

Institut Supérieur d'Électronique de Paris  
**Projet de Fin d'Études**

Reponsable: M. Hugueney

# Finite State Transducers Just-In-Time Compiling

Do you hear the bytecode ?

Émilien Boulben  
Victor Delepine  
Corentin Peuvrel

17 juin 2015  
Paris

# Table des matières

<b>Introduction</b>	<b>1</b>
<b>1 Analyse préalable</b>	<b>2</b>
<b>2 Des FST en JIT</b>	<b>3</b>
<b>3 Optimiser : comment ?</b>	<b>4</b>
<b>Conclusion</b>	<b>5</b>
<b>A Annexe : tests avec un script shell</b>	<b>6</b>
A.1 Dictionnaires . . . . .	6
A.2 FST . . . . .	7
A.3 Générer à la volée du code C . . . . .	8
A.3.1 Script shell . . . . .	8
A.3.2 Code C généré pour la FST définie dans le Tableau 3 . . . . .	10
A.3.3 Code C généré pour la FST définie dans le Tableau 4 . . . . .	13

## Listings

1	Script pour générer un code C à la volée d'une FST . . . . .	8
2	Code C généré pour la FST définie dans Tableau 3 . . . . .	10
3	Code C généré pour la FST définie dans Tableau 4 . . . . .	13

## Liste des tableaux

1	Dictionnaire à utiliser avec la FST dans le Tableau 3 . . . . .	6
2	Dictionnaire à utiliser avec la FST dans le Tableau 4 . . . . .	6
3	FST utilisée avec le dictionnaire Tableau 1, voir Figure 1 page 7 . . . . .	7
4	FST utilisée avec le dictionnaire Tableau 2, voir Figure 2 page 7 . . . . .	7

## Table des figures

1	La FST associée avec le Tableau 3 page 7 . . . . .	7
2	La FST associée avec le Tableau 4 page 7 . . . . .	7

# Introduction

# 1 Analyse préalable

## 2 Des FST en JIT



### **3 Optimiser : comment ?**

## Conclusion

## A Annexe : tests avec un script shell

### A.1 Dictionnaires

Value	Word
0	mop
1	moth
2	pop
3	star
4	stop
5	top

TABLEAU 1 – Dictionnaire à utiliser avec la FST dans le Tableau 3

Value	Word
0	mop
1	moth
2	pop
3	slop
4	sloth
5	stop
6	top

TABLEAU 2 – Dictionnaire à utiliser avec la FST dans le Tableau 4

## A.2 FST

Nœu	0	0	0	0	1	2	3	2	4	6	7	5	7	8	
Nœu suivant	1	4	4	6	2	3	9	9	5	7	5	9	8	9	
Nœu final															9
Caractère	M	P	T	S	O	T	H	P	O	T	O	P	A	R	
Poids		2	5	3			1				1				

TABLEAU 3 – FST utilisée avec le dictionnaire Tableau 1, voir Figure 1 page 7

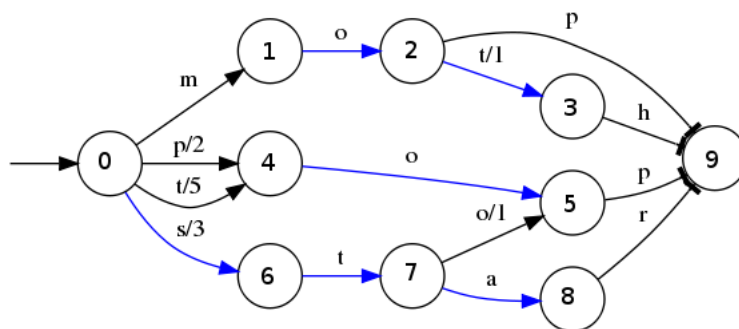


FIGURE 1 – La FST associée avec le Tableau 3 page 7

Nœu	0	0	0	0	3	3	1	2	4	5	6	5	
Nœu suivant	1	1	3	4	1	4	2	7	5	6	7	7	
Nœu final													7
Caractère	P	T	S	M	T	L	O	P	O	T	H	P	
Poids	2	6	3		2					1			

