Informe Compiladors

José Ruiz Bravo, 123456789 <joseruizbravo@gmail.com>,
Biel Moyà Alcover, 43142617E <bilibiel@gmail.com>,
Álvaro Medina Ballester, 43176576X <alvaro@comiendolimones.com>

20 de novembre de 2009

Resum

Compilador *compilemon* creat amb el llenguatge Ada. Està composat per un subconjunt bàsic d'instruccions en Ada.

1 Anàlisi Lèxica

1.1 Descripció del lèxic: *compilemon.l*

```
1  -- Macros
2
3  lletra  [A-Za-z]
4
5  digit   [0-9]
6
7  separadors      [\n\b\t\f]
8
9  caracter        \'[^\'\n\t]\'
10
11
12 %%
13
14
15 -- Paraules clau
16
17 procedure       {mt_atom(tok_begin_line, tok_begin_col,
18                 yylval); return pc_procedure;}
19
20 begin   {mt_atom(tok_begin_line, tok_begin_col, yylval);
```

```
21          return pc_begin;}
22
23  while   {mt_atom(tok_begin_line, tok_begin_col, yylval);
24          return pc_while;}
25
26  if      {mt_atom(tok_begin_line, tok_begin_col, yylval);
27          return pc_if;}
28
29  else    {mt_atom(tok_begin_line, tok_begin_col, yylval);
30          return pc_else;}
31
32  end     {mt_atom(tok_begin_line, tok_begin_col, yylval);
33          return pc_end;}
34
35  do      {mt_atom(tok_begin_line, tok_begin_col, yylval);
36          return pc_do;}
37
38  constant        {mt_atom(tok_begin_line, tok_begin_col, yylval);
39          return pc_constant;}
40
41  type    {mt_atom(tok_begin_line, tok_begin_col, yylval);
42          return pc_type;}
43
44  array   {mt_atom(tok_begin_line, tok_begin_col, yylval);
45          return pc_array;}
46
47  record  {mt_atom(tok_begin_line, tok_begin_col, yylval);
48          return pc_record;}
49
50  is      {mt_atom(tok_begin_line, tok_begin_col, yylval);
51          return pc_is;}
52
53  then    {mt_atom(tok_begin_line, tok_begin_col, yylval);
54          return pc_then;}
55
56  not     {mt_atom(tok_begin_line, tok_begin_col, yylval);
57          return pc_not;}
58
59  in      {mt_atom(tok_begin_line, tok_begin_col, yylval);
60          return pc_in;}
61
62  out     {mt_atom(tok_begin_line, tok_begin_col, yylval);
63          return pc_out;}
```

```
64
65 new      {mt_atom(tok_begin_line, tok_begin_col, yylval);
66          return pc_new;}
67
68 null     {mt_atom(tok_begin_line, tok_begin_col, yylval);
69          return pc_null;}
70
71 of       {mt_atom(tok_begin_line, tok_begin_col, yylval);
72          return pc_of;}
73
74 mod      {mt_atom(tok_begin_line, tok_begin_col, yylval);
75          return pc_mod;}
76
77 range    {mt_atom(tok_begin_line, tok_begin_col, yylval);
78          return pc_range;}
79
80 and      {mt_atom(tok_begin_line, tok_begin_col, yylval);
81          return pc_or;}
82
83 or       {mt_atom(tok_begin_line, tok_begin_col, yylval);
84          return pc_and;}
85
86
87 --Simbols
88
89 ":="     {mt_atom(tok_begin_line, tok_begin_col, yylval);
90          return s_assignacio;}
91
92 ":"      {mt_atom(tok_begin_line, tok_begin_col, yylval);
93          return s_dospunts;}
94
95 ";"      {mt_atom(tok_begin_line, tok_begin_col, yylval);
96          return s_final;}
97
98 ","      {mt_atom(tok_begin_line, tok_begin_col, yylval);
99          return s_coma;}
100
101 "("     {mt_atom(tok_begin_line, tok_begin_col, yylval);
102         return s_parentesiobert;}
103
104 ")"     {mt_atom(tok_begin_line, tok_begin_col, yylval);
105         return s_parentesitancat;}
106
```

```
107 ".."     {mt_atom(tok_begin_line, tok_begin_col, yylval);
108         return s_puntsrang;}
109
110 "."      {mt_atom(tok_begin_line, tok_begin_col, yylval);
111         return s_puntrec;}
112
113
114 --Operadors
115
116 "<"      {mt_atom(tok_begin_line, tok_begin_col, yylval);
117         return op_menor;}
118
119 "<="     {mt_atom(tok_begin_line, tok_begin_col, yylval);
120         return op_menorigual;}
121
122 ">="     {mt_atom(tok_begin_line, tok_begin_col, yylval);
123         return op_majorigual;}
124
125 ">"      {mt_atom(tok_begin_line, tok_begin_col, yylval);
126         return op_major;}
127
128 "="      {mt_atom(tok_begin_line, tok_begin_col, yylval);
129         return op_igual;}
130
131 "/="     {mt_atom(tok_begin_line, tok_begin_col, yylval);
132         return op_distint;}
133
134 "+"      {mt_atom(tok_begin_line, tok_begin_col, yylval);
135         return op_suma;}
136
137 "-"      {mt_atom(tok_begin_line, tok_begin_col, yylval);
138         return op_resta;}
139
140 "*"      {mt_atom(tok_begin_line, tok_begin_col, yylval);
141         return op_multiplicacio;}
142
143 "/"      {mt_atom(tok_begin_line, tok_begin_col, yylval);
144         return op_divisio;}
145
146
147
148 --EXPRESSIONS REGULARS
149
```

```
150 --Digit
151
152 {digit}+         {mt_numero(tok_begin_line, tok_begin_col,
153                  yytext, yylval); return const;}
154
155
156 --Lletra
157
158 {caracter}       {mt_caracter(tok_begin_line, tok_begin_col,
159                  yytext, yylval); return const;}
160
161
162 --String
163
164 \"[^\"\n\t]*\"   {mt_string(tok_begin_line, tok_begin_col,
165                  yytext, yylval); return const;}
166
167
168 --Identificador
169
170 {lletra}({digit}|{lletra})* {mt_identificador(tok_begin_line,
171                             tok_begin_col, yytext, yylval);
172                             return id;}
173
174
175
176 --Comentaris
177
178 "-""-"[^\n]*     {null;}
179
180
181 --Separadors
182
183 " "              {null;}
184
185 {separadors}*    {null;}
186
187
188 --Error
189
190 .        {return error;}
191
192
```

```
193
194 %%
195
196
197
198 with     decls.d_taula_de_noms,
199                 --Ada.Text_IO,
200                 --Ada.Command_Line,
201                 --decls.dgenerals,
202                 d_token,
203                 --decls.tn,
204                 --compilemon_io,
205                 d_atribut;
206         --pk_usintactica_tokens;
207
208
209 use      decls.d_taula_de_noms,
210         --Ada.Text_IO,
211         --Ada.Command_Line,
212         --decls.dgenerals,
213         d_token,
214         --decls.tn,
215         --compilemon_io,
216         d_atribut;
217         --pk_usintactica_tokens;
218
219
220 package u_lexica is
221
222         yylval: atribut;
223         tn : taula_de_noms;
224         function YYLex return token;
225         tok_begin_line : integer;
226         tok_begin_col : integer;
227
228 end u_lexica;
229
230
231
232 package body u_lexica is
233
234 ##
235
```

```
236 begin
237
238         tbuida(tn);
239
240 end u_lexica;
```

2 Taula de noms

2.1 Fitxer *decls-d_taula_de_noms.ads*

```
1  -- ------------------------------------------------
2  --  Paquet de declaracions de la taula de noms
3  -- ------------------------------------------------
4  --  Versio  :   0.1
5  --  Autors  :   Jose Ruiz Bravo
6  --              Biel Moya Alcover
7  --              Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --  Especificacio de l'estructura necessaria
10 -- per el maneig de la taula de noms i dels metodes
11 -- per tractar-la.
12 --
13 -- ------------------------------------------------
14
15 with     decls.dgenerals,
16   Ada.Integer_Text_IO,
17   Ada.Text_IO,
18         decls.d_hash;
19
20 use      decls.dgenerals,
21         decls.d_hash;
22
23
24 package decls.d_taula_de_noms is
25
26     --pragma pure;
27
28     -- Excepcions
29     E_Tids_Plena : exception;
30     E_Tcar_Plena : exception;
31
32     type taula_de_noms is limited private;
33
34     procedure tbuida    (tn : out taula_de_noms);
35
36     procedure posa_id   (tn : in out taula_de_noms;
37                          idn : out id_nom;
38                          nom : in string);
39
```

```
40      procedure posa_str  (tn : in out taula_de_noms;
41                           ids : out rang_tcar;
42                             s : in string);
43
44      function cons_nom   (tn : in taula_de_noms;
45                           idn : in id_nom) return string;
46
47      function cons_str   (tn : in taula_de_noms;
48                           ids : in rang_tcar) return string;
49
50
51      private
52
53          type t_identificador is record
54                  pos_tcar : rang_tcar;
55                   seguent : id_nom;
56              long_paraula : Natural;
57          end record;
58
59          type taula_identificadors is array
60            (1 .. id_nom'Last) of t_identificador;
61
62          type taula_caracters is array (rang_tcar)
63            of character;
64
65          type taula_de_noms is record
66              td : taula_dispersio;
67             tid : taula_identificadors;
68              tc : taula_caracters;
69             nid : id_nom;
70            ncar : rang_tcar;
71          end record;
72
73          -- Funcio de comparacio de dues paraules
74          function par_iguals (par1, par2 : in string)
75                               return boolean;
76
77
78  end decls.d_taula_de_noms;
```

