien Bezeichnern in e abgezogen.

Ein Ausdruck, der keine freien Bezeichner enthält, heißt ***geschlossen***.

13. Invarianten der Semantik

Invariante der statischen Semantik:

Die statische Semantik typisiert nur Ausdrücke, deren freie Bezeichner in der Signatur aufgeführt werden. — Bevor ein Teilausdruck, der freie Bezeichner enthält, typisiert wird, wird zunächst die Signatur um die Typen der freien Bezeichner erweitert.

Invariante der dynamischen Semantik:

Die dynamische Semantik legt nur die Bedeutung von Ausdrücken fest, deren freie Bezeichner in der Umgebung aufgeführt werden. — Bevor ein Teilausdruck, der freie Bezeichner enthält, ausgewertet wird, wird zunächst die Umgebung um die Werte der freien Bezeichner erweitert.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Redefinition

Was passiert, wenn man den gleichen Bezeichner mehrfach definiert?

$$\dfrac{\dfrac{}{\varnothing \vdash 4 \Downarrow 4}}{\varnothing \vdash \mathit{let}\ s = 4 \Downarrow \{s \mapsto 4\}} \qquad \dfrac{\dfrac{}{\{s \mapsto 4\} \vdash 7 \Downarrow 7}}{\{s \mapsto 4\} \vdash \mathit{let}\ s = 7 \Downarrow \{s \mapsto 7\}} \qquad \dfrac{\dfrac{}{\{s \mapsto 7\} \vdash s \Downarrow 7} \quad \vdots}{\star\{s \mapsto 7\} \vdash s * s \Downarrow 49}$$

$$\dfrac{\varnothing \vdash \mathit{let}\ s = 4 \Downarrow \{s \mapsto 4\} \qquad \{s \mapsto 4\} \vdash \mathit{let}\ s = 7 \Downarrow \{s \mapsto 7\}}{\{s \mapsto 4\} \vdash \mathit{let}\ s = 7\ \mathit{in}\ s * s \Downarrow 49}$$

$$\varnothing \vdash \mathit{let}\ s = 4\ \mathit{in}\ \mathit{let}\ s = 7\ \mathit{in}\ s * s \Downarrow 49$$

⋆ Der Kommaoperator räumt der »neuen« Definition Vorrang ein:
$\{s \mapsto 4\}, \{s \mapsto 7\} = \{s \mapsto 7\}$.

13. Redefinition

Sollte man den überhaupt Bezeichner redefinieren? Ich meine, ist das nicht schlechter Programmierstil?

Man soll in der Tat darauf achten, Bezeichner nicht zu redefinieren. *Allerdings:* neue Namen zu erfinden ist schwer.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Umbenennungen

Die Festlegung, »neuen« Definitionen Vorrang einzuräumen, ist sinnvoll. Sie ist insbesondere kompatibel zum Umbenennen von Bezeichnern.

Der Bezeichner *s* in

$$\textbf{\textit{let}}\ s = 815\ \textbf{\textit{in}}\ s * s$$

kann zu *size* umbenannt werden,

$$\textbf{\textit{let}}\ size = 815\ \textbf{\textit{in}}\ size * size$$

ohne die Bedeutung des Programms zu ändern.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Umbenennungen

Die Bedeutung bleibt ebenso unverändert, wenn der Ausdruck *Teil* eines größeren Programms ist.

Der Bezeichner *s* im rechten ***in***-Ausdruck

$$\textbf{\textit{let}}\ s = 4711\ \textbf{\textit{in let}}\ s = 815\ \textbf{\textit{in}}\ s * s$$

kann zu *size* umbenannt werden,

$$\textbf{\textit{let}}\ s = 4711\ \textbf{\textit{in let}}\ \mathit{size} = 815\ \textbf{\textit{in}}\ \mathit{size} * \mathit{size}$$

ohne die Bedeutung des Programms zu ändern.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Demo

$\ggg$ (*let* $s = 4711$ *in* $s * s$) + (*let* $s = 815$ *in* $s * s$)
22857746
$\ggg$ *let* $s = 4711$ *in let* $s = 815$ *in* $s * s$
664225
$\ggg$ *let* $s_1 = 4711$ *in let* $s_2 = 815$ *in* $s_2 * s_2$
664225
$\ggg$ *let* $s = 4711$ *in let* $a = s * s$ *in* $a + a$
44387042
$\ggg$ *let* $a =$ *let* $s = 4711$ *in* $s * s$ *in* $a + a$
44387042

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Über den Tellerrand: F#

Wollen wir mehrere Definitionen in einem Ausdruck verwenden, müssen wir ***in***-Ausdrücke aneinanderreihen.

```
let a = 4711
      + 815
in let b = a * a
  in a + b
```

F# erlaubt alternativ das Ende einer lokalen Definition und den Anfang des Rumpfes mit Hilfe des *Layouts* festzulegen.

