UNIVERSIDADE FEDERAL DO RIO DE JANEIRO – UFRJ
PROGRAMA DE ENGENHARIA ELÉTRICA

ENTREGA DO TRABALHO 1

Aluno: Christiano Henrique Rezende

Matrícula: 120175020

Disciplina: CPE 743 - Controle Supervisório

Prof. Dr. Gustavo da Silva Viana

Fevereiro de 2021
Rio de Janeiro, RJ

QUESTÃO 1: EVENTO ILEGAL

Código desenvolvido em python:

```
from deslab import *

# Definindo Autômato G

syms('a b c')

X = [1,2,3,4,5]
Sigma = [a,b,c]
X0 = [1]
Xm = [2]
T = [(1,a,2),(1,b,3),(2,a,3),(3,b,2),(3,c,4),(5,a,3)]
G = fsa(X,Sigma,T,X0,Xm,name = 'G')

# Função que impede evento ilegal de acontecer
def IlegalEvent(G, event):
    S = sigmakleeneclos(G.Sigma)     # Faz fecho de Kleene com os eventos de G
    S = S.deletevent(event)          # Deleta o evento ilegal (esse é o Supervisor)
    Ha = product(G,S)                # Faz o produto de G x S
    Ha.name = "Ha"                   # Renomeia o autômato para Ha
    return Ha                        # Retorna autômato controlado

draw(G, IlegalEvent(G,c), 'figure')
```

Figuras geradas:

Fig. 0.0 G

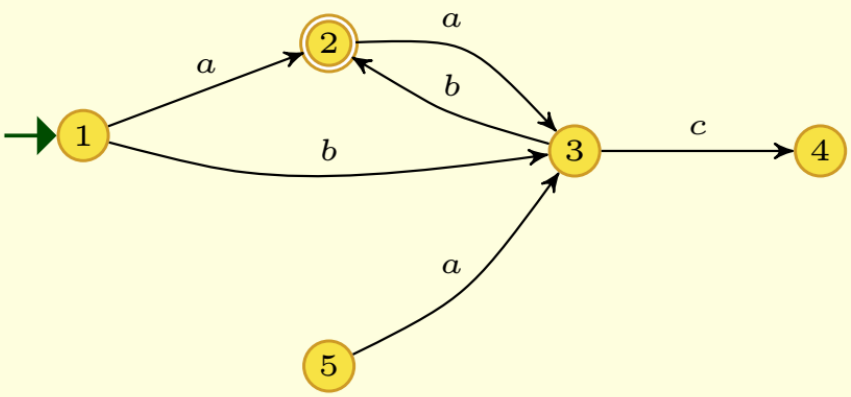
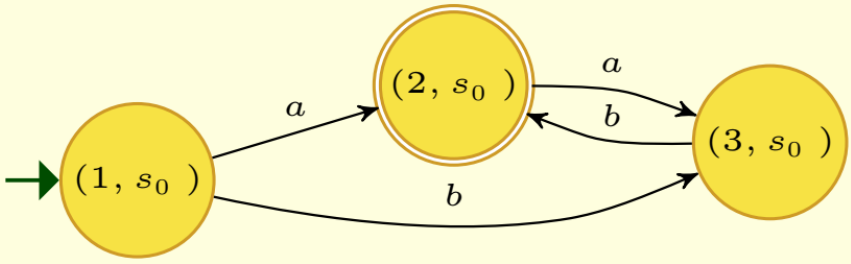


Fig. 0.1 Ha



QUESTÃO 2: ESTADO ILEGAL

Código desenvolvido em python:

```
from deslab import *

# Definindo Autômato G

syms('a b c')

X = [1,2,3,4,5]
Sigma = [a,b,c]
X0 = [1]
Xm = [2]
T = [(1,a,2),(1,b,3),(2,a,3),(3,b,2),(3,c,4),(5,a,3)]
G = fsa(X,Sigma,T,X0,Xm,name = 'G')

# Função que impede estado ilegal de acontecer

def IlegalState(G, state):
    S = deletestate(G, state)          # Deleta o estado ilegal
    S = ac(S)                          # Tira a parte acessivel (Esse é o Supervisor)
    Ha = parallel(G,S)                 # Faz o paralelo G|S
    Ha.name = "Ha"                     # Renomeia o autômato para Ha
    return Ha                          # Retorna autômato controlado

draw(G, IlegalState(G,3))
```

Figuras geradas:

