This page is intentionally left blank.

Kazimir Majorinc

Planner (I).

Matchless

Povijest Lispa 39.

Razmjena vještina
Hacklab u mami
16. studeni 2013.

Planner - programski jezik (?), dijalekt Lispa (?), *proširenje Lispa*, često se spominje u diskusijama o Lispu. Autor: **Carl Hewitt**. Desetak memoa, konferencijskih članaka. Neuspjeh?

Hewitt, *Planner - A language for proving theorems*, AIM-131, 1967.

Hewitt, *Planner - Functional Abstraction in Lisp and Planner*, AIM-151, 1968.

Hewitt, *Planner - A language for proving theorems in robots,* IJCAI, 1969

Sussman & Winograd, *Micro-planner*, AIM-203,**1970**

Hewitt, *Description and theoretical analysis of Planner, A language for proving theorems and manipulatings models in robots*, doktorska radnja, 1972, 500 strana

„Although we say that PLANNER is a programming language, we do not mean to imply that it is a purely procedural language like the lambda calculus in pure LISP. PLANNER is different from pure LISP in that function calls can be made indirectly through recommendations specifying the form of the data on which the function is supposed to work. In such a call the actual name of the called function is usually unknown.“

Osnovna ideja jezika je dualitet imperativnih i deklarativnih naredbi. Na primjer

```
(implies a b).
```

je deklarativna naredba, govori o odnosu a i b, ali i uputa za izračunavanje b.

MATCHLESS

programski jezik koji služiti kao osnova Plannera. Najvažnija je funkcija za pridruživanje **assign?** U slučaju uspjeha vraća **t**, u slučaju neuspjeha **()**.

```
{prog ((a ptr) (h atom) (c seg))
      {assign? ($_a k $_h $_c) ((l) k b l a)}}
     a:=(l)
     h:=b
     c:= - l a -
```

```
{prog ((c seg) (h atom) (a ptr))
      {assign? ($_c $_h k $_a) (a l b k q)}}
     c:= -a l-
     h:= b
     a:= q

{prog ((first ptr) (middle seg) (last ptr))
  {assign? ($_first $_middle $_last) (1,2,3,4)}}
  first := 1
  middle := -2 3-
  last := 4
```

```
{prog ((a ptr) (b ptr))
      {assign? ($_a $_b) (d)}}
```

neuspjeh

```
{prog ((a atom))
      {assign? $_a (1 2)}}
```

neuspjeh.

$$a znači vrijednost a

```
{prog ((a seg))
      {assign? ($_a $$a) (1 2 3 1 2 3)}}
```

a := -1 2 3-

```
{prog ((a seg) (b seg))
{assign? ($_a x $$a $-b) (a b x d x a b x d q)}}
```

```
a := -a b x d-
b := -q-
```

Izraz oblika **$?a** pridružuje vrijednost **a** samo ako nije prethodno pridružena neka različita vrijednost.

```
{prog ((a ptr))
  {assign? $?a 3}}
a := 3
```

```
{prog (((a 5) ptr))
  {assign? $?a 4}}
```

a ima vrijednost 5, zato se pridruživanje ne može izvršiti. Neuspjeh.

```
{prog ((a seg))
  {assign? ($_a $?a) (1 2 3 2 1)}}
```

Neuspjeh.

```
{prog ((a ptr) (b ptr) (c ptr))
  {assign? (a <conj $_a $_b> $_c) (a 1 2 3)}}
a := -1 2-
b := -1 2-
c := -3-

{prog ((x seg) (c seg))
  {assign? ($_x <disj (3) (2)> $_c) (a 1 2 3)}}
x := -a 1-
c := -3-
```

Funkcija koja provjerava je li lista palindrom.

```
(def palindrome
  (kappa ((x))
    {disj
       ()
       {block ((x ptr)) ($_x <palindrome> $$x)}}))
```

kappa - lambda u Matchlessu

block - pojednostavljeni assign

kraj.