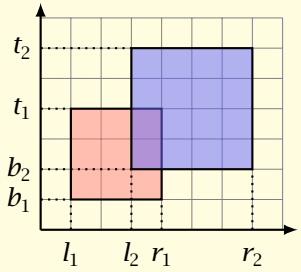
```
let a = 4711
      + 815
let b = a * a
a + b
```

Abseitsregel: die Einrückung bestimmt die Zugehörigkeit von Teilausdrücken. Einrückung: der »aktuelle« Ausdruck wird weitergeführt; keine Einrückung: eine neue Definition oder der Rumpfausdruck wird begonnen.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Beispiel: Berechnung der Wohnfläche

Die Fläche einer Wohnung mit dem unten skizzierten Grundriss soll berechnet werden.



(Das ist eine Vereinfachung, das Ergebnis des schon angesprochenen Abstraktionsprozesses, mit dem wir Aufgaben in Rechenaufgaben verwandeln.)

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Beispiel: Berechnung der Wohnfläche

Erfassung der Daten:

$$
\begin{aligned}
&\textit{let } l_1 = 1\\
&\textit{let } r_1 = 4\\
&\textit{let } b_1 = 1\\
&\textit{let } t_1 = 4\\
&\textit{let } l_2 = 3\\
&\textit{let } r_2 = 7\\
&\textit{let } b_2 = 2\\
&\textit{let } t_2 = 6
\end{aligned}
$$

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Beispiel: Berechnung der Wohnfläche

Gesamtfläche: Summe der beiden Quadratflächen minus der Schnittfläche.

$$
\begin{array}{l}
\textbf{\textit{let}}\ s_1 = r_1 \dot{-} l_1 \\
\textbf{\textit{let}}\ s_2 = r_2 \dot{-} l_2 \\
\textbf{\textit{let}}\ w = r_1 \dot{-} l_2 \\
\textbf{\textit{let}}\ h = t_1 \dot{-} b_2 \\
s_1 * s_1 + s_2 * s_2 \dot{-} w * h
\end{array}
$$

☞ Vermeidung von Mehrfachrechnungen: $s_1 * s_1$ statt $(r_1 \dot{-} l_1) * (r_1 \dot{-} l_1)$.

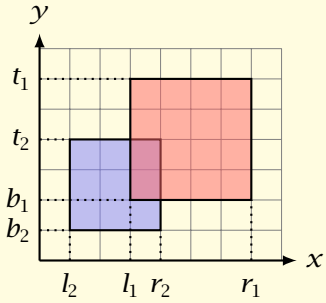
$$
\begin{array}{ll}
\ggg & \ldots \\
\ggg & s_1 * s_1 + s_2 * s_2 \dot{-} w * h \\
23 &
\end{array}
$$

Die Wohnfläche beträgt somit $23\,m^2$.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Beispiel: Berechnung der Wohnfläche

Ändern wir die Daten, indem wir die Koordinaten der beiden Quadrate »vertauschen«,



$$\begin{aligned}
&\textit{let } l_1 = 3\\
&\textit{let } r_1 = 7\\
&\textit{let } b_1 = 2\\
&\textit{let } t_1 = 6\\
&\textit{let } l_2 = 1\\
&\textit{let } r_2 = 4\\
&\textit{let } b_2 = 1\\
&\textit{let } t_2 = 4
\end{aligned}$$

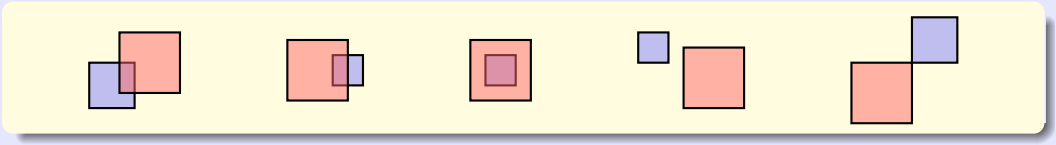
erleben wir eine unangenehme Überraschung:

$$\begin{aligned}
&\ggg \quad \textit{let } l_1 = 3\\
&\ggg \quad \ldots\\
&\ggg \quad s_1 * s_1 + s_2 * s_2 \doteq w * h\\
&0
\end{aligned}$$

13. Beispiel: Berechnung der Wohnfläche

☞ Das obige Programm macht unausgesprochene Annahmen über die relative Lage der beiden Quadrate. (Welche?)

