Programmierung 1

Vorlesung 19

Livestream beginnt um 14:15 Uhr

Statische und dynamische Semantik

Programmierung 1

Verarbeitungsphasen eines Interpreters

Lexikalische Analyse
Lexikalische Syntax

Kann die eingegebene Zeichenfolge als **Folge von Wörtern** aufgefasst werden?

Syntaktische Analyse
Phrasale Syntax

Beschreibt die Wortfolge ein **Programm**?

Kapitel 13

Semantische Analyse
Statische Semantik

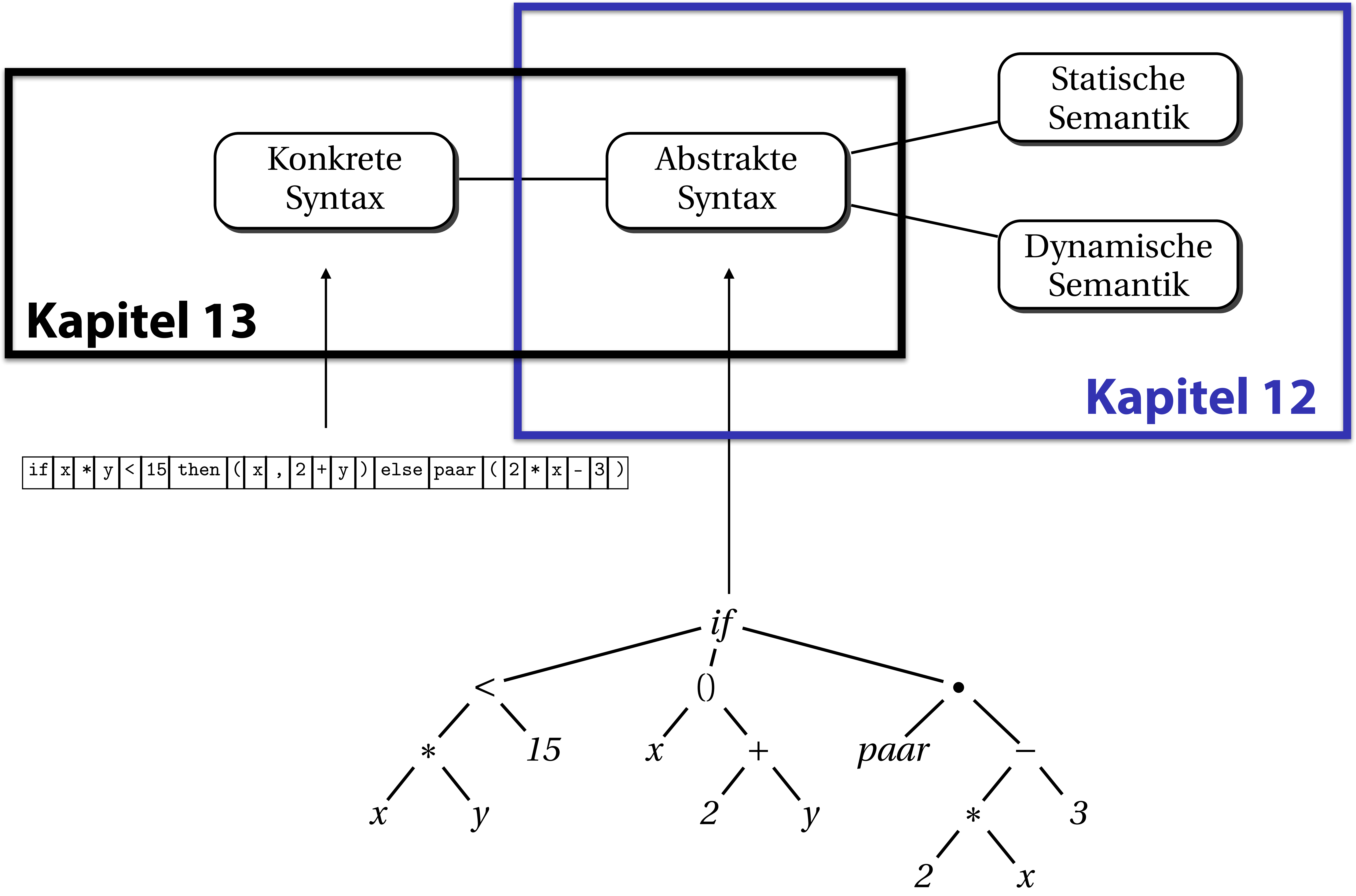
Ist das Programm **semantisch zulässig**?

Ausführung
Dynamische Semantik

Führe das Programm **aus.**

Kapitel 12

Abstrakte Syntax



Abstrakte Syntax von F

Die abstrakte Syntax wird üblicherweise mithilfe einer schematischen Darstellung, der **abstrakten Grammatik**, definiert.

z	$\in$	$\mathbb{Z}$	Zahlen
c	$\in$	$\mathit{Con} = \mathit{false} \mid \mathit{true} \mid z$	Konstanten
x	$\in$	$\mathit{Id} = \mathbb{N}$	Bezeichner
o	$\in$	$\mathit{Opr} = + \mid - \mid * \mid \leq$	Operatoren
t	$\in$	$\mathit{Ty} = \mathit{bool} \mid \mathit{int} \mid t \to t$	Typen
e	$\in$	$\mathit{Exp} =$	Ausdrücke
		c	Konstante
		$\mid x$	Bezeichner
		$\mid eoe$	Operatoranwendung
		$\mid \mathit{if}\ e\ \mathit{then}\ e\ \mathit{else}\ e$	Konditional
		$\mid \mathit{fn}\ x : t \Rightarrow e$	Abstraktion
		$\mid ee$	Prozeduranwendung

Die **Grammatik** definiert die **Mengen** *Con, Id, Opr, Ty,* und *Exp*

z, x, o, t, e sind **Metavariablen:** sie bezeichnen Objekte der Mengen

Typdeklarationen

In Standard ML können wir die **abstrakte Syntax** einer Sprache als **Typdeklarationen** darstellen.

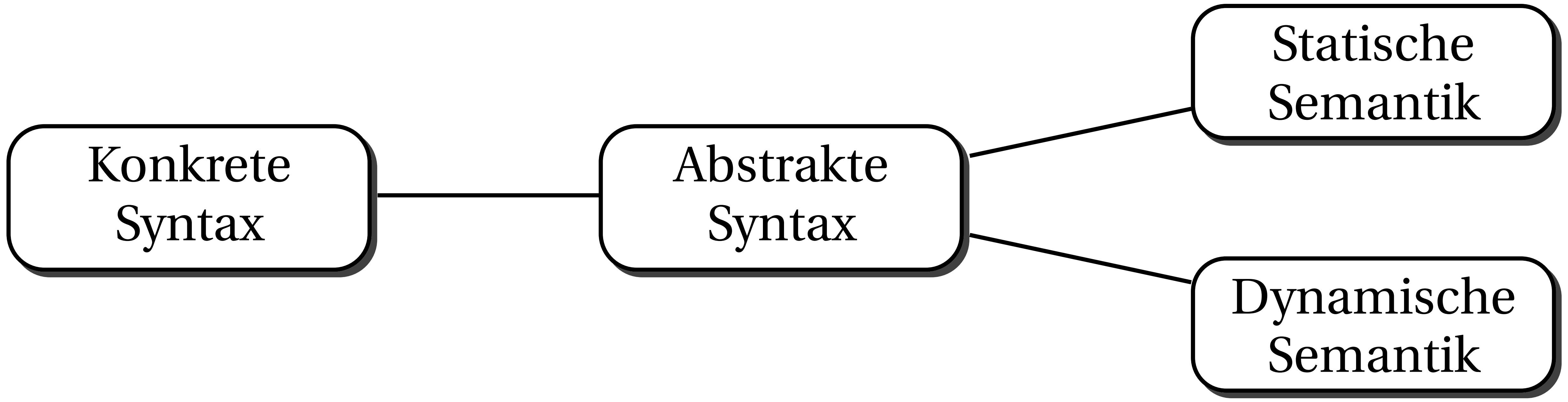
$$
\begin{array}{lll}
z & \in & \mathbb{Z} \\
c & \in & \mathit{Con} = \mathit{false} \mid \mathit{true} \mid z \\
x & \in & \mathit{Id} \;\; = \mathbb{N} \\
o & \in & \mathit{Opr} = + \mid - \mid * \mid \leq \\
t & \in & \mathit{Ty} \;\; = \mathit{bool} \mid \mathit{int} \mid t \to t \\
e & \in & \mathit{Exp} = \\
 & & \quad c \\
 & & \mid x \\
 & & \mid eoe \\
 & & \mid \mathit{if}\ e\ \mathit{then}\ e\ \mathit{else}\ e \\
 & & \mid \mathit{fn}\ x : t \Rightarrow e \\
 & & \mid ee
\end{array}
$$

```
datatype con = False | True | IC of int
type     id  = string
datatype opr = Add | Sub | Mul | Leq

datatype ty  =
    Bool
  | Int
  | Arrow of ty * ty

datatype exp =
    Con of con
  | Id  of id
  | Opr  of opr * exp * exp
  | If  of exp * exp * exp
  | Abs of id  * ty  * exp
  | App of exp * exp
```

Statische Semantik



- Die **statische Semantik** prüft die **semantische Zulässigkeit** eines Ausdrucks.
  Sind alle auftretenden **Bezeichner** bekannt?
  Wenn ja, ist der Ausdruck **wohlgetypt**?
- Die **dynamische Semantik** wertet einen semantisch zulässigen Ausdruck aus.
  Terminiert die Auswertung?
  Wenn ja, mit welchem **Wert**?

Typregeln

$$\frac{e_1 : t_1 \qquad o : t_1 * t_2 \rightarrow t \qquad e_2 : t_2}{e_1\ o\ e_2\ :\ t}$$

Die Regel besagt, dass eine Anwendung $e_1\ o\ e_2$ **wohlgetypt** ist und **den Typ** t **hat**, wenn

1. der linke Teilausdruck e_1 den Typ t_1 hat,
2. der Operator o den Typ $t_1 * t_2 \rightarrow t$ hat, und
3. der rechte Teilausdruck e_2 den Typ t_2 hat.

Beispiel: Ausdruck x+5

Wenn wir annehmen, dass x den Typ int hat, dann folgt, weil + den Typ int * int → int und 5 den Typ int hat, dass x+5 wohlgetypt ist und den Typ int hat.

Statische Semantik

- Eine **Typumgebung** ist eine Funktion T, die endlich vielen Bezeichnern je einen Typ zuordnet:

$$T \quad \in \quad TE \quad = \quad Id \xrightarrow{\text{fin}} Ty$$

Beispiel: $\{(x, int),\ (y, bool)\}$
Notation: $[x := int,\ y := bool]$

- Die **statische Semantik von F** ist eine Menge

$$SS \subseteq TE \times Exp \times Ty$$

die ein Tupel $\langle T, e, t\rangle$ genau dann enthält, wenn der Ausdruck e für die Typumgebung T **zulässig** ist und den **Typ** t hat.

Beispiel: SS enthält $\langle [x := int],\ 2 * x + 3,\ int\rangle$

Für welche Typumgebungen ist der Ausdruck

if true then *x* else *y*

zulässig?

A: [x:=bool]

B: [x:=bool, y:=bool, z:=int]

C: [x:=bool, y:=bool]

D: [x:=bool, y:=int, z:=bool]

Statische Semantik

- Wir definieren *SS* mithilfe von **Inferenzregeln**:

$$\frac{\langle T, e_1, t' \to t\rangle \in SS \qquad \langle T, e_2, t'\rangle \in SS}{\langle T, e_1 e_2, t\rangle \in SS}$$

- **Notation:** $T \vdash e : t \qquad :\Longleftrightarrow \qquad \langle T, e, t\rangle \in SS$
- Damit:

$$\frac{T \vdash e_1 : t' \to t \qquad T \vdash e_2 : t'}{T \vdash e_1 e_2 : t}$$

Statische Semantik von F

Sif $$\frac{T \vdash e_1 : bool \qquad T \vdash e_2 : t \qquad T \vdash e_3 : t}{T \vdash \mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3 : t}$$

Sabs $$\frac{T[x := t] \vdash e : t'}{T \vdash \mathit{fn}\ x : t \Rightarrow e : t \to t'}$$

Sapp $$\frac{T \vdash e_1 : t' \to t \qquad T \vdash e_2 : t'}{T \vdash e_1 e_2 : t}$$

Eine Ableitung

$$
\textbf{Soai}\ \dfrac{\textbf{Snum}\ \dfrac{2 \in \mathbb{Z}}{[x := int,\ b := bool] \vdash 2 : int} \qquad \textbf{Sid}\ \dfrac{[x := int,\ b := bool]\ x = int}{[x := int,\ b := bool] \vdash x : int}}{[x := int,\ b := bool] \vdash 2 * x : int}
$$

$$
\textbf{Sabs}\ \dfrac{\textbf{Sif}\ \dfrac{\textbf{Sid}\ \dfrac{[x := int,\ b := bool]\ b = bool}{[x := int,\ b := bool] \vdash b : bool} \qquad \textbf{Sid}\ \dfrac{[x := int,\ b := bool]\ x = int}{[x := int,\ b := bool] \vdash x : int} \qquad [x := int,\ b := bool] \vdash 2 * x : int}{[x := int,\ b := bool] \vdash if\ b\ then\ x\ else\ 2 * x\ : int}}{[x := int] \vdash fn\ b : bool \Rightarrow if\ b\ then\ x\ else\ 2 * x\ : bool \to int}
$$

Eigenschaften der statischen Semantik von F

Proposition 12.1 (Determinismus)

Sei $T \vdash e : t$ *und* $T \vdash e : t'$. *Dann* $t = t'$.

Beweis durch strukturelle Induktion über $e \in exp$:

- Tupel mit dem Ausdruck e können jeweils nur mit **einer** Regel abgeleitet werden.
- Die Konklusionen der Regeln bestimmen die Umgebungen und Ausdrücke der Prämissen **eindeutig.**

Elaborierung

Die **Inferenzregeln** beschreiben einen **Algorithmus**, der für eine Typumgebung *T* und einen Ausdruck *e* **entscheidet**, ob *e* für *T* **zulässig** ist und im positiven Fall den Typ von *e* liefert.

$$elab\colon\ ty\ env \rightarrow exp \rightarrow ty$$

```
fun elabCon True    = Bool
  | elabCon False   = Bool
  | elabCon (IC _)  = Int

fun elab f (Con c) = elabCon c
```

Strue $\dfrac{}{T \vdash true : bool}$

Sfalse $\dfrac{}{T \vdash false : bool}$

Snum $\dfrac{z \in \mathbb{Z}}{T \vdash z : int}$

Elaborierung

$$\textbf{Soai}\quad \frac{o \in \{+,-,*\} \qquad T \vdash e_1 : \mathit{int} \qquad T \vdash e_2 : \mathit{int}}{T \vdash e_1\, o\, e_2 : \mathit{int}}$$

$$\textbf{Soab}\quad \frac{T \vdash e_1 : \mathit{int} \qquad T \vdash e_2 : \mathit{int}}{T \vdash e_1 \leq e_2 : \mathit{bool}}$$

```
exception Error of string
```

$$\textbf{Sid}\quad \frac{T\,x = t}{T \vdash x : t}$$

```
fun elabCon True    = Bool
  | elabCon False   = Bool
  | elabCon (IC _)  = Int
fun elabOpr Add Int Int = Int
  | elabOpr Sub Int Int = Int
  | elabOpr Mul Int Int = Int
  | elabOpr Leq Int Int = Bool
  | elabOpr _   _   _   = raise Error "T Opr"
fun elab f (Con c) = elabCon c
  | elab f (Id x) = f x
  | elab f (Opr(opr,e1,e2)) = elabOpr opr (elab f e1)
                                           (elab f e2)
```

Elaborierung

$$\textbf{Sif}\quad \frac{T \vdash e_1 : bool \qquad T \vdash e_2 : t \qquad T \vdash e_3 : t}{T \vdash \mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3 : t}$$

```
| elab f (If(e1,e2,e3)) =
            (case (elab f e1, elab f e2, elab f e3) of
               (Bool, t2, t3) => if t2=t3 then t2
                                 else raise Error "T If1"
             |  _ => raise Error "T If2")
| elab f (Abs(x,t,e)) = Arrow(t, elab (update f x t) e)
```

$$\textbf{Sabs}\quad \frac{T[x := t] \vdash e : t'}{T \vdash fn\ x : t \Rightarrow e : t \rightarrow t'}$$

```
| elab f (App(e1,e2)) = (case elab f e1 of
               Arrow(t',t) => if t' = elab f e2 then t
                              else raise Error "T App1"
             | _  => raise Error "T App2")
```

$$\textbf{Sapp}\quad \frac{T \vdash e_1 : t' \rightarrow t \qquad T \vdash e_2 : t'}{T \vdash e_1 e_2 : t}$$

```
fun update env x a y = if y=x then a else env y
```

```
| elabOpr Leq Int Int = Bool
| elabOpr _   _   _   = raise Error "T Opr"

| elab f (App(e1,e2)) = (case elab f e1 of
                    Arrow(t',t) => if t' = elab f e2 then t
                                   else raise Error "T App1"
                  | _  => raise Error "T App2")
```

Was ist das Ergebnis von
elab empty (App (Con True, Con True)) ?

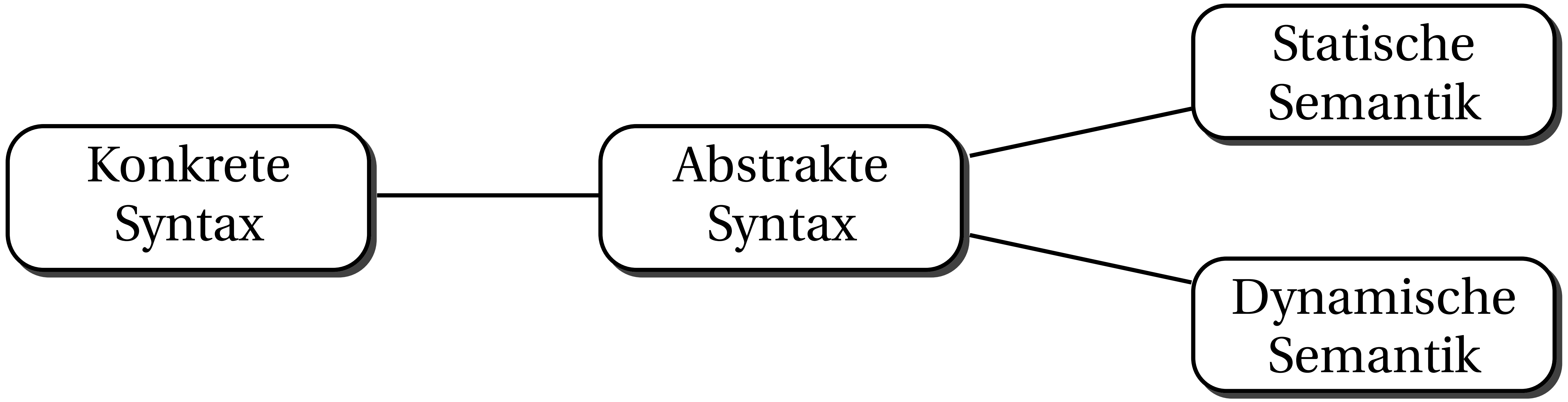
A: exception (Unbound True) ✗

B: exception (Error "T App1") ✗

C: exception (Error "T App2") ✓

D: exception (Error "T Opr") ✗

Statische Semantik



- Die **statische Semantik** prüft die **semantische Zulässigkeit** eines Ausdrucks.
  Sind alle auftretenden **Bezeichner** bekannt?
  Wenn ja, ist der Ausdruck **wohlgetypt**?
- Die **dynamische Semantik** wertet einen semantisch zulässigen Ausdruck aus.
  Terminiert die Auswertung?
  Wenn ja, mit welchem **Wert**?

