Pohdittavaa

K: Listojen loppua ilmaisee yleensä jokin erityinen alkio, esim. `[]`. Mikä on sen tyyppi?

V: `[`α`]`

K: Osaatko löytää funktion, jonka tyyppi on $\alpha \rightarrow \beta$?

V: Tämä edellyttää, että kielessä on ohjelmoijalle näkyvänä vakiolausekkeena arvo, jonka tyyppi on kaikkien muiden tyyppien alityyppi. Näin näyttää olevan varsin harvassa staattisesti tyypitetyssä kielessä.

ML:ssä "lähimmäksi" pääsee yksiargumenttinen funktio, joka palauttaa tyhjän listan. Sen tyyppi on $\alpha \rightarrow$`[`β`]`.

K: Mikä on τ:n position varianssi tyyppikaavassa
($\tau \rightarrow$`Int`) $\rightarrow$`Int` ?

V: Kovariantti.

K: Kehitä esitettyjä alityyppisääntöjä vastaava tyyppikielioppi, jossa tyyppimuuttuja voi esiintyä vain kovarianteissa positioissa.

V: Esim.
cov_type: basic_type | cov_constructed_type | type_var
contrav_type: basic_type | contrav_constructed_type
cov_constructed_type: contrav_type → cov_type | ...
contrav_constructed_type: cov_type → contrav_type | ...
basic_type: `Int` | `String` | ...

K: Olkoon tyyppikaavojen T(τ) ja S(τ) varianssit tunnetut τ:n suhteen. Mitkä ovat tyyppikaavojen <T(τ),S(τ)>, T(τ) $\rightarrow$ S(τ), ja T(S(τ)) varianssit τ:n suhteen?

V: Allaolevista taulukoista aina ylimmän sopivan sarakkeen mukaisesti.

T(τ)	S(τ)	<T(τ),S(τ)>
inv	inv	inv
{ inv, cov }	{ inv, cov }	cov
{ inv, contrav }	{ inv, contrav }	contrav
_	_	biv

T(τ)	S(τ)	T(τ) $\rightarrow$ S(τ)
inv	inv	inv
{ inv, contrav }	{ inv, cov }	cov
{ inv, cov }	{ inv, contrav }	contrav
_	_	biv

T(τ)	S(τ)	T(S(τ))
inv	_	inv
_	inv	inv
cov	cov	cov
cov	contrav	contrav
contrav	cov	contrav
contrav	contrav	cov
_	_	biv

K: Onko esimerkiksi C:n yhteenlasku `1 + 2.0` esimerkki koersiosta vai ylikuormituksesta?

V: Riippuu siitä, mitä kääntäjä tuottaa. Jos prosessori osaa laskea yhdellä käskyllä yhteen liukuluvun ja kokonaisluvun, on kyseessä ylikuormitus. Jos koodi ensin muuttaa `1`:n liukuluvuksi ja laskee vasta sitten yhteen `2.0`:n kanssa, on kyseessä koersio.

K: Ovatko C++:n templatet parametrista vai ad-hoc–polymorfiaa?

V: Ilmaisuvoimaltaan ne ovat parametrista polymorfiaa, mutta yleensä toteutettu tavalla, joka tuottaa yhtä paljon koodia kuin ad-hoc–polymorfia.