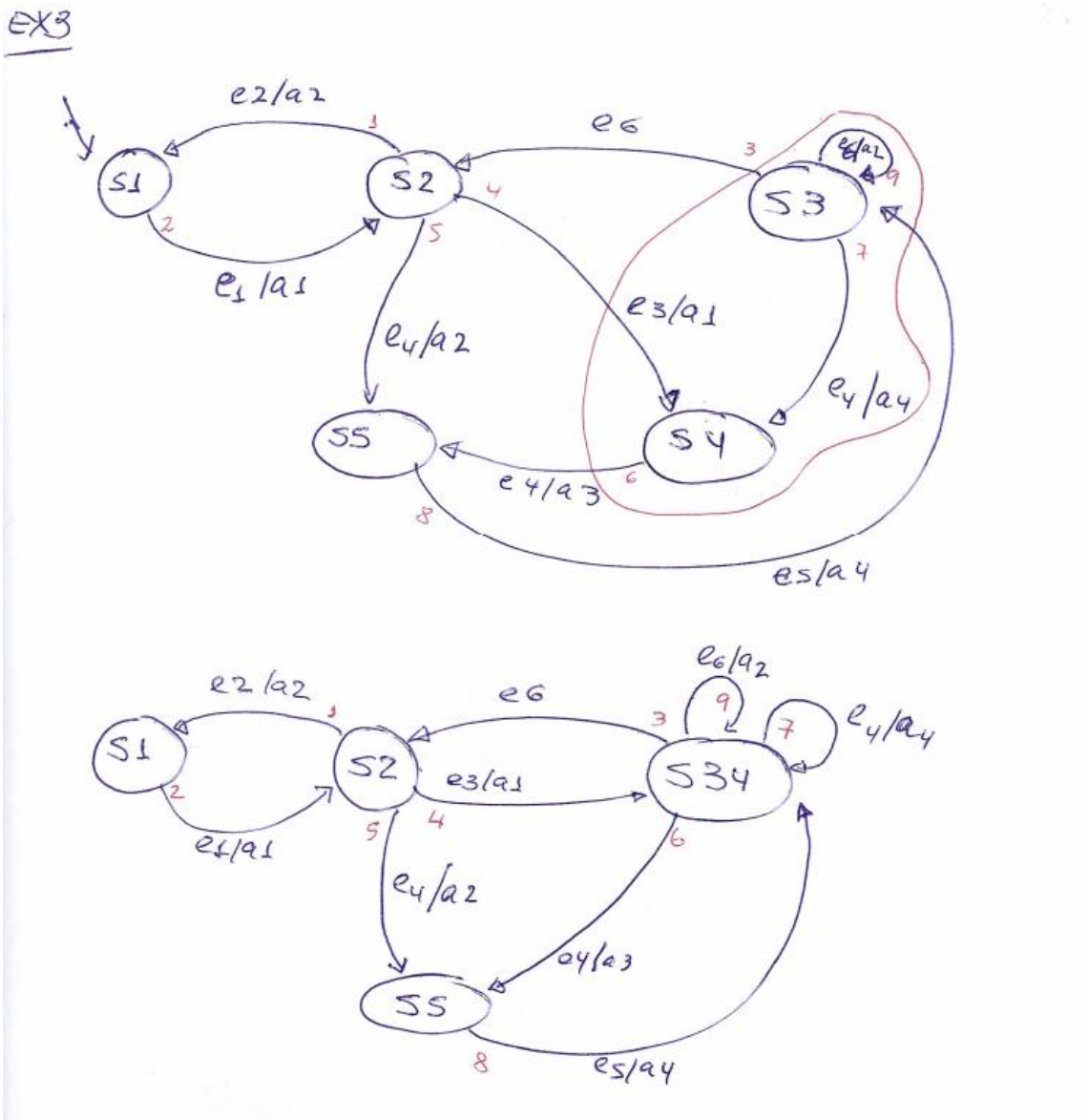
Exercício 2 - Transformações.

