Aihe: Lisp-tulkki

Suunnitellaan ja toteutetaan toimiva, yksinkertainen Lisp-tulkki Java-kielellä. Tulkki sisältää Lisp-kieliperheeseen kuuluvan Scheme-kielen olennaisimmat osat: read-eval-print -silmukan sekä kaikki Daniel Friedmanin “The Little Schemer" -kirjassa luodun Schemen toiminnallisuuksista ja proseduureista. Funktioilla on kaikkien Lisp-varianttien mukaisesti first-class -ominaisuus siten, että niitä voidaan antaa argumentteina toisille funktioille. Condilla ei tätä ominaisuutta ole, johtuen siitä, että Con ei ole määritelty funktioksi SkeletonLispissä.

Tulkki sisältää tuen myös anonyymien funktioiden luontiin ja käyttöön lambda-käskyn avulla. Tulkki ei sisällä call-with-current-continuationeja eikä makroja. Tulkki sisältää ennaltamääritellyn globaalin ympäristön, primitiiviympäristön. sekä mahdollisuuden laajentaa paikallisia ympäristöjä funktioiden ja arvojen määrittelyjä sekä käyttöä varten.

Käyttäjät: Lisp-ohjelmoija

Käyttäjien toiminnot
Syöttää tulkille SkeletonLisp-lauseita (L-expressions), joiden arvoja tulkki evaluoi.

hyväksyttäviä Lisp-lauseita ovat:
* NIL: [“NIL"]
* atomit : [‘A, ‘ATOMI, ‘TERVE-76]
* kokonaisluvut: [56, -254 : rajoituksena samat kuin Javan int-tyypin rajoitukset]
* ID-t: [F, ADD, SQRT, USER-DEFINED-ID], joitakin rajoituksia sisältyy Id:itten käyttöön, kuten se, että ne eivät see alkaa numeromerkillä, eikä bindaamattomia ID:itä saa laitaa mm. listaan/pariin alkioksi.
* lambda-lauseet, jotka voivat olla muotoa:
  (LAMBDA (var-1 ... var-n) body) tai:
* “applikaatiot” (applications), jotka voivat olla jotakin seuraavista:
  1. (Id val-1 ... val-n)
  2. (lambda-expression val-1 ... val-n)
  3. (application val-1 ... val-n)
* cond-lause:
  (COND (ehto-1 arvo-1)
  ...
  (ehto-n arvo-n))

[COND-lauseen ei tarvitse sisältää else-osiota [Common Lispin tapaan “#T” - SkeletonLispissä], mutta mikäli tätä ei ole, eikä mikään ehdoista toteudu, ilmoittaa tulkki virhetilanteesta käyttäjälle, eikä lausetta pystytä evaluoimaan.

“LIST” ja “CONS" -primitiivien avulla luodaan uusia LPair-tyyppejä, jotka sisältävät kaksi alkiota:
((CAR LExp) (CDR LExp)), ja LPair:ien tulostusmuoto riippuu siitä, minkälainen lause (CDR LExp) on; jos se on jotain muuta kuin toinen pari, tai nil, tulostusmuoto on:
((CAR LExp) **.** (CDR LExp)),
jos se on toinen LPair, tulostusmuoto on:

((CAR LExp) (CDR LExp)),
muuten (NIL:llä) sen muoto on:
((car LExp))
“LIST”-primitiivi luo aina listan, jonka CDR on joko toinen lista, tai NIL