Conveter-

Na segunda secção desenvolvemos o programa converter.py, este recebe um conjunto de frases em lógica proposicional pelo stdin e retorna-as numa frase em clause normal form (CNF) pelo stdout.

Mais concretamente o programa recebe uma arvore sintática de tuples e retorna uma sequência de disjunções onde cada uma delas é uma lista de literais; cada literal pode ser um símbolo ou a negação de um símbolo.

O programa começa por aplicar as 4 regras de transformação de lógica proposicional para CNF (funções do ficheiro to\_cnf.py). Esta funções funcionam todas de forma recursiva aplicando as suas transformações para cada sub frase da árvore proposicional, percorrendo a árvore até a sua raíz de forma a que as transformações se apliquem a todas as relações proposicionais desta.

Primeiro aplicamos a transformação de eliminação de equivalências transformando-as em implicações através do método: (A <=> B) é o mesmo que se dizer ( 'and', ( '=>' , A , B ), ( '=>' , B , A ) ). De seguida aplicamos a segunda transformação de eliminação de implicações transformando-as em disjunções através do método: (A => B) é o mesmo que dizer ( 'or' , ( 'not', A ) , B ). Na terceira transformação removemos as negações de todas as disjunções ou conjunções de forma a que negações apