

Teil X

Grammatiken

18. Syntax Hacks — A tribute to Haskell B. Curry

- ▶ Alphanumerische Bezeichner werden präfix notiert:

```
min 4711 815  
contains s ["f"; "female"]
```

- ▶ Symbolische Bezeichner werden infix notiert:

```
4711 + 815  
"Hello, " ^ "world!"
```

- ▶ Im Folgenden:
 - ▶ Wie können wir alphanumerische Bezeichner infix schreiben?
 - ▶ Wie können wir symbolische Bezeichner präfix schreiben?
 - ▶ (Damit Sie für den nächsten „Obfuscated F# Contest“ gewappnet sind.)

18. Syntax Hacks: infix nach präfix

- ▶ Ein symbolischer Bezeichner verliert seinen Infixstatus, wenn er in Klammern eingeschlossen wird.

```
(+) 4711 815  
(^) "Hello, " "world!"
```

- ▶ Nützlich, um Funktionen höherer Ordnung mit Argumenten zu versorgen:

```
merge-sort (≥) [4; 7; 1; 1]  
List.map2 (+) [1; 2; 3] [1; 2; 1]
```

- ▶ Symbolische Bezeichner können auch selbst definiert werden:

```
Mini> let (*) a b = a + b  
Mini> 4 * 7 + 11  
22
```

18. Syntax Hacks: Empfehlungen

Empfehlungen:

- ▶ $f : A \rightarrow A \rightarrow A$ assoziativ: infix;
- ▶ $f : A \rightarrow A \rightarrow A$ kommutativ: spiegelsymmetrischer Operator.

The good, the bad and the ugly:

- ▶ good: $+$, $*$ (assoziativ und kommutativ);
- ▶ good: \gg , \ll , $@$ (nicht kommutativ);
- ▶ bad: \wedge , \circ (nicht kommutativ);
- ▶ bad: $\%$ (nicht assoziativ);
- ▶ ugly: $-$, \div (nicht assoziativ).

18. Syntax Hacks: präfix nach infix

Die umgekehrte Richtung, präfix nach infix, wird von F# von Haus aus nicht unterstützt.

Idee: mit speziellen Infixoperatoren simulieren:

```
Mini> let x = [1; 2; 3]
```


```
Mini> let y = [1; 2; 1]
```

```
Mini> x <List.map2 (+)> y
```

```
[2; 4; 4]
```

```
Mini> x <List.map2 (*)> x <List.map2 (+)> y
```

```
[2; 6; 10]
```

 Innerhalb der „Infixklammern“ darf ein beliebiger Ausdruck stehen.

18. Syntax Hacks: präfix nach infix

Wie müssen \langle und \rangle definiert werden? Definition gesucht, so dass:

$$a \langle f \rangle b = f a b$$

Konkrete Definition hängt von der Klammerung ab:

$ \begin{aligned} & a \langle f \rangle b \\ = & \quad \{ \text{Klammerung links} \} \\ & (a \langle f \rangle b) \\ = & \quad \{ \text{definiere } x \langle f = f x \} \\ & f a \rangle b \\ = & \quad \{ \text{definiere } g \rangle y = g y \} \\ & f a b \end{aligned} $	$ \begin{aligned} & a \langle f \rangle b \\ = & \quad \{ \text{Klammerung rechts} \} \\ & a \langle (f \rangle b) \\ = & \quad \{ \text{definiere } f \rangle y = \text{fun } x \rightarrow f x y \} \\ & a \langle (\text{fun } x \rightarrow f x b) \\ = & \quad \{ \text{definiere } x \langle g = g x \} \\ & f a b \end{aligned} $
---	--

Links: \langle ist Postfixapplikation und \rangle ist Präfixapplikation.

Rechts: etwas komplizierter, da f das zweite Argument zuerst erhält.

18. Syntax Hacks: präfix nach infix

Ist der „Infixoperator“ $\langle f \rangle$ selbst links- oder rechtsassoziierend?

Hängt von der Bindungsstärke und Assozierung der verwendeten Operatoren \langle und \rangle ab.

- ▶ gleiche Priorität, beide linksassoziierend:

$$a \langle f \rangle b \langle g \rangle c = (((a \langle f \rangle b) \langle g \rangle c)$$

Damit ist auch $\langle f \rangle$ linksassoziierend.

- ▶ gleiche Priorität, beide rechtsassoziierend:

$$a \langle f \rangle b \langle g \rangle c = a \langle f \rangle (b \langle g \rangle c)$$

Damit ist auch $\langle f \rangle$ rechtsassoziierend.

18. Syntax Hacks: präfix nach infix

- ▶ \langle bindet stärker; \rangle ist linksassoziiierend: erlaubt keine passende Definition von \langle und \rangle .

- ▶ \langle bindet stärker; \rangle ist rechtsassoziiierend:

$$a \langle f \rangle b \langle g \rangle c = (a \langle f \rangle (b \langle g \rangle c))$$

Damit ist auch $\langle f \rangle$ rechtsassoziiierend.

- ▶ \rangle bindet stärker; \langle ist linksassoziiierend:

$$a \langle f \rangle b \langle g \rangle c = (a \langle (f \rangle b) \rangle (g \rangle c))$$

Damit ist auch $\langle f \rangle$ linksassoziiierend.

- ▶ \rangle bindet stärker; \langle ist rechtsassoziiierend: erlaubt keine passende Definition von \langle und \rangle .

18. Syntax Hacks: präfix nach infix

Wie bekomme ich heraus, wie geklammert wird?

Dokumentation studieren; *oder* ausprobieren.

```
Mini> let (<) a b = (a, "<.", b)
```

```
Mini> let (>) a b = (a, ">.", b)
```

```
Mini> "a" < "f" > "b"
```

```
(( "a", "<.", "f"), ">.", "b")
```

```
Mini> "a" < "f" > "b" < "g" > "c"
```

```
((((( "a", "<.", "f"), ">.", "b"), "<.", "g"), ">.", "c"))
```

```
Mini> let (^<) a b = (a, "^<", b)
```

```
Mini> let (^>) a b = (a, "^>", b)
```

```
Mini> "a" ^< "f" ^> "b"
```

```
("a", "^<", ("f", "^>", "b"))
```

```
Mini> "a" ^< "f" ^> "b" ^< "g" ^> "c"
```

```
("a", "^<", ("f", "^>", ("b", "^<", ("g", "^>", "c"))))
```

18. Syntax Hacks: präfix nach infix

Beispiele:

Linksassoziiierend (gleiche Bindungsstärke):

let ($<$) $a f = f a$

let ($>$) $f b = f b$

Linksassoziiierend ($/>$ bindet stärker als $</$):

let ($</$) $a f = f a$

let ($/>$) $f b = \mathbf{fun} a \rightarrow f a b$

Rechtsassoziiierend (gleiche Bindungsstärke):

let ($\wedge<$) $a f = f a$

let ($\wedge>$) $f b = \mathbf{fun} a \rightarrow f a b$