This page is intentionally left blank.

Kazimir Majorinc

PLANNER u 1972

Povijest Lispa 45.

Razmjena vještina
Hacklab u mami
18. siječnja 2014.

Formalizam za PLANNER uključuje unificirani skup „problem solving primitives“ koji se izvršavaju pod „multiprocesnom backtrack kontrolnom strukturom.“ Sam je formalizam nezavisan od domene za rješavanje problema.

Primitivni pojmovi formalizma donose defaultne odluke o načinu izračunavanja ako /kada nije dodana specifična informacija. Stvar je principa da primitivne operacije dozvoljavaju dodavanje specifičnih informacija koje olakšavaju izračunavanje.

Iako postoje teorije da je korištenje „side efekata“ u suprotnosti sa dobrim dizajnom jezika, njihovo korištenje u PLANNER-u se pokazalo dobrim.

Ime PLANNER rezultat je želje za stvaranjem formalizma u kojem je jednostavno izraziti „plan akcije.“ Konstruirati plan akcije je isto što i konstruirati PLANNER teorem. Miješanje planiranja i dedukcije je vrlo jednostavno. Uvjetni planovi su eksplicitno ponuđeni, npr. mogućnost bactkrackinga u slučaju neuspjeha (engl. failure.)

Pogledajmo, primjerice, izjavu koja usklađuje uzorak **[IMPLIES |x| |y|].** Ova izjava ima nekoliko imperativnih upotreba:

(1) Ako možemo izvesti |x|, onda možemo izvesti |y|.
U PLANNERU se ova izjava (statement) može izraziti sa

```
<ANTECENDENT [ ] |x| <ASSERT |y|> >
```

(2) Ako želimo izvesti |y| onda možemo postaviti |x| kao podcilj.

```
<CONSEQUENT [ ] |x| <GOAL |x|> <ASSERT |y|> >
```

(3) NOT |x| i |y| su alternative

```
 <CLAUSE [ ] [NOT |x|] |y|]>
```

„Hierarchical Backtrack Control Structure“

PLANNER koristi kontrolnu strukturu u kojoj je hijerarhija poziva sačuvana tako da se izračunavanje može vršiti „backtracking“ (vraćanje, odustajanje).

Primitivne funkcije su FAIL i FAILPOINT. Forma <FAIL> izaziva neuspjeh koji izaziva vraćanje do posljednje FAILPOINT forme.

```
<prog [[x 3]]
      <+ <prog foo []
               <failpoint []
                          .x
                          [¬"optional"]
                          <.foo <_ :x 4>>
                          ; „ako je uklapanje x i 4 moguće
                          ;  postavlja x na 4 i vraća 4“
                          ;"exit .foo with 4"
                          >>
         ;"the first time through the above
           expression has .x as its value"
         <cond [<is? 3 .x> <fail>]
               [¬"else" 5]>>>
```

Ovaj kod se izračunava u <+ 4 5>. Forma <UNIQUE> izaziva neuspjeh ako PLANNER ustanovi da cilj koji trenutno pokušava dokazati nije jedinstven, pokušava ga dokazati i negdje drugdje, u nekom drugom procesu. **Planner sve pamti.**

Multiprocesiranje

Planner podržava multiprocesiranje. Proces je u PLANNERU tip. Po definiciji „program counter together with stack“. Najvažnije primitivne operacije su:

<STEP |p| |n| |condition|>

izvršava proces |p| za |n| elementarnih koraka, a završava odmah kad se ispuni |condition |. |n| može biti negativan.

<PROCESS |f| … >

|f| može biti funkcija, ali i izraz; Ako je funkcija onda <<PROCESS sin> 3>

<CALL … <|p| -arguments-> …>
nastavlja prekinuti proces

<STOP|p|>

<SUSPEND … |p|>

kraj