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,
```

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

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

```
107 ".."            {mt_atom(tok_begin_line, tok_begin_col,
108                         yylval); return s_puntsrang;}
109
110 "."                     {mt_atom(tok_begin_line, tok_begin_col,
111                         yylval); return s_puntrec;}
112
113
114 --Operadors
115
116 "<"                     {mt_atom(tok_begin_line, tok_begin_col,
117                         yylval); return op_menor;}
118
119 "<="            {mt_atom(tok_begin_line, tok_begin_col,
120                         yylval); return op_menorigual;}
121
122 ">="            {mt_atom(tok_begin_line, tok_begin_col,
123                         yylval); return op_majorigual;}
124
125 ">"                     {mt_atom(tok_begin_line, tok_begin_col,
126                         yylval); return op_major;}
127
128 "="                     {mt_atom(tok_begin_line, tok_begin_col,
129                         yylval); return op_igual;}
130
131 "/="            {mt_atom(tok_begin_line, tok_begin_col,
132                         yylval); return op_distint;}
133
134 "+"                     {mt_atom(tok_begin_line, tok_begin_col,
135                         yylval); return op_suma;}
136
137 "-"                     {mt_atom(tok_begin_line, tok_begin_col,
138                         yylval); return op_resta;}
139
140 "*"                     {mt_atom(tok_begin_line, tok_begin_col,
141                         yylval); return op_multiplicacio;}
142
143 "/"                     {mt_atom(tok_begin_line, tok_begin_col,
144                         yylval); 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                 d_token,
200                 decls.d_atribut;
201
202
203 use              decls.d_taula_de_noms,
204                 d_token,
205                 decls.d_atribut;
206
207
208 package u_lexica is
209
210         yylval: atribut;
211         tn : taula_de_noms;
212         function YYLex return token;
213         tok_begin_line : integer;
214         tok_begin_col : integer;
215
216 end u_lexica;
217
218
219
220 package body u_lexica is
221
222 ##
223
224 begin
225
226         tbuida(tn);
227
228 end u_lexica;
```

1.2 Taula de noms

1.2.1 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         decls.d_hash;
17
18 use      decls.dgenerals,
19         decls.d_hash;
20
21
22 package decls.d_taula_de_noms is
23
24     --pragma pure;
25
26     -- Excepcions
27     E_Tids_Plena : exception;
28     E_Tcar_Plena : exception;
29
30     type taula_de_noms is limited private;
31
32     procedure tbuida    (tn : out taula_de_noms);
33
34     procedure posa_id   (tn : in out taula_de_noms;
35                          idn : out id_nom;
36                          nom : in string);
37
38     procedure posa_tc   (tn : in out taula_de_noms;
39                          nom : in string);
40
```

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

1.2.2 Fitxer *decls-d_taula_de_noms.adb*

```
1  -- ------------------------------------------------
2  --  Paquet de declaracions de la taula de noms
3  -- ------------------------------------------------
4  --  Versio  :   0.3
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.
23     procedure tbuida (tn : out taula_de_noms) is
24
25     begin
26
27         for i in tn.td'range loop
28             tn.td(i) := id_nul;
29         end loop;
30
31         tn.nid := 1;
32         Tn.Ncar := 1;
33
34         tn.tid(1).seguent := id_nul;
35
36     end tbuida;
37
38
39
40     procedure posa_id   (tn : in out taula_de_noms;
41                          idn : out id_nom;
```

```
42                          nom : in string) is
43
44          -- Variable per el valor de la funcio de dispersio.
45          p_tid : rang_dispersio;
46
47          -- Index per recorrer la taula d'identificadors.
48          idx : id_nom;
49          Trobat : boolean;
50
51          p : taula_identificadors renames tn.tid;
52
53      begin
54
55          p_tid := fdisp_tn(nom);
56          Idx := Tn.Td(P_Tid);
57          Trobat := False;
58
59          while not Trobat and Idx/=Id_Nul loop
60              if (Nom = Cons_Nom(Tn, Idx)) then
61                  Trobat := True;
62              else
63                  Idx := p(Idx).Seguent;
64              end if;
65          end loop;
66
67          if not Trobat then
68              Idn := Tn.Nid;
69              p(idn).Pos_Tcar := Tn.Ncar;
70              p(idn).Seguent := Tn.Td(P_Tid);
71              p(idn).Long_Paraula := Nom'Length;
72
73              Tn.Td(P_Tid) := Tn.Nid;
74
75              posa_tc(tn, nom);
76
77              Tn.Tc(Tn.Ncar) := '$';
78              Tn.Ncar := Tn.Ncar + 1;
79          end if;
80
81      end posa_id;
82
83
84
```

```
85      procedure posa_tc   (tn : in out taula_de_noms;
86                          nom : in string) is
87
88      begin
89
90         tn.nid := tn.nid + 1;
91
92         for i in 1 .. nom'Length loop
93            tn.tc(tn.ncar) := nom(i);
94            tn.ncar := tn.ncar + 1;
95         end loop;
96
97      end posa_tc;
98
99
100
101     procedure posa_str  (tn : in out taula_de_noms;
102                          ids : out rang_tcar;
103                            s : in string) is
104
105         -- Index per recorrer la taula de caracters.
106         jdx : rang_tcar;
107
108     begin
109
110         -- Excepcio per a controlar tc plena
111         if (tn.ncar + s'Length) > rang_tcar'Last then
112             raise E_Tcar_Plena;
113         end if;
114
115         -- Omplim la taula de caracters, desde la primera
116         -- posicio lliure 'ncar'.
117         jdx := tn.ncar;
118         ids := tn.ncar;
119
120         for i in 1..s'Length loop
121             tn.tc(jdx) := s(i);
122             jdx := jdx + 1;
123         end loop;
124
125         tn.ncar := jdx + 1;
126         tn.tc(jdx) := '$';
127
```

```
128
129     end posa_str;
130
131
132
133     function cons_nom  (tn : in taula_de_noms;
134                         idn : in id_nom)
135                         return string is
136
137        It1, It2 : Rang_Tcar;
138
139     begin
140
141        It1 := Tn.Tid(Idn).Pos_Tcar;
142        It2 := Rang_Tcar(Tn.Tid(Idn).Long_Paraula);
143        It2 := It2 + It1 - 1;
144
145        return String(Tn.Tc(it1 .. it2));
146
147
148     end cons_nom;
149
150
151
152     function cons_str   (tn : in taula_de_noms;
153                          ids : in rang_tcar)
154                          return string is
155
156         idx : rang_tcar;
157
158     begin
159
160         idx := ids;
161         while (tn.tc(idx) /= '$') loop
162             idx := idx+1;
163         end loop;
164
165         return string(tn.tc(ids..idx-1));
166
167     end cons_str;
168
169
170 end decls.d_taula_de_noms;
```