Ausführung von Ausdrücken

- **Ausführung eines Ausdrucks** in einer **Umgebung** V:

3. …
4. …
5. …
6. Die Ausführung einer **Prozeduranwendung** $e_1\ e_2$ beginnt mit der Ausführung der Teilausdrücke e_1 und e_2 in V.
   Wenn diese die **Prozedur** p und den **Wert** v liefern, wird der **Prozeduraufruf** $p\ v$ ausgeführt.
   Die Prozeduranwendung liefert den so erhaltenen Wert.

Dynamische Semantik

- Eine **Wertumgebung** ist eine Funktion *V*, die endlich vielen Bezeichnern je einen Wert zuordnet:

$$\begin{array}{lllll} v & \in & \mathit{Val} & = & \mathbb{Z} \cup \mathit{Pro} & \text{Werte} \\ & & \mathit{Pro} & = & \mathit{Id} \times \mathit{Exp} \times \mathit{VE} & \text{Prozeduren} \\ V & \in & \mathit{VE} & = & \mathit{Id} \xrightarrow{\text{fin}} \mathit{Val} & \text{Wertumgebungen} \end{array}$$

Argument

- Die **dynamische Semantik von F** ist eine Menge

$$\mathit{DS} \subseteq \mathit{VE} \times \mathit{Exp} \times \mathit{Val}$$

die ein Tupel $\langle V,e,v \rangle$ genau dann enthält, wenn die Auswertung des Ausdrucks *e* in der Wertumgebung *V* mit dem **Wert** *v* **terminiert.**

- Notation: $V \vdash e \rhd v \quad :\Longleftrightarrow \quad \langle V, e, v \rangle \in \mathit{DS}$

Dynamische Semantik

Dfalse $$\frac{}{V \vdash \mathit{false} \rhd 0}$$ **Dtrue** $$\frac{}{V \vdash \mathit{true} \rhd 1}$$

Dnum $$\frac{z \in \mathbb{Z}}{V \vdash z \rhd z}$$ **Did** $$\frac{V\,x = v}{V \vdash x \rhd v}$$

D+ $$\frac{V \vdash e_1 \rhd z_1 \qquad V \vdash e_2 \rhd z_2 \qquad v = z_1 + z_2}{V \vdash e_1 + e_2 \rhd v}$$

D– $$\frac{V \vdash e_1 \rhd z_1 \qquad V \vdash e_2 \rhd z_2 \qquad z = z_1 - z_2}{V \vdash e_1 - e_2 \rhd z}$$

D∗ $$\frac{V \vdash e_1 \rhd z_1 \qquad V \vdash e_2 \rhd z_2 \qquad z = z_1 \cdot z_2}{V \vdash e_1 * e_2 \rhd z}$$

Dynamische Semantik

D≤ $$\frac{V \vdash e_1 \rhd z_1 \qquad V \vdash e_2 \rhd z_2 \qquad z = \text{if } z_1 \leq z_2 \text{ then } 1 \text{ else } 0}{V \vdash e_1 \leq e_2 \rhd z}$$

Diftrue $$\frac{V \vdash e_1 \rhd 1 \qquad V \vdash e_2 \rhd v}{V \vdash \mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3 \rhd v}$$

Diffalse $$\frac{V \vdash e_1 \rhd 0 \qquad V \vdash e_3 \rhd v}{V \vdash \mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3 \rhd v}$$

Dabs $$\frac{}{V \vdash \mathit{fn}\ x : t \Rightarrow e \rhd \langle x, e, V \rangle}$$

Dapp $$\frac{V \vdash e_1 \rhd \langle x, e, V' \rangle \qquad V \vdash e_2 \rhd v_2 \qquad V'[x := v_2] \vdash e \rhd v}{V \vdash e_1\, e_2 \rhd v}$$

Eigenschaften der dynamischen Semantik von F

Proposition 12.2 (Determinismus)

Sei $V \vdash e \rhd v$ *und* $V \vdash e \rhd v'$. *Dann* $v = v'$.