TABLEAU 4 – FST utilisée avec le dictionnaire Tableau 2, voir Figure 2 page 7

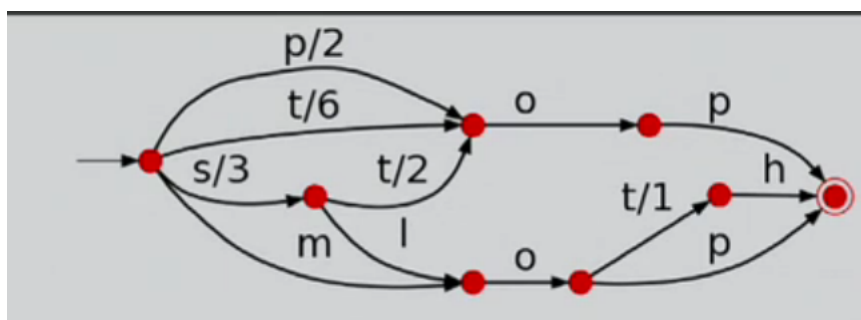


FIGURE 2 – La FST associée avec le Tableau 4 page 7

## A.3 Générer à la volée du code C

### A.3.1 Script shell

```

1  #!/bin/bash
2
3  FST="$1"
4
5  sort "$FST" > "$FST.sort"
6
7  FIRST_CALL=1
8
9  cat <<EOF
10 #include <stdio.h>
11
12 int compute_fst(const char* token)
13 {
14     int pos=0;
15     int total=0;
16
17 EOF
18
19 while read DEP ARR CHAR WEIGHT ; do
20     # If it's a final node
21     if [[ ! "$CHAR" ]]; then
22         WEIGHT=${ARR:-0}
23         cat <<EOF
24         default:
25             return -1;
26     }
27
28 NODE_$DEP :
29 EOF
30     (( WEIGHT != 0 )) &&
31     echo "        total += $WEIGHT;"
32     echo "        goto END;"
33
34     continue
35 fi
36
37 : ${WEIGHT:=0}
38
39 if [[ $DEP != $PREV_DEP ]]; then
40     if [[ ! "$FIRST_CALL" ]]; then
41         cat <<EOF
42         default:
43             return -1;
44     }
45
46 EOF
47     fi
48
49     cat <<EOF
50 NODE_$DEP :
51     pos++;
52     switch (token[pos-1]) {
53 EOF
54     fi
55
56     echo "        case '$CHAR':"
```

```
57      (( WEIGHT != 0 )) &&
58          echo "          total += $WEIGHT;"
59      echo "          goto NODE_$ARR;"
60
61      PREV_DEP=$DEP
62      FIRST_CALL=
63  done < "$FST.sort"
64
65  cat <<EOF
66
67  END :
68      return total;
69  }
70
71  int main(int argc, const char *argv[])
72  {
73      if (argc < 2)
74          return 1;
75
76      printf("%d\n", compute_fst(argv[1]));
77      return 0;
78  }
79  EOF
80
81  rm "$FST.sort"
```

Listing 1 – Script pour générer un code C à la volée d'une FST

### A.3.2 Code C généré pour la FST définie dans le Tableau 3

```

1  #include <stdio.h>
2
3  int compute_fst(const char* token)
4  {
5      int pos=0;
6      int total=0;
7
8  NODE_0 :
9      pos++;
10     switch (token[pos-1]) {
11     case 'M':
12         goto NODE_1;
13     case 'P':
14         total += 2;
15         goto NODE_4;
16     case 'T':
17         total += 5;
18         goto NODE_4;
19     case 'S':
20         total += 3;
21         goto NODE_6;
22     default:
23         return -1;
24     }
25
26  NODE_1 :
27      pos++;
28      switch (token[pos-1]) {
29      case 'O':
30          goto NODE_2;
31      default:
32          return -1;
33      }
34
35  NODE_2 :
36      pos++;
37      switch (token[pos-1]) {
38      case 'T':
39          total += 1;
40          goto NODE_3;
41      case 'P':
42          goto NODE_9;
43      default:
44          return -1;
45      }
46
47  NODE_3 :
48      pos++;
49      switch (token[pos-1]) {
50      case 'H':
51          goto NODE_9;
52      default:
53          return -1;
54      }
55
56  NODE_4 :
57      pos++;
58      switch (token[pos-1]) {