1.3 Tokens i atributs

1.3.1 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;
```

1.3.2 Fitxer *decls-d_ atribut.ads*

```
1  -- ------------------------------------------------
2  --  Paquet de procediments dels atributs
3  -- ------------------------------------------------
4  --  Versio  :   0.2
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 decls.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        => car : character;
47
48              when a_lit_string     => ids : rang_tcar;
49
50          end case;
51
52      end record;
53
54
55      procedure mt_atom
56        (l, c : in natural;
57             a : out atribut);
58
59      procedure mt_identificador
60        (l, c : in natural;
61             s : in string;
62             a : out atribut);
63
64      procedure mt_string
65        (l, c : in natural;
66             s : in string;
67             a : out atribut);
68
69      procedure mt_caracter
70        (l, c : in natural;
71          car : in string;
72             a : out atribut);
73
74      procedure mt_numero
75        (l, c : in natural;
76             s : in string;
77             a : out atribut);
78
79
80  end decls.d_atribut;
```

2 Anàlisi Sintàctica

2.1 Gramàtica del nostre llenguatge

3 Declaracions i altres paquets

3.1 Fitxer *decls.ads*

```
1  -- ------------------------------------------------
2  --  Paquet de Declaracions
3  -- ------------------------------------------------
4  --  Versio  :   0.1
5  --  Autors  :   Jose Ruiz Bravo
6  --              Biel Moya Alcover
7  --              Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --  Paquet de declaracions pare.
10 --
11 -- ------------------------------------------------
12
13 package decls is
14
15    pragma pure;
16
17
18 end decls;
```

3.2 Fitxer *decls-dgenerals.ads*

```
1  -- ------------------------------------------------
2  --  Paquet de declaracions generals
3  -- ------------------------------------------------
4  --  Versio  :   0.2
5  --  Autors  :   Jose Ruiz Bravo
6  --              Biel Moya Alcover
7  --              Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --  Declaracions generals.
10 --
11 -- ------------------------------------------------
12
13 package decls.dgenerals is
14
15    pragma pure;
16
17    -- TAULA DE NOMS
18    max_id : constant integer := 1000;
19    type id_nom is new integer
20      range 0 .. max_id;
21
22    -- Valor nul per al tipus id_nom
23    id_nul : constant id_nom := 0;
24
25    long_num_ident : constant integer := 40;
26    type rang_tcar is new integer
27      range 0 .. (long_num_ident*max_id);
28
29    -- Taula de dispersio:
30    tam_dispersio : constant integer := 101;
31    type rang_dispersio is new integer
32      range -1 .. tam_dispersio;
33
34    -- Valor nul per el rang dispersio
35    dispersio_nul : constant rang_dispersio := -1;
36
37    type taula_dispersio is array (rang_dispersio)
38      of id_nom;
39
40
41 end decls.dgenerals;
```

3.3 Fitxer *decls-d_hash.ads*