Wollen wir das Programm »robuster« machen, müssen wir die Lage der Quadrate berücksichtigen. Viele unterschiedliche Konstellationen sind denkbar:

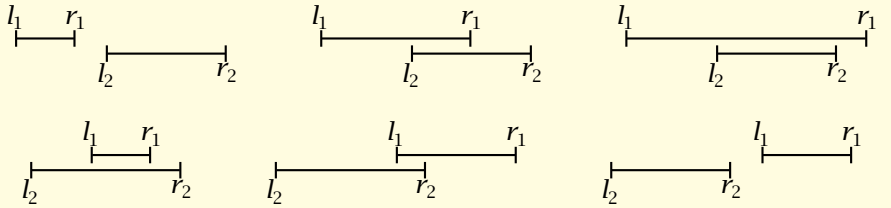


Wie werden wir Herr oder Frau der Lage?

☞ *Entscheidende Einsicht:* Überschneidungen können *getrennt* für jede der beiden Koordinatenachsen ausgerechnet werden. Auf diese Weise wird ein 2-dimensionales auf ein 1-dimensionales Problem zurückgeführt.

13. Beispiel: Berechnung der Wohnfläche

Konzentrieren wir uns auf die x-Achse. Wenn wir annehmen, dass $l_1 < r_1$ und $l_2 < r_2$ gilt, dann ergeben sich sechs mögliche Konstellationen:



Länge der Überschneidung: $\min\ r_1\ r_2 \div \max\ l_1\ l_2$.

13. Intermezzo: Minimum und Maximum

- ***Minimum*** von e_1 und e_2:

  if $e_1 \leqslant e_2$ ***then*** e_1 ***else*** e_2

- ***Maximum*** von e_1 und e_2:

  if $e_1 \leqslant e_2$ ***then*** e_2 ***else*** e_1

☞ Wir kürzen das Minimum von e_1 und e_2 mit *min* e_1 e_2 und das Maximum mit *max* e_1 e_2 ab.

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster

13. Intermezzo: Minimum und Maximum

☞ Ist die Berechnung von e_1 oder e_2 aufwändig, formuliert man besser:

- ***Minimum*** von e_1 und e_2:

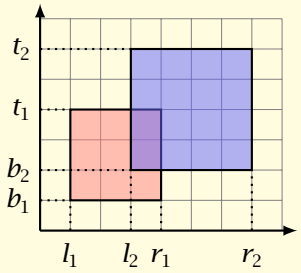
  let $a_1 = e_1$
  let $a_2 = e_2$
  if $a_1 \leqslant a_2$ ***then*** a_1 ***else*** a_2

- ***Maximum*** von e_1 und e_2:

  let $a_1 = e_1$
  let $a_2 = e_2$
  if $a_1 \leqslant a_2$ ***then*** a_2 ***else*** a_1

Warum?

13. Beispiel: Berechnung der Wohnfläche



$$
\begin{aligned}
&\textit{let } l_1 = 1\\
&\textit{let } r_1 = 4\\
&\textit{let } b_1 = 1\\
&\textit{let } t_1 = 4\\
&\textit{let } l_2 = 3\\
&\textit{let } r_2 = 7\\
&\textit{let } b_2 = 2\\
&\textit{let } t_2 = 6
\end{aligned}
$$

Lösung:

$$
\begin{aligned}
&\textit{let } s_1 = r_1 \dot{-} l_1\\
&\textit{let } s_2 = r_2 \dot{-} l_2\\
&\textit{let } w = \mathit{min}\ r_1\ r_2 \dot{-} \mathit{max}\ l_1\ \ l_2\\
&\textit{let } h = \mathit{min}\ t_1\ t_2 \dot{-} \mathit{max}\ b_1\ b_2\\
&s_1 * s_1 + s_2 * s_2 \dot{-} w * h
\end{aligned}
$$

Natürliche Zahlen
Boolesche Werte
Werte/-definitionen
Motivation
Abstrakte Syntax
Statische Semantik
Dynamische Semantik
Vertiefung
Über den Tellerrand
Funktions/-definitionen
Funktions-ausdrücke
Rekursion
Entwurfsmuster