“Transducers” Project- Report

DESCRIÇÃO DAS OPÇÕES TOMADAS / COMENTÁRIOS DA SOLUÇÃO DESENVOLVIDA

Necessitamos de criar dois transducers auxiliares: ***e.fst*** e ***doubleZero.fst***. O primeiro processa o determinante “e” que pode ligar as horas e os minutos quando escritos por extenso, colocando “:” nos formatos em que tal é pedido. O segundo é auxiliar no *rich2num* e *lazy2num*, no caso em que o input/output é simplesmente uma hora sem minutos (exemplo: *dez*), colocando “:00” depois de processada.

Transducer ***text2num***: para este transducer foi necessário a utilização do transducer auxiliar ***e.fst*** para desta forma conseguir dar parse do ***e*** entre as horas e minutos. Depois foi necessária uma concatenação do ***horas.fst***, ***e.fst*** e o ***minutos.fst***.

Transducer ***lazy2num***: para resolver este problema realizamos uma primeira concatenação do transducer auxiliar **e.fst** com **minutos.fst**; posteriormente fazemos uma união com o transducer auxiliar ***doubeZero.fst****;* para finalizar fazemos uma concatenação com o transducer ***horas.fst***. Deste modo, com a união e as concatenações, obtemos um transducer que consiga resolver o input que contenha horas e minutos, ou apenas horas.

Transducer ***rich2text***: é de salientar a primeira concatenação entre os transducers ***horas*** e ***e****:* esta é feita juntamente com a flag *--project_type=input* do *fstproject*, visto que queremos fazer parsing das horas, mas não as queremos passar para o formato numérico.

Transducer ***rich2num***: para obtermos os resultados dos transducers ***meias*** e ***quartos*** em formato por extenso, concretizamos dois *fstcompose* com o transducer ***minutos***: desta forma, evitamos a repetição do transducer ***minutos*** e obtivemos apenas as ligações pretendidas. A utilização do transducer ***doubleZero.fst*** é feita logo depois de processarmos as horas: caso seja uma hora sem minutos, o processamento termina colocando “:00”; caso contrário, continua o processamento dos minutos.

Transducer ***num2text***: para obtermos este transducer começamos por usar o *fstinvert* nos seguintes transducers: ***horas.fst***, ***minutos.fst*** e ***e.fst***; posteriormente fazemos uma concatenação entre o ***inverted_horas.fst***, ***inverted_e.fst*** e ***inverted_minutos.fst***. Para resolver a obrigatoriedade do uso das expressões ***horas*** e ***minutos,*** usamos os pesos nos ramos **eps:eps** e **horas/minutos:eps** de 1 e 0.5 respetivamente; estes pesos constam nos ***horas.fst*** e ***minutos.fst*** originais, que se mantêm após o *fstinvert;* desta forma é escolhido sempre o arco de peso 0.5 e o transducer imprime sempre ***horas*** e ***minutos*****.**

ESTIMATIVA DA CONTRIBUIÇÃO DE CADA ELEMENTO PARA O TRABALHO / JUSTIFICAÇÃO

A contribuição dos elementos foi de 50% para cada. A distribuição de trabalho foi feita para ir ao encontro desta estimativa: os primeiros quatro transducers foram feitos dois a dois: um elemento fez os ***horas*** e ***quartos***, e o outro elemento fez os ***minutos*** e ***meias***; o transducer ***text2num*** foi feito em conjunto, para estudarmos as operações a realizar sobre transducers e conseguirmos ter uma ideia de como as realizar; finalmente, os transducers ***rich2text*** e ***rich2num*** foram feitos por um elemento e os transducers ***lazy2num*** e ***num2text*** foram feitos pelo outro elemento. No final, ambos analisamos todos os transducers resolvidos para prevenir erros.

Grupo 19 87636 - Bernardo Faria

87699 - Ricardo Gueifão