Fig. 0.0 G

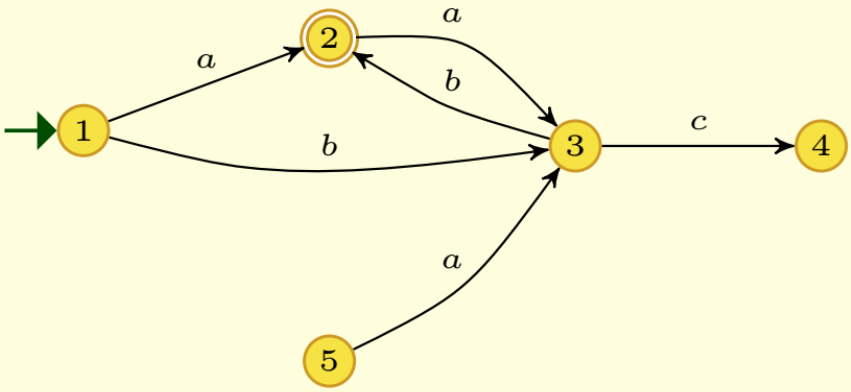
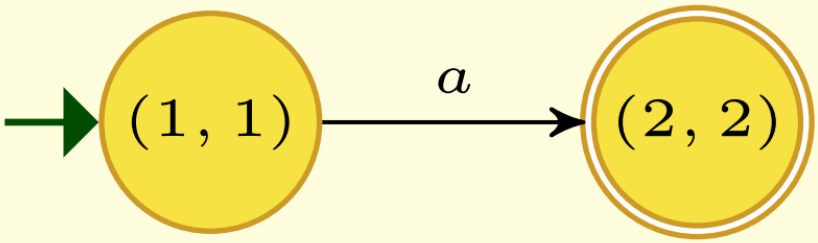


Fig. 0.1 Ha



QUESTÃO 3: ALTERNÂNCIA DE EVENTOS

Código desenvolvido em python:

```
from deslab import *

# Definindo Autômato G
syms('a b c')

X = [1,2,3,4,5]
Sigma = [a,b,c]
X0 = [1]
Xm = [2]
T = [(1,a,2),(1,b,3),(2,b,3),(3,a,2),(3,c,4),(5,a,3)]
G = fsa(X,Sigma,T,X0,Xm,name = 'G')

# Função que obriga a alternância dos eventos especificados
syms('S0 S1 S2 S3 S4')                  # Definindo nome dos estados de S

def EventAlternation(G, ev1, ev2):
    # Criando o Supervisor de alternância de eventos
    Xs = [S0, S1, S2, S3, S4]
    Sigma_s = [ev1, ev2]
    Ts = [(S0,ev1,S1),(S0,ev2,S3),(S1,ev2,S2),(S2,ev1,S1),(S3,ev1,S4),(S4,ev2,S3)]
    X0s = [S0]
    Xms = Xs
    S = fsa(Xs, Sigma_s, Ts, X0s, Xms, name = 'S')
    draw(S)

    Ha = parallel(G,S)          # Faz paralelo G|S
    Ha.name = "Ha"              # Renomeia o autômato para Ha
    return Ha                   # Retorna autômato controlado

draw(G, EventAlternation(G,a,b))
```

Figuras geradas:

Fig. 1.0 G

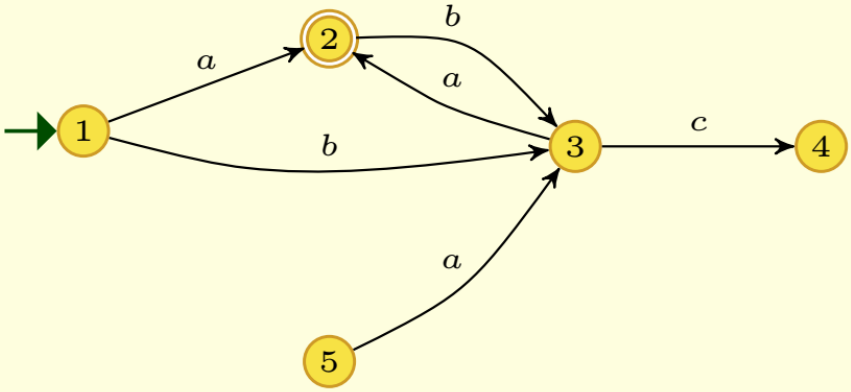


Fig. 0.0 S

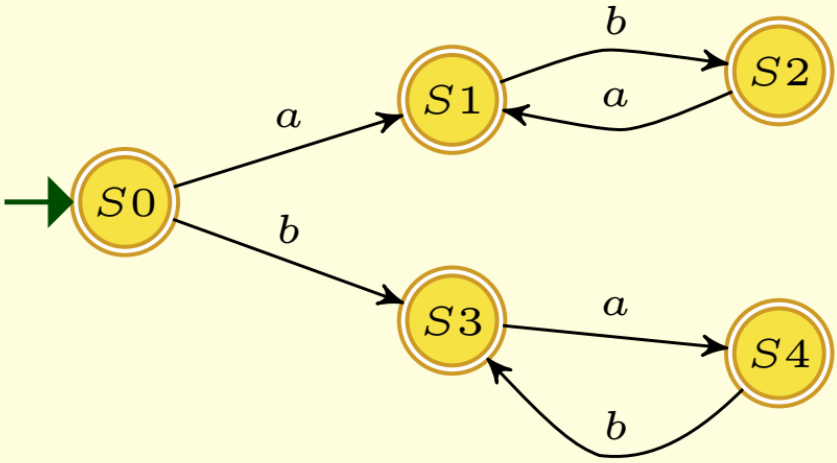
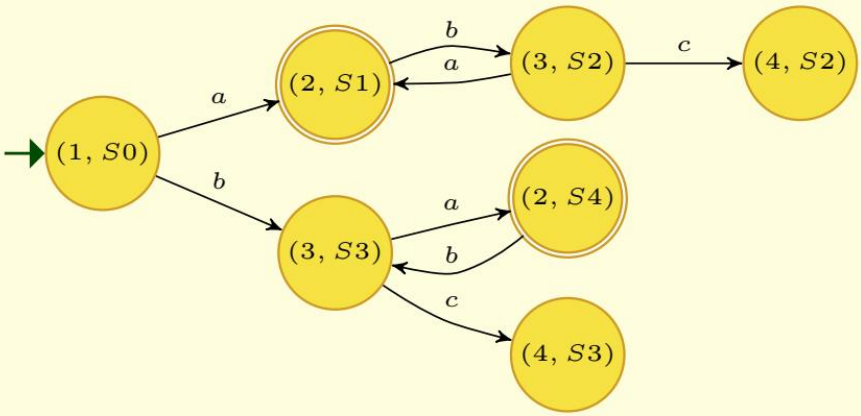


Fig. 1.1 Ha



QUESTÃO 4: SUBSEQUÊNCIA ILEGAL

Código desenvolvido em python:

```
from deslab import *

# Definindo Autômato G

syms('a b c')

X = [1,2,3,4,5]
Sigma = [a,b,c]
X0 = [1]
Xm = [2]
T = [(1,a,2),(1,b,3),(2,a,3),(3,b,2),(3,c,4),(5,a,3)]
G = fsa(X,Sigma,T,X0,Xm,name = 'G')

# Criando função que impede uma subsequencia ilegal de acontecer

def IlegalSubstring(G, substring):
    Xs = []       # Iniciando array de estados
    Ts = []       # Iniciando array de transiçõ

    # Criando os estados e transições da subsequencia ilegal
    for i in range(len(substring)):
        statename = 'S' + str(i)   # Nomeando o estado
        syms(statename)            # Criando Símbolo
        Xs.append(statename)   # Adicionando Estado no autômato
        # A partir do segundo estado criar transição que liga a ele
        if i>0:
            Ts.append((Xs[i-1],substring[i-1],Xs[i]))

    Xs0 = Xs[0]          # Estado inicial
    Xsm = Xs             # Marcando todos os estados

    # Varrendo os estados para criar as transições
    for x in Xs:
        ind = Xs.index(x)          # Guardando o índice do estado
        pref = substring[:ind]     # Guardando os eventos ocorridos ate o estado atual

        # Varrendo os eventos
        for event in Sigma:
            # Se for o evento da substring, a transição ja foi criada, então ignora
            if event != substring[ind]:
                # Concatenando os eventos até o estado com o evento da transição
                prefx = pref+event
                maxsuffix = 0   # Instanciando inteiro para comparação
```

```
            for i in range(ind+1):
                # Compara o sufixo da palavra que estou verificando
                # com o prefixo da substring ilegal
                if prefx[(len(prefx)-1)-i:] == substring[:(i+1)]:
                    # Se achou um sufixo em comum, verifica se vai para um
                    # estado maior e salva esse valor
                    maxsuffix = max(maxsuffix,(i+1))
            # Cria transição do estado para o índice que achou
            Ts.append((x,event,Xs[maxsuffix]))

    # Cria autômato supervisor S
    S = fsa(Xs,Sigma,Ts,Xs0,Xsm,name = 'S')
    draw(S)

    Ha = parallel(G,S)  # Faz o paralelo G|S
    Ha.name = 'Ha'      # Renomeia o autômato para Ha
    return Ha           # Retorna autômato controlado

draw(G, IlegalSubstring(G,"aac"))
```