```

```
59     case 'O':
60         goto NODE_5;
61     default:
62         return -1;
63 }
64
65 NODE_5 :
66     pos++;
67     switch (token[pos-1]) {
68     case 'P':
69         goto NODE_9;
70     default:
71         return -1;
72     }
73
74 NODE_6 :
75     pos++;
76     switch (token[pos-1]) {
77     case 'T':
78         goto NODE_7;
79     default:
80         return -1;
81     }
82
83 NODE_7 :
84     pos++;
85     switch (token[pos-1]) {
86     case 'O':
87         total += 1;
88         goto NODE_5;
89     case 'A':
90         goto NODE_8;
91     default:
92         return -1;
93     }
94
95 NODE_8 :
96     pos++;
97     switch (token[pos-1]) {
98     case 'R':
99         goto NODE_9;
100    default:
101        return -1;
102    }
103
104 NODE_9 :
105     goto END;
106
107 END :
108     return total;
109 }
110
111 int main(int argc, const char *argv[])
112 {
113     if (argc < 2)
114         return 1;
115
116     printf("%d\n", compute_fst(argv[1]));
117     return 0;
```



118 || }

Listing 2 – Code C généré pour la FST définie dans Tableau 3

### A.3.3 Code C généré pour la FST définie dans le Tableau 4

```

1  #include <stdio.h>
2
3  int compute_fst(const char* token)
4  {
5      int pos=0;
6      int total=0;
7
8  NODE_0 :
9      pos++;
10     switch (token[pos-1]) {
11     case 'P':
12         total += 2;
13         goto NODE_1;
14     case 'T':
15         total += 6;
16         goto NODE_1;
17     case 'S':
18         total += 3;
19         goto NODE_3;
20     case 'M':
21         goto NODE_4;
22     default:
23         return -1;
24     }
25
26  NODE_1 :
27     pos++;
28     switch (token[pos-1]) {
29     case 'O':
30         goto NODE_2;
31     default:
32         return -1;
33     }
34
35  NODE_2 :
36     pos++;
37     switch (token[pos-1]) {
38     case 'P':
39         goto NODE_7;
40     default:
41         return -1;
42     }
43
44  NODE_3 :
45     pos++;
46     switch (token[pos-1]) {
47     case 'T':
48         total += 2;
49         goto NODE_1;
50     case 'L':
51         goto NODE_4;
52     default:
53         return -1;
54     }
55
56  NODE_4 :
57     pos++;
58     switch (token[pos-1]) {

```

```
59     case '0':
60         goto NODE_5;
61     default:
62         return -1;
63     }
64
65 NODE_5 :
66     pos++;
67     switch (token[pos-1]) {
68     case 'T':
69         total += 1;
70         goto NODE_6;
71     case 'P':
72         goto NODE_7;
73     default:
74         return -1;
75     }
76
77 NODE_6 :
78     pos++;
79     switch (token[pos-1]) {
80     case 'H':
81         goto NODE_7;
82     default:
83         return -1;
84     }
85
86 NODE_7 :
87     goto END;
88
89 END :
90     return total;
91 }
92
93 int main(int argc, const char *argv[])
94 {
95     if (argc < 2)
96         return 1;
97
98     printf("%d\n", compute_fst(argv[1]));
99     return 0;
100 }
```

Listing 3 – Code C généré pour la FST définie dans Tableau 4