Digital Logic Design Project 4 – State Minimization

Due: 23:55, Dec. 23, 2021

In sequential circuit, the time sequence of inputs, states, and outputs can be graphically represented in a state diagram, a.k.a. state transition graph (STG). In a STG, states are represented by circles/vertices and the transitions between two states are indicated by directed lines/edges. According to the input condition, each directed line/edge is originated at a "present state" and terminated at a "next state". Since the number of states is correlated with the number of required state flip-flops in a sequential circuit, minimizing the number of states could reduce the number of state flip-flops as well as hardware cost. In this project, your program reads a completely specified STG (i.e., STG without don't cares) in KISS format and outputs its minimum STG in KISS format and in pictorial DOT format.

Please submit your report and program according to the following rules:

1- The font size of your report is 12 in PDF format.
2- The filename of your report is your student ID (e.g., B12345678.pdf).
3- Generate your own KISS file (4 to 6 states, 1 to 2 inputs, and 1 output).
4- Post the content of input and output KISS files.
5- Post the screenshots of your STG (before and after state minimization).
6- Upload a tarball (e.g., B12345678.tgz) of your program source code, your KISS files, and a text ReadMe file, which illustrates how your program to be compiled and executed on a CentOS Linux Workstation.

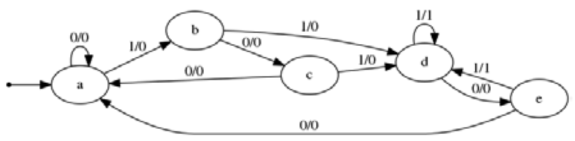
```
KISS Example: input.kiss
.start_kiss
.i 1
.o 1
.p 14
.s 7
.r a
0 a a 0
1 a b 0
0 b c 0
1 b d 0
0 c a 0
1 c d 0
0 d e 0
1 d f 1
0 e a 0
1 e f 1
0 f g 0
1 f f 1
0 g a 0
1 g f 1
.end_kiss

SYNOPSIS
%> PROGRAM KISS_IN KISS_OUT DOT_FILE

Run-time Example:
%> smin input.kiss output.kiss output.dot
```

```
%> cat output.kiss
.start_kiss
.i 1
.o 1
.p 10
.s 5
.r a
0 a a 0
1 a b 0
0 b c 0
1 b d 0
0 c a 0
1 c d 0
0 d e 0
1 d d 1
0 e a 0
1 e d 1
.end_kiss

%> dot -T png output.dot > output.png
```



Basic DOT (graph description language) Reference

Install Graphviz if you want to visualize your DOT file.

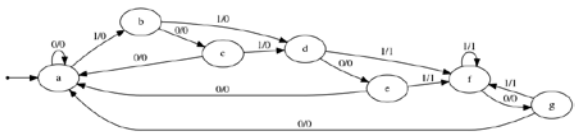
```
%> sudo yum install graphviz
```

```
%> cat input.dot
digraph STG {
   rankdir=LR;

   INIT [shape=point];
   a [label="a"];
   b [label="b"];
   c [label="c"];
   d [label="d"];
   e [label="e"];
   f [label="f"];
   g [label="g"];

   INIT -> a;
   a -> a [label="0/0"];
   a -> b [label="1/0"];
   b -> c [label="0/0"];
   b -> d [label="1/0"];
   c -> a [label="0/0"];
   c -> d [label="1/0"];
   d -> e [label="0/0"];
   d -> f [label="1/1"];
   e -> a [label="0/0"];
   e -> f [label="1/1"];
   f -> g [label="0/0"];
   f -> f [label="1/1"];
   g -> a [label="0/0"];
   g -> f [label="1/1"];
}
```

```
%> dot -T png input.dot > input.png
```

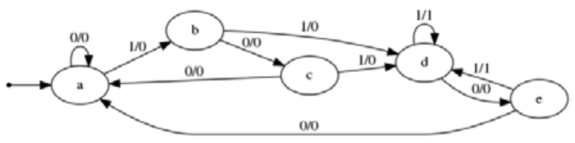


```
%> cat output.dot
digraph STG {
   rankdir=LR;

   INIT [shape=point];
   a [label="a"];
   b [label="b"];
   c [label="c"];
   d [label="d"];
   e [label="e"];

   INIT -> a;
   a -> a [label="0/0"];
   a -> b [label="1/0"];
   b -> c [label="0/0"];
   b -> d [label="1/0"];
   c -> a [label="0/0"];
   c -> d [label="1/0"];
   d -> e [label="0/0"];
   d -> d [label="1/1"];
   e -> a [label="0/0"];
   e -> d [label="1/1"];
}
```

```
%> dot -T png output.dot > output.png
```



Two more examples for your reference (case4):

