

José Pereira (pg 27748), Marta Azevedo (pg27763), Tiago Brito (pg27724)

1 Elevador

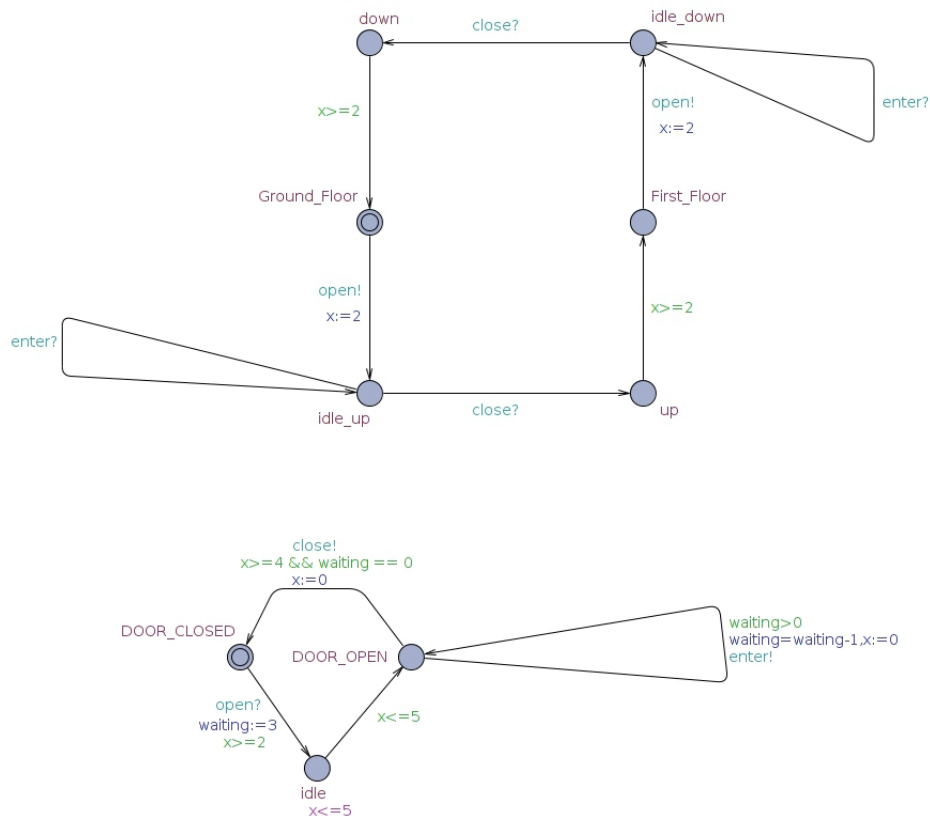
Neste exercício, foi pedido para modelar um elevador autónomo que anda entre dois andares - *Ground Floor* e *First Floor*. As condições são as seguintes:

- O elevador pode parar no *Ground Floor* ou no *First Floor*;
- Quando o elevador chega a um dado piso, a sua porta abre automaticamente.
- Leva 2 segundos a abrir a porta desde a sua chegada, mas a porta abre dentro de 5 segundos;
- Os passageiros podem entrar assim que a porta abrir;
- Os passageiros entram um por um e não há limite máximo de passageiros no elevador;
- A porta fecha 4 segundos depois do último passageiro entrar;
- Depois da porta fechar, o elevador espera 2 segundos e só então se desloca.

Para resolver este exercício, usámos 2 templates: *Elevator* e *Door*.

Foram necessários 5 canais de sincronização (*open, close, up, down, enter*), um relógio x que controla o tempo do elevador e uma variável *waiting* que significa o número de pessoas que está à espera para entrar no elevador.

Sendo assim, o nossos autómatos temporais são:



2 QoS

Neste exercício, devemos modelar um *stream channel* tal que:

1. A *Source* envia mensagens a cada 50ms;
2. O *Channel* pode perder mensagens mas não mais que 20%;
3. Se a mensagem não chegar ao *Receiver* em 90ms, é considerada perdida: *lost*;
4. O *Receiver* recebe mensagens e demora 5ms a processá-las;
5. Se em 1 segundo (1000 ms) não chegarem 15 mensagens, é gerada uma mensagem de erro: *error*.

Para resolver estes problemas, criamos 3 templates: *Soucer*, *Worker*, *Source_Worker* e inserimos as seguintes especificações para solucionar os problemas acima mencionados:

1. No *Source*, o clock é reiniciado e espera até chegar aos 50 ms para enviar a mensagem;
2. Há um contador para as mensagens enviadas (`msgSend`) e as que foram perdidas (`msgLost`) e também foi criada uma função chamada *value()*. Assim que as mensagens perdidas atingirem mais que 20% das mensagens enviadas, o autômato pára a sua execução;
3. Cada mensagem tem um relógio (solucionado através da introdução de um array de relógios). Quando o receiver recebe a mensagem, vê no respectivo relógio se já passaram 90ms. Se já tiver passado, envia uma mensagem a avisar que essa mensagem foi perdida (*lost!*) ;
4. Existe um nodo no *Receiver* chamado *Process* que só avança quando o seu temporizador chegar a 5ms;
5. Há um contador geral e cada vez que atinge os 1000 ms (1 segundo), o *Receiver* vê quantas mensagens já foram enviadas (`msgSend`), e se forem menos de 15 é disparado um sincronismo *error!*.

Para receber os sincronismos de falha (*error*, *lost*) e para receber o de sucesso *reply* existe o template *Source_Worker*. No caso dos sincronismos de falha, reinicia o autômato e no caso do *reply* incrementa o numero de mensagens enviadas com sucesso (`msgOK`) e reinicia o autômato.