```
1  -- ------------------------------------------------
2  --  Paquet de declaracions de les funcions hash
3  -- ------------------------------------------------
4  --  Versio  :   0.3
5  --  Autors  :   Jose Ruiz Bravo
6  --              Biel Moya Alcover
7  --              Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --  Especificacio de les funcions de hash:
10 --      - Hash de quadratic per accedir a la
11 --      taula de noms.
12 --
13 -- ------------------------------------------------
14
15 with     decls.dgenerals;
16
17 use      decls.dgenerals;
18
19
20 package decls.d_hash is
21
22     pragma pure;
23
24     function fdisp_tn (nom : in string)
25                        return rang_dispersio;
26
27
28 end decls.d_hash;
```

3.4 Fitxer *decls-d_hash.adb*

```
1  -- ------------------------------------------------
2  --  Paquet de procediments de les funcions hash
3  -- ------------------------------------------------
4  --  Versio  :   0.2
5  --  Autors  :   Jose Ruiz Bravo
6  --              Biel Moya Alcover
7  --              Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --  Procediments de les funcions de hash:
10 --      - Hashing quadratic
11 --
12 -- ------------------------------------------------
13
14 package body decls.d_hash is
15
16    function fdisp_tn (nom : in string)
17                       return rang_dispersio is
18
19         a : array (nom'Range) of integer;
20         r : array (1..2*nom'Last) of integer;
21
22         k, c, m, n : integer;
23
24         base : constant Integer :=
25           Character'Pos(Character'Last)+1;
26
27     begin
28
29         n := nom'Last;
30         m := nom'Length;
31
32         for i in 1..n loop
33             a(i) := character'Pos(nom(i));
34         end loop;
35
36         for i in 1..2*n loop
37             r(i) := 0;
38         end loop;
39
40         for i in 1..n loop
41             c := 0; k := i - 1;
```

```
42              for j in 1..n loop
43                  c := c + r(k+j) + a(i) + a(j);
44                  r(k+j) := c mod base;
45                  c := c/base;
46              end loop;
47              r(k+n+1) := r(k+n+1) + c;
48          end loop;
49
50          c := (r(n+1) * base + r(n)) mod (tam_dispersio);
51
52          return rang_dispersio(c);
53
54      end fdisp_tn;
55
56
57  end decls.d_hash;
```

4 Proves i programa principal

4.1 Fitxer *compilemon.adb*, programa principal

```
1  -- ------------------------------------------------
2  --  Programa de prova
3  -- ------------------------------------------------
4  --  Versio  :   0.1
5  --  Autors  :   Jose Ruiz Bravo
6  --              Biel Moya Alcover
7  --              Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --      Programa per comprovar les funcionalitats
10 --  del lexic i la taula de noms.
11 --
12 -- ------------------------------------------------
13
14 with     Ada.Text_IO,
15         Ada.Command_Line,
16         decls.d_taula_de_noms,
17         decls.tn,
18         decls.dgenerals,
19         d_token,
20         compilemon_io,
21         u_lexica;
22
23 use      Ada.Text_IO,
24         Ada.Command_Line,
25         decls.d_taula_de_noms,
26         decls.tn,
27         decls.dgenerals,
28         d_token,
29         compilemon_io,
30         u_lexica;
31
32
33 procedure compilemon is
34     Tk:Token;
35 begin
36
37     --tbuida(tn);
38
39     Open_Input(Argument(1));
```

```
40      tk := Yylex;
41
42    while tk /= end_of_input loop
43        Put_Line(Token'Image(Tk));
44        tk := Yylex;
45    end loop;
46
47     close_Input;
48
49  --    exception
50    --      when E_Tids_Plena =>
51      --        Put_Line("ERROR: La taula d'identificadors es plena.");
52
53  --        when E_Tcar_Plena =>
54    --          Put_Line("ERROR: La taula de caracters es plena.");
55
56  --        when Syntax_Error =>
57    --          Put_Line("ERROR: Error a la linea "&yy_line_number'img&" i colu
58
59  end compilemon;
```

4.2 Proves

4.2.1 Prova 1: fitxer *prova01.lem*

```
procedimiento
begin
Begin
end
estocastico
proves
Es
ES
procedimiento
prova
prova
si
sino
```

Índex

1 Anàlisi Lèxica **1**
1.1 Descripció del lèxic: *compilemon.l* 1
1.2 Taula de noms 7
1.2.1 Fitxer *decls-d_taula_de_noms.ads* 7
1.2.2 Fitxer *decls-d_taula_de_noms.adb* 9
1.3 Tokens i atributs 13
1.3.1 Fitxer *d_token.ads* 13
1.3.2 Fitxer *decls-d_atribut.ads* 15

2 Anàlisi Sintàctica **17**
2.1 Gramàtica del nostre llenguatge 17

3 Declaracions i altres paquets **18**
3.1 Fitxer *decls.ads* 18
3.2 Fitxer *decls-dgenerals.ads* 19
3.3 Fitxer *decls-d_hash.ads* 20
3.4 Fitxer *decls-d_hash.adb* 21

4 Proves i programa principal **23**
4.1 Fitxer *compilemon.adb*, programa principal 23
4.2 Proves 25
4.2.1 Prova 1: fitxer *prova01.lem* 25