2.2 Fitxer *decls-d_taula_de_noms.adb*

```
1  -- ------------------------------------------------
2  --  Paquet de declaracions de la taula de noms
3  -- ------------------------------------------------
4  --  Versio  :   0.2
5  --  Autors  :   Jose Ruiz Bravo
6  --              Biel Moya Alcover
7  --              Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --  Implementacio dels procediments per al
10 -- tractament de la taula de noms:
11 --
12 --          - Buidat de la taula
13 --          - Insercio
14 --          - Insercio d'strings
15 --          - Consulta
16 --
17 -- ------------------------------------------------
18
19
20 package body decls.d_taula_de_noms is
21
22     -- Donam els valors per defecte de cada camp, sempre
23     -- que un camp no sigui utilitzat valdra 0.
24     procedure tbuida (tn : out taula_de_noms) is
25
26     begin
27
28         for i in tn.td'range loop
29             tn.td(i) := id_nul;
30         end loop;
31
32         tn.nid := 1;
33         --tn.ncar := 0;
34         Tn.Ncar := 1;
35
36         tn.tid(1).seguent := id_nul;
37
38     end tbuida;
39
40
41
```

```
42      function par_iguals (par1, par2 : in string) return boolean is
43
44          it_p1 : integer;
45          it_p2 : integer;
46
47      begin
48
49          if par1'Length = par2'Length then
50
51              it_p1 := par1'First;
52              it_p2 := par2'First;
53
54              while it_p1 < par2'Length and par2(it_p2) = par1(it_p1) loop
55                 Ada.Text_IO.Put_Line("par2: "&Par2(It_P2)'Img&" / par1: "&
56                 it_p1 := it_p1 + 1;
57                  it_p2 := it_p2 + 1;
58              end loop;
59
60              if par1(it_p1) = par2(it_p2) then
61                  return true;
62              end if;
63
64          end if;
65
66          return false;
67
68      end par_iguals;
69
70
71
72      procedure posa_id   (tn : in out taula_de_noms;
73                           idn : out id_nom;
74                           nom : in string) is
75
76          -- Variable per el valor de la funcio de dispersio.
77          p_tid : rang_dispersio;
78
79          -- Indexos per recorrer la taula d'identificadors.
80          idx : id_nom;
81          Trobat : boolean;
82
83          -- Index per recorrer la taula de caracters.
84          jdx : rang_tcar;
```

```
85
86          p : taula_identificadors renames tn.tid;
87
88      begin
89
90          p_tid := fdisp_tn(nom);
91          Idx := Tn.Td(P_Tid);
92          Trobat := False;
93
94          while not Trobat and Idx/=Id_Nul loop
95             Ada.Text_IO.Put_Line(Nom);
96             Ada.Text_IO.Put_Line("long paraula: "&Tn.Tid(Idx).Long_paraula'
97             Ada.Text_IO.Put_Line(Cons_Nom(Tn, Idx));
98
99             --if (Par_Iguals(Nom, Cons_Nom(Tn, Idx))) then
100            if (Nom = Cons_Nom(Tn, Idx)) then
101            --Ada.Text_IO.Put_Line("son iguals");
102               Trobat := True;
103            else
104               Idx := Tn.Tid(Idx).Seguent;
105            end if;
106         end loop;
107
108         if not Trobat then
109            Idx := Tn.Nid;
110
111            Ada.Text_IO.Put(".."&Idx'IMG);
112            Ada.Text_IO.Put("    tn.nid: "&Tn.Nid'Img);
113
114            Tn.Tid(Tn.nid).Pos_Tcar := Tn.Ncar;
115            Tn.Tid(Tn.nid).Seguent := Tn.Td(P_Tid);
116            Tn.Tid(Tn.nid).Long_Paraula := Nom'Length;
117
118            Tn.Td(P_Tid) := Tn.Nid;
119
120            Tn.Nid := Tn.Nid + 1;
121            for I in 1 .. Nom'Length loop
122               Tn.Tc(Tn.ncar) := Nom(I);
123               Tn.Ncar := Tn.Ncar + 1;
124            end loop;
125
126            Ada.Text_IO.Put_Line("");
127            Jdx := 0;
```

```
128            while Jdx < 80 loop
129               Ada.Text_IO.Put(""&Tn.Tc(jdx)'Img);
130               Jdx := Jdx + 1;
131            end loop;
132
133            Tn.Tc(Tn.Ncar) := '$';
134            Tn.Ncar := Tn.Ncar + 1;
135         else
136            Ada.Text_IO.Put_Line("Trobat es true");
137         end if;
138
139      end posa_id;
140
141
142
143      procedure posa_str  (tn : in out taula_de_noms;
144                           ids : out rang_tcar;
145                             s : in string) is
146
147         -- Index per recorrer la taula de caracters.
148         jdx : rang_tcar;
149
150      begin
151
152         -- Excepcio per a controlar tc plena
153         if (tn.ncar + s'Length) > rang_tcar'Last then
154             raise E_Tcar_Plena;
155         end if;
156
157         -- Omplim la taula de caracters, desde la primera
158         -- posicio lliure 'ncar'.
159         jdx := tn.ncar;
160         ids := tn.ncar;
161
162         for i in 1..s'Length loop
163
164             tn.tc(jdx) := s(i);
165             jdx := jdx + 1;
166
167         end loop;
168
169         tn.ncar := jdx + 1;
170         tn.tc(jdx) := '$';
```

```
171
172
173     end posa_str;
174
175
176
177     function cons_nom (tn : in taula_de_noms; idn : in id_nom) return str
178
179        --S : String(1..Tn.Tid(Idn).Long_Paraula);
180
181        It1, It2 : Rang_Tcar;
182
183     begin
184        --Ada.Text_IO.Put_Line("Idn: "&Idn'Img);
185        --Ada.Text_IO.Put_Line("pos tcar: "&Tn.Tid(Idn).Pos_Tcar'Img);
186        --Ada.Text_IO.Put_Line("long para (li sumam postcar i restam 1): "
187        --Ada.Text_IO.Put_Line(String(Tn.Tc(Tn.Tid(Idn).Pos_Tcar .. (Tn.Ti
188        --for I in 0 .. 13 loop
189
190        It1 := Tn.Tid(Idn).Pos_Tcar;
191        It2 := Rang_Tcar(Tn.Tid(Idn).Long_Paraula);
192        It2 := It2 + It1 - 1;
193        --It2 := It2 + It1;
194
195        return String(
196                      Tn.Tc(it1 .. it2)
197                        );
198
199
200        --for I in It1 .. It2 loop
201        --   S := S&Character(Tn.Tc(Rang_Tcar(I)));
202        --Ada.Text_Io.Put(Tn.Tc(Rang_Tcar(I))'Img);
203        --end loop;
204
205
206        --return string(tn.tc(tn.tid(idn).pos_tcar .. tn.tid(idn).pos_tcar
207
208     end cons_nom;
209
210
211
212     function cons_str   (tn : in taula_de_noms; ids : in rang_tcar) retur
213
```

```
214         idx : rang_tcar;
215
216     begin
217
218         idx := ids;
219
220         while (tn.tc(idx) /= '$') loop
221             idx := idx+1;
222         end loop;
223
224         return string(tn.tc(ids..idx-1));
225
226     end cons_str;
227
228
229 end decls.d_taula_de_noms;
```

3 Tokens i atributs

3.1 Fitxer *d_token.ads*

```
1  -- ------------------------------------------------
2  --  Paquet de declaracions dels tokens
3  -- ------------------------------------------------
4  --  Versio      :       0.2
5  --  Autors      :       Jose Ruiz Bravo
6  --                      Biel Moya Alcover
7  --                      Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --      Definicio del tipus token.
10 --
11 -- ------------------------------------------------
12
13 package d_token is
14
15         type token is  (pc_procedure,
16                         pc_begin,
17                         pc_while,
18                         pc_if,
19                         pc_else,
20                         pc_end,
21                         pc_do,
22                         pc_constant,
23                         pc_type,
24                         pc_array,
25                         pc_record,
26                         pc_is,
27                         pc_then,
28                         pc_not,
29                         pc_in,
30                         pc_out,
31                         pc_new,
32                         pc_null,
33                         pc_of,
34                         pc_mod,
35                         pc_range,
36                         pc_and,
37                         pc_or,
38                         s_assignacio,
39                         s_dospunts,
```

```
40                              s_final,
41                              s_coma,
42                              s_parentesiobert,
43                              s_parentesitancat,
44                              s_puntsrang,
45                              s_puntrec,
46                              op_menor,
47                              op_menorigual,
48                              op_majorigual,
49                              op_major,
50                              op_igual,
51                              op_distint,
52                              op_suma,
53                              op_resta,
54                              op_multiplicacio,
55                              op_divisio,
56                              id,
57                              cadena,
58                              const,
59                              Error,
60                              End_of_Input);
61
62
63 end d_token;
```

