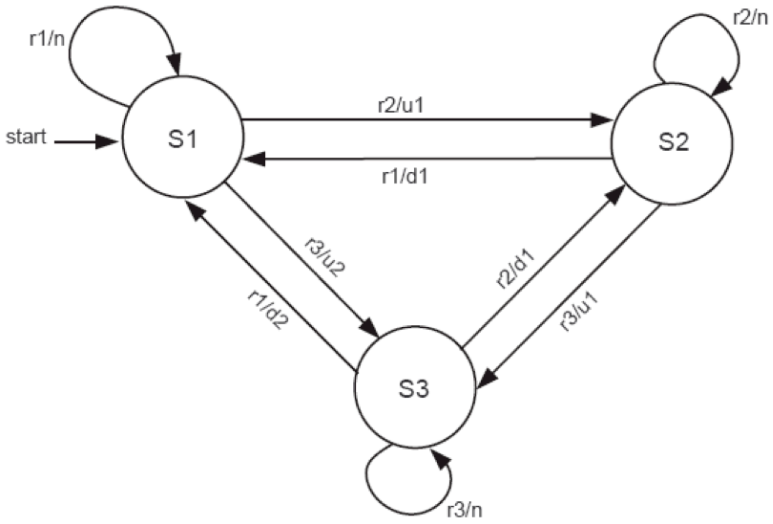
SÉTIMO TRABALHO DE INTRODUÇÃO AOS SISTEMAS EMBARCADOS
TRADUÇÃO DE MÁQUINAS DE ESTADOS FINITOS PARA CÓDIGO EM C

Considere **Máquina de Estados Finitos de Mealy**, ou seja, em cada transição há um evento de entrada e uma função a ser executada na saída (ou durante a transição).

Suponha a MEF da figura abaixo.



A linguagem para especificar tal MEF deve ser como abaixo.

```
fsm elevador {
    input = {r1, r2, r3};
    output = {u1, u2, d1, d2, nop};
    states = {S1, S2, S3};
    initial state = {S1};
    transitions {
        (S1, r1, nop, S1),
        (S1, r2, u1, S2),
        (S1, r3, u2, S3),
        (S2, r2, nop, S2),
        (S2, r1, d1, S1),
        (S2, r3, u1, S3),
        (S3, r3, nop, S3),
        (S3, r1, d2, S1),
        (S3, r2, d1, S2)
    }
}
```

Algumas observações sobre a especificação acima:

1) Não há uma ordem específica para a inclusão dos itens “input”, “output”, “states”, “initial state” e “transitions”, ou seja, tais itens podem vir intercalados;
2) Entretanto, “states” sempre precede “initial state”; e “input”, “output” e “initial state” sempre precede “transitions”. Por exemplo, a seguinte sequência é válida: output +

states + initial state + input + transitions. Outra sequência válida pode ser: states + initial state + input + output + transitions;

3) Dentro das "transitions", os estados iniciais também podem vir intercalados, isto é, nada impede que algum caso de teste tenha a seguinte entrada: transitions {(S1, r1, nop, S1), (S2, r1, d1, S1), (S1, r3, u2, S3), (S3, r1, d2, S1), (S1, r2, u1, S2)}
4) Assuma que todas as entradas serão entradas válidas.

A partir desta especificação deve ser gerado automaticamente a seguinte saída:

```
enum elevadorStates {S1, S2, S3};
enum elevadorStates state = S1;
enum elevadorInput {r1, r2, r3};

void u1();
void u2();
void d1();
void d2();
void nop();

void elevadorTransition (enum elevadorInput e)
{
    switch (state)
    {
        case S1:
            switch (e) {
                case r1: state=S1; nop(); break;
                case r2: state=S2; u1(); break;
                case r3: state=S3; u2(); break;
            }
            break;
        case S2:
            switch (e) {
                case r2: state=S2; nop(); break;
                case r1: state=S1; d1(); break;
                case r3: state=S3; u1(); break;
            }
            break;
        case S3:
            switch (e) {
                case r3: state=S3; nop(); break;
                case r1: state=S1; d2(); break;
                case r2: state=S2; d1(); break;
            }
            break;
    }
}
```

Vale a pena notar que tal saída pode ser totalmente gerada a partir da MEF de entrada. Obviamente que vou fazer testes com outras MEFs. Este trabalho pode ser feito em qualquer linguagem de programação, incluindo Shell Script, Python, C/C++, Java, dentre outras. Sugiro fortemente que se use Expressões Regulares para fazer tal tradução. Há diversos sites que

testam expressões regulares online. Por exemplo, http://www.regexp.com.br/ ou http://rubular.com/.

Data de Entrega: 07 de dezembro de 2018 (sábado) até meia-noite. A partir desta data será descontado 0.1 ponto por HORA de atraso.