Abstrakte Datentypen: `abstype`

Abstrakte Datentypen verbergen wie opake Struktureinschränkungen die Wertkonstruktoren:

⟨*abstype*⟩≡

```
abstype ''a bag = Bag of (''a * int) list
   with
     val empty = Bag []
     fun add (a, Bag []) = Bag [(a,1)]
       | add (a, Bag ((b,n)::rest)) =
         if a=b then Bag((b,n+1)::rest)
         else
             let val (Bag bs) = add (a, Bag rest)
             in
                 Bag ((b,n)::bs)
             end
     fun count (a, Bag []) = 0
       | count (a, Bag ((b,n)::rest)) =
         if a=b then n else count (a,Bag rest)
     fun union (Bag []) bs = bs
       | union bs (Bag []) = bs
       | union (Bag ((a,n)::rest)) (Bag ((b,m)::bs)) =
         if a=b then union (Bag rest) (Bag ((a,n+m)::bs))
         else
             let val (Bag newbs) =
                 union (Bag [(a,n)]) (Bag bs)
             in
                 union (Bag rest) (Bag ((b,m)::newbs))
             end
   end
```

Die Auswertung der Deklaration ergibt:

⟨*abstype*⟩+≡
```
type 'a bag
val empty = - : 'a bag
val add = fn : ''a * ''a bag -> ''a bag
val count = fn : ''a * ''a bag -> int
val union = fn : ''a bag -> ''a bag -> ''a bag
```

Einen `''a bag` kann man nur mit `empty` und `add` erzeugen; der *Typkonstruktor* `Bag` ist außerhalb der Deklaration nicht verfügbar:

⟨⟩+≡
```
- Bag [];
Error: unbound variable or constructor: Bag
```