Figuras geradas:

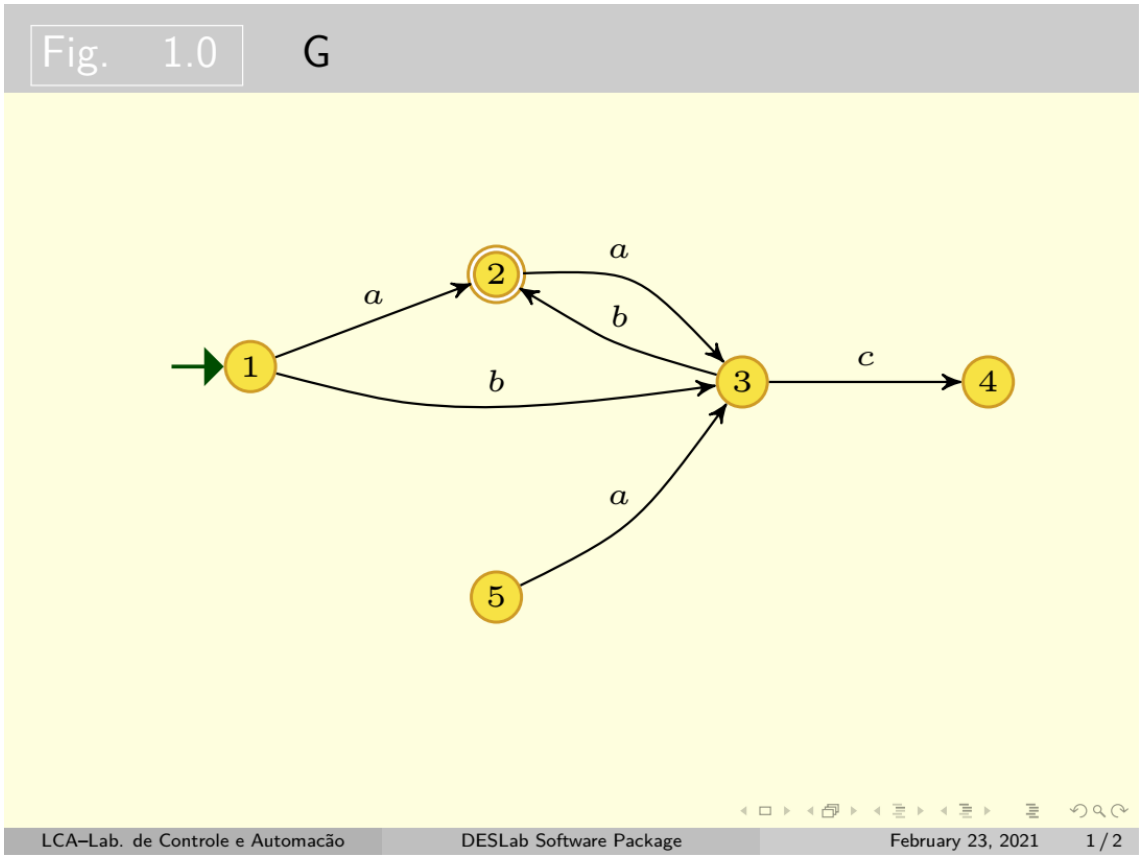


Fig. 0.0 S

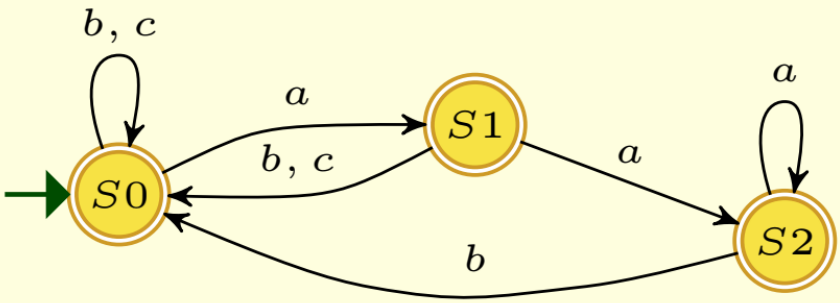


Fig. 1.1 Ha