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:

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

The good, the bad and the ugly:

- good: $+$, $*$ (assoziativ und kommutativ);
- good: $\gg$, $\ll$, @ (nicht kommutativ);
- bad: ^, $\circ$ (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 $<.$ und $.>$ definiert werden? Definition gesucht, so dass:

$$a <. f .> b = f\ a\ b$$

Konkrete Definition hängt von der Klammerung ab:

$$\begin{array}{ll} & a <. f .> b \\ = & \quad \{ \text{ Klammerung links } \} \\ & (a <. f) .> b \\ = & \quad \{ \text{ definiere } x <. f = f\ x \ \} \\ & f\ a .> b \\ = & \quad \{ \text{ definiere } g .> y = g\ y \ \} \\ & f\ a\ b \end{array} \qquad \begin{array}{ll} & a <. f .> b \\ = & \quad \{ \text{ Klammerung rechts } \} \\ & a <. (f .> b) \\ = & \quad \{ \text{ definiere } f .> y = \textbf{\textit{fun}}\ x \to f\ x\ y \ \} \\ & a <. (\textbf{\textit{fun}}\ x \to f\ x\ b) \\ = & \quad \{ \text{ definiere } x <. g = g\ x \ \} \\ & f\ a\ b \end{array}$$

Links: $<.$ ist Postfixapplikation und $.>$ ist Präfixapplikation.
Rechts: etwas komplizierter, da f das zweite Argument zuerst erhält.

18. Syntax Hacks: präfix nach infix

Ist der „Infixoperator“ $<.f.>$ selbst links- oder rechtsassoziierend?

Hängt von der Bindungsstärke und Assozierung der verwendeten Operatoren $<.$ und $.>$ ab.

- gleiche Priorität, beide linksassoziierend:

  $$a <. f .> b <. g .> c = (((a <. f) .> b) <. g) .> c$$

  Damit ist auch $<.f.>$ linksassoziierend.
- gleiche Priorität, beide rechtsassoziierend:

  $$a <. f .> b <. g .> c = a <. (f .> (b <. (g .> c)))$$

  Damit ist auch $<.f.>$ rechtsassoziierend.

18. Syntax Hacks: präfix nach infix

- $<.$ bindet stärker; $.>$ ist linkssassoziierend: erlaubt keine passende Definition von $<.$ und $.>$.
- $<.$ bindet stärker; $.>$ ist rechtsassoziierend:

  $$a <. f .> b <. g .> c = (a <. f) .> ((b <. g) .> c)$$

  Damit ist auch $<.f.>$ rechtsassoziierend.
- $.>$ bindet stärker; $<.$ ist linkssassoziierend:

  $$a <. f .> b <. g .> c = (a <. (f .> b)) <. (g .> c)$$

  Damit ist auch $<.f.>$ linkssassoziierend.
- $.>$ bindet stärker; $<.$ ist rechtsassoziierend: erlaubt keine passende Definition von $<.$ und $.>$.

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:

Linkssassoziierend (gleiche Bindungsstärke):

```
let (<.) a f = f a
let (.>) f b = f b
```

Linkssassoziierend (/> bindet stärker als </):

```
let (</) a f = f a
let (/>) f b = fun a → f a b
```

Rechtsassoziierend (gleiche Bindungsstärke):

```
let (^<) a f = f a
let (^>) f b = fun a → f a b
```