Beispiel: Übersetzung und Optimierung von list comprehensions

- Auf die Frage:
  Kann man list comprehensions allgemein durch Aufrufe von filter, map, concat ersetzen?
- … könnte man ja mal schauen, was der Compiler tut:

```
[ e | x ← xs ]       ↦   map (\x → e) xs
[ e | b ]            ↦   if b then [ e ] else [ ]
[ e | b, Q ]         ↦   if b then [ e | Q ] else [ ]
[ e | x ← xs, Q ]    ↦   concat (map (\x → [ e | Q ]) xs)
```

- Kein filter? Dafür „umständliche“ Ausdrücke, etwa:

```
concat (map (\x → concat (map (\y → if x `mod` y > 0 then [ (x, y) ] else [ ]) [1 .. x])) [1 .. 100])
```

für:

```
[ (x, y) | x ← [1 .. 100], y ← [1 .. x], x `mod` y > 0 ]
```

- Kann man da was mit Post-Processing tun?

Beispiel: Übersetzung und Optimierung von list comprehensions

- Ja, man kann. Allgemein gilt:

```
concat (map (\y → if p y then [ f y ] else [ ]) ys)
```

ist äquivalent zu:

```
map f (filter p ys)
```

- Folglich ist:

```
concat (map (\x → concat (map (\y → if x `mod` y > 0 then [ (x, y) ] else [ ]) [1 .. x])) [1 .. 100])
```

äquivalent zu:

```
concat (map (\x → map (\y → (x, y)) (filter (\y → x `mod` y > 0) [1 .. x])) [1 .. 100])
```

- Aber, es wäre ratsam, die obige allgemeine Aussage zunächst zu beweisen!