Satz 12.3 (Auswertbarkeit)

Sei $\emptyset \vdash e : t$. *Dann existiert genau ein Wert* v *mit* $\emptyset \vdash e \rhd v$.

Satz 12.4 (Typkorrektheit)

1. *Sei* $\emptyset \vdash e : int$ *und* $\emptyset \vdash e \rhd v$. *Dann* $v \in \mathbb{Z}$.
2. *Sei* $\emptyset \vdash e : bool$ *und* $\emptyset \vdash e \rhd v$. *Dann* $v \in \{0, 1\}$.

Evaluierung

eval: *value env* → *exp* → *value*

```
datatype value =
    IV    of int
  | Proc  of id * exp * value env

fun evalCon False  = IV 0
  | evalCon True   = IV 1
  | evalCon (IC x) = IV x
fun evalOpr Add (IV x1) (IV x2) = IV(x1+x2)
  | evalOpr Sub (IV x1) (IV x2) = IV(x1-x2)
  | evalOpr Mul (IV x1) (IV x2) = IV(x1*x2)
  | evalOpr Leq (IV x1) (IV x2) = IV(if x1<=x2 then 1 else 0)
  | evalOpr  _      _       _   = raise Error "R Opr"
fun eval f (Con c) = evalCon c
  | eval f (Id x) = f x
  | eval f (Opr(opr,e1,e2)) = evalOpr opr (eval f e1) (eval f e2)
```

Evaluierung

eval: *value env* $\rightarrow$ *exp* $\rightarrow$ *value*

```
fun eval f (Con c) = evalCon c
  | eval f (Id x) = f x
  | eval f (Opr(opr,e1,e2)) = evalOpr opr (eval f e1) (eval f e2)
  | eval f (If(e1,e2,e3)) = (case eval f e1 of
                   IV 1 => eval f e2
                 | IV 0 => eval f e3
                 |  _   => raise Error "R If")
  | eval f (Abs(x,t,e)) = Proc(x, e, f)
  | eval f (App(e1,e2)) = (case (eval f e1, eval f e2) of
                   (Proc(x,e,f'), v) =>  eval (update f' x v) e
                 | _ => raise Error "R App")
```

Rekursive Prozeduren

$rfn\, f(x:t):t' \Rightarrow e$

Beispiel:

```
rfn fac (n:int):int => if n<=0 then 1 else n*fac(n-1)
```

Statische Semantik:

$$\textbf{Srabs}\quad \frac{(T[f := t \to t'])[x := t] \vdash e : t'}{T \vdash rfn\, f(x:t):t' \Rightarrow e : t \to t'}$$

Dynamische Semantik:

$$Val = \mathbb{Z} \cup Pro \cup RPro$$

$$RPro = Id \times Id \times Exp \times VE$$

Prozedurbezeichner

Dynamische Semantik

$$\textbf{Drabs}\quad \frac{}{V \vdash \mathit{rfn}\ f\,(x : t) : t' \Rightarrow e \rhd \langle f, x, e, V\rangle}$$

$$\textbf{Drapp}\quad \frac{\begin{array}{lll} V \vdash e_1 \rhd v_1 & V \vdash e_2 \rhd v_2 & \\ v_1 = \langle f, x, e, V'\rangle & V'' = (V'[f := v_1])[x := v_2] & V'' \vdash e \rhd v \end{array}}{V \vdash e_1\, e_2 \rhd v}$$

Welche Eigenschaften gelten nach der Erweiterung von F um rekursive Abstraktionen?

A: Determinismus ✓

B: Auswertbarkeit ✗

C: Typkorrektheit ✓

www.prog1.saarland