3.2 Fitxer *d_atribut.ads*

```
1  -- ------------------------------------------------
2  --  Paquet de procediments dels atributs
3  -- ------------------------------------------------
4  --  Versio  :   0.1
5  --  Autors  :   Jose Ruiz Bravo
6  --              Biel Moya Alcover
7  --              Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --      En aquest fitxer tenim implementats les
10 --  assignacions de cada tipus de token al tipus
11 --  atribut que li correspon. Cal destacar
12 --  l'utilitzacio de la taula de noms en els
13 --  casos d'identificadors i strings.
14 --
15 -- ------------------------------------------------
16
17 with     decls.dgenerals,
18         decls.d_taula_de_noms;
19
20 use      decls.dgenerals,
21         decls.d_taula_de_noms;
22
23
24 package d_atribut is
25
26
27     type tipus_atribut is (atom,
28                            a_ident,
29                            a_lit_num,
30                            a_lit_car,
31                            a_lit_string);
32
33
34     type atribut (t : tipus_atribut := atom) is record
35
36         lin, col : natural;
37
38         case t is
39
40             when atom           => null;
41
```

```
42              when a_ident         => idn : id_nom;
43
44              when a_lit_num       => int : integer;
45
46              when a_lit_car       => val : character;
47
48              when a_lit_string    => ids : rang_tcar;
49
50          end case;
51
52      end record;
53
54
55      procedure mt_atom       (l, c : in natural;
56                                   a : out atribut);
57
58      procedure mt_identificador  (l, c : in natural;
59                                       s : in string;
60                                       a : out atribut);
61
62      procedure mt_string (l, c : in natural;
63                               s : in string;
64                               a : out atribut);
65
66      procedure mt_caracter   (l, c : in natural;
67                                 car : in string;
68                                   a : out atribut);
69
70      procedure mt_numero (l, c : in natural;
71                                   i : in string;
72                                   a : out atribut);
73
74
75  end d_atribut;
```

3.3 Fitxer *d_atribut.adb*

```
1  -- ------------------------------------------------
2  --  Paquet de procediments dels atributs
3  -- ------------------------------------------------
4  --  Versio  :   0.1
5  --  Autors  :   Jose Ruiz Bravo
6  --              Biel Moya Alcover
7  --              Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --      En aquest fitxer tenim implementats les
10 --  assignacions de cada tipus de token al tipus
11 --  atribut que li correspon. Cal destacar
12 --  l'utilitzacio de la taula de noms en els
13 --  casos d'identificadors i strings.
14 --
15 -- ------------------------------------------------
16
17 with     U_Lexica;
18         --decls.tn;
19
20 use      U_Lexica;
21         --decls.tn;
22
23
24 package body d_atribut is
25
26
27     procedure mt_atom (l, c : in natural;
28                        a : out atribut) is
29     begin
30         a := (atom, l, c);
31     end mt_atom;
32
33
34     procedure mt_identificador (l, c : in natural;
35                                 s : in string;
36                                 a : out atribut) is
37         id : id_nom;
38     begin
39         id := id_nul;
40         posa_id(tn, id, s);
41         a := (a_ident, l, c, id);
```

```
42      end mt_identificador;
43
44
45      procedure mt_string (l, c : in natural;
46                           s : in string;
47                           a : out atribut) is
48          id : rang_tcar;
49      begin
50          posa_str(tn, id, s);
51          a := (a_lit_string, l, c, id);
52      end mt_string;
53
54
55      procedure mt_caracter (l, c : in natural;
56                             car : in string;
57                               a : out atribut) is
58      begin
59          a := (a_lit_car, l, c, car(car'First+1));
60      end mt_caracter;
61
62
63      procedure mt_numero (l, c : in natural;
64                           i : in string;
65                           a : out atribut) is
66      begin
67          a := (a_lit_num, l, c, Integer'value(i));
68      end mt_numero;
69
70
71  end d_atribut;
```

Índex