A data de entrega é a próxima quarta antes do início da aula, em pdf, pelo moodle/stoa. Discutiremos na aula as dúvidas e eventuais problemas encontrados e se for necessário posso depois reabrir o sistema para que vocês possam substituir o exercício.

Considerando o exemplo abaixo que mostra uma operação de junção de dois estados de um diagrama de estados



Note que:

1. Transição que liga dois estados a serem juntados são transformadas em auto-transições. Idem para auto-transições de estados a serem juntados.
2. Transições de e para os estados juntados são modificadas em sua origem ou destino.

3. Se ao menos um dos estados a serem juntados é um estado final, então o estado resultante também é final.
4. O diagrama de estado resultante pode nãos ser bem formado. Por exemplo: pode ocorrer que duas transições com a mesma origem tenham o mesmo evento de disparo. (Isso deveria ser evitado).

Faça:

1. Usando o metamodelo abaixo, faça um diagrama de objetos que represente as duas versões dos diagramas (modelos) mostrados no exemplo 3.

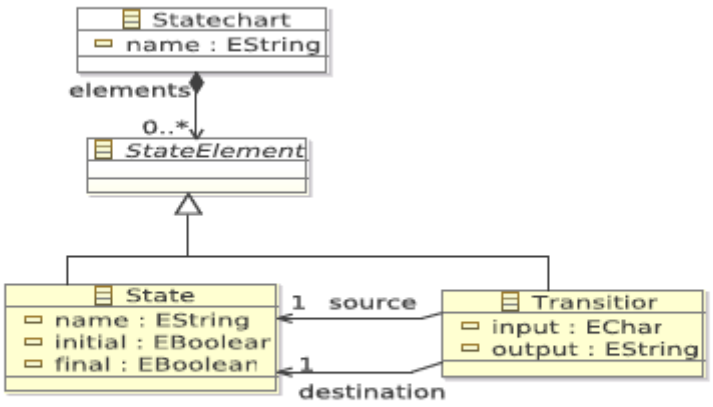


Figura 1: Diagrama de metaclasses do metamodelo.

2. Escreva uma transformação em ATL que , dados dois estados de um diagrama de estados, junte os dois estados, conforme mostrado acima. Sugiro que inicialmente façam a transformação usando uma linguagem tipo pseudo-código, usando regras sequenciais, IF e operadores lógicos. Se não conseguir fazer a regra, entregue pelo menos o pseudo-código.

Abaixo segue um exemplo de uma transformação feita pelo Thiago em ATL que copia um diagrama de estado para outro, só acrescentado um sufixo no nome do estado. A parte inicial em azul só faz algumas checagens e na prática poderia ser omitida. A primeira regra em cor de rosa transforma o nome do diagrama, a segunda em verde transforma os estados e a terceira em preto transforma as transições.

```
-- @path state=/State/model/state.ecore
module copiador;

create OUT: state from IN: state;
```

```
helper context state!Statechart def: isStatechartDefined():
Boolean =
      not (self.oclIsUndefined());

helper context state!State def: isStateDefined(): Boolean =
      not (self.oclIsUndefined());

helper context state!Transition def: isTransitionDefined():
Boolean =
      not (self.oclIsUndefined());


rule Statechart2Statechart {
      from
            scEntrada: state!Statechart (
                        scEntrada.isStatechartDefined()
                  )
      to
            scSaida: state!Statechart (
                  elements <- scEntrada.elements,
                  name <- scEntrada.name.concat('Copiado')
            )

}


rule State2State {
      from
            sEntrada: state!State (
                  sEntrada.isStateDefined()
            )
      to
            sSaida : state!State
            (
                  name <- sEntrada.name.concat('Copiado'),
                  initial <- sEntrada.initial,
                  final <- sEntrada.final
            )
}

rule Transition2Transition {
      from
            tEntrada: state!Transition (
                  tEntrada.isTransitionDefined()
            )
      to
            tSaida: state!Transition (
                  destination <- tEntrada.destination,
                  source <- tEntrada.source,
                  input <- tEntrada.input,
                  output <- tEntrada.output
            )
}
```