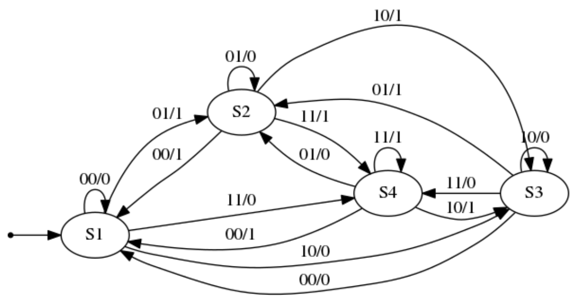
```
%> cat case4.kiss
.start_kiss
.i 2
.o 1
.p 16
.s 4
.r S1
00 S1 S1 0
01 S1 S2 1
10 S1 S3 0
11 S1 S4 0
00 S2 S1 1
01 S2 S2 0
10 S2 S3 1
11 S2 S4 1
00 S3 S1 0
01 S3 S2 1
10 S3 S3 0
11 S3 S4 0
00 S4 S1 1
01 S4 S2 0
10 S4 S3 1
11 S4 S4 1
.end_kiss

%> cat case4.dot
digraph STG {
   rankdir=LR;

   INIT [shape=point];
   S1 [label="S1"];
   S2 [label="S2"];
   S3 [label="S3"];
   S4 [label="S4"];

   INIT -> S1;
   S1 -> S1 [label="00/0"];
   S1 -> S2 [label="01/1"];
   S1 -> S3 [label="10/0"];
   S1 -> S4 [label="11/0"];
   S2 -> S1 [label="00/1"];
   S2 -> S2 [label="01/0"];
   S2 -> S3 [label="10/1"];
   S2 -> S4 [label="11/1"];
   S3 -> S1 [label="00/0"];
   S3 -> S2 [label="01/1"];
   S3 -> S3 [label="10/0"];
   S3 -> S4 [label="11/0"];
   S4 -> S1 [label="00/1"];
   S4 -> S2 [label="01/0"];
   S4 -> S3 [label="10/1"];
   S4 -> S4 [label="11/1"];
}

%> dot -T png case4.dot > case4.png
```



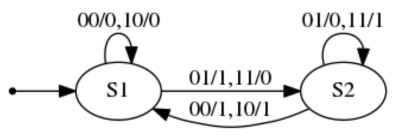
```
%> cat result.kiss
.start_kiss
.i 2
.o 1
.p 8
.s 2
.r S1
00 S1 S1 0
01 S1 S2 1
10 S1 S1 0
11 S1 S2 0
00 S2 S1 1
01 S2 S2 0
10 S2 S1 1
11 S2 S2 1
.end_kiss

%> cat result.dot
digraph STG {
   rankdir=LR;

   INIT [shape=point];
   S1 [label="S1"];
   S2 [label="S2"];

   INIT -> S1;
   S1 -> S1 [label="00/0,10/0"];
   S1 -> S2 [label="01/1,11/0"];
   S2 -> S1 [label="00/1,10/1"];
   S2 -> S2 [label="01/0,11/1"];
}

%> dot -T png result.dot > result.png
```



Two more examples for your reference (case5):

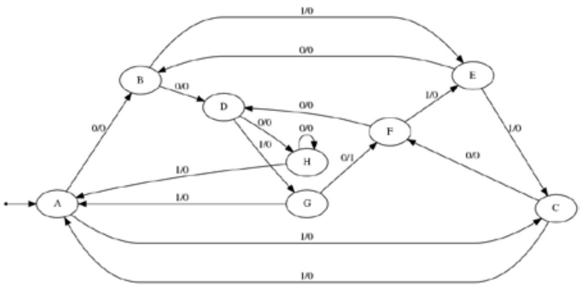
```
%> cat case5.kiss
.start_kiss
.i 1
.o 1
.p 16
.s 8
.r A
0 A  B 0
1 A  C 0
0 B  D 0
1 B  E 0
0 C  F 0
1 C  A 0
0 D  H 0
1 D  G 0
0 E  B 0
1 E  C 0
0 F  D 0
1 F  E 0
0 G  F 1
1 G  A 0
0 H  H 0
1 H  A 0
.end_kiss
```

```
%> cat case5.dot
digraph STG {
   rankdir=LR;

   INIT [shape=point];
   A [label="A"];
   B [label="B"];
   C [label="C"];
   D [label="D"];
   E [label="E"];
   F [label="F"];
   G [label="G"];
   H [label="H"];

   INIT -> A;
   A -> B [label="0/0"];
   A -> C [label="1/0"];
   B -> D [label="0/0"];
   B -> E [label="1/0"];
   C -> F [label="0/0"];
   C -> A [label="1/0"];
   D -> H [label="0/0"];
   D -> G [label="1/0"];
   E -> B [label="0/0"];
   E -> C [label="1/0"];
   F -> D [label="0/0"];
   F -> E [label="1/0"];
   G -> F [label="0/1"];
   G -> A [label="1/0"];
   H -> H [label="0/0"];
   H -> A [label="1/0"];
}
```

```
%> dot -T png case5.dot > case5.png
```



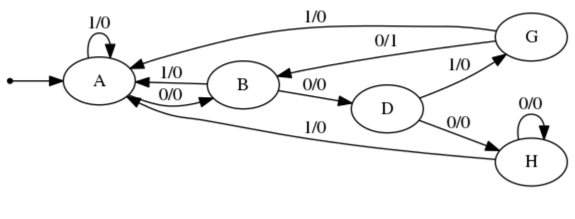
```
%> cat result.kiss
.start_kiss
.i 1
.o 1
.p 10
.s 5
.r A
0 A  B 0
1 A  A 0
0 B  D 0
1 B  A 0
0 D  H 0
1 D  G 0
0 G  B 1
1 G  A 0
0 H  H 0
1 H  A 0
.end_kiss
```

```
%> cat result.dot
digraph STG {
   rankdir=LR;

   INIT [shape=point];
   A [label="A"];
   B [label="B"];
   D [label="D"];
   G [label="G"];
   H [label="H"];

   INIT -> A;
   A -> B [label="0/0"];
   A -> A [label="1/0"];
   B -> D [label="0/0"];
   B -> A [label="1/0"];
   D -> H [label="0/0"];
   D -> G [label="1/0"];
   G -> B [label="0/1"];
   G -> A [label="1/0"];
   H -> H [label="0/0"];
   H -> A [label="1/0"];
}
```

```
%> dot -T png result.dot > result.png
```


A
B
D
G
H
1/0
0/0
0/1