Teil X

Grammatiken

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

- Alphanumerische Bezeichner werden präfix notiert:

$$\mathit{min}\ 4711\ 815$$
$$\mathit{contains}\ s\ [\texttt{"f"}; \texttt{"female"}]$$

- 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 $\lessdot$ und $\gtrdot$ definiert werden? Definition gesucht, so dass:

$$a \lessdot f \gtrdot b = f\ a\ b$$

Konkrete Definition hängt von der Klammerung ab:

$$\begin{aligned}
& a \lessdot f \gtrdot b \\
= & \quad \{ \text{ Klammerung links } \} \\
& (a \lessdot f) \gtrdot b \\
= & \quad \{ \text{ definiere } x \lessdot f = f\ x \ \} \\
& f\ a \gtrdot b \\
= & \quad \{ \text{ definiere } g \gtrdot y = g\ y \ \} \\
& f\ a\ b
\end{aligned}$$

$$\begin{aligned}
& a \lessdot f \gtrdot b \\
= & \quad \{ \text{ Klammerung rechts } \} \\
& a \lessdot (f \gtrdot b) \\
= & \quad \{ \text{ definiere } f \gtrdot y = \textbf{\textit{fun}}\ x \to f\ x\ y \ \} \\
& a \lessdot (\textbf{\textit{fun}}\ x \to f\ x\ b) \\
= & \quad \{ \text{ definiere } x \lessdot g = g\ x \ \} \\
& f\ a\ b
\end{aligned}$$

Links: $\lessdot$ ist Postfixapplikation und $\gtrdot$ 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

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

$$a <\!\cdot f \cdot\!> b <\!\cdot g \cdot\!> c = (a <\!\cdot f) \cdot\!> ((b <\!\cdot g) \cdot\!> c)$$

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

$$a <\!\cdot f \cdot\!> b <\!\cdot g \cdot\!> c = (a <\!\cdot (f \cdot\!> b)) <\!\cdot (g \cdot\!> c)$$

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

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):

$$\textbf{let}\ (<.)\ a\ f = f\ a$$
$$\textbf{let}\ (.>)\ f\ b = f\ b$$

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

$$\textbf{let}\ (</)\ a\ f = f\ a$$
$$\textbf{let}\ (/>)\ f\ b = \textbf{fun}\ a \to f\ a\ b$$

Rechtsassoziierend (gleiche Bindungsstärke):

$$\textbf{let}\ (\hat{}<)\ a\ f = f\ a$$
$$\textbf{let}\ (\hat{}>)\ f\ b = \textbf{fun}\ a \to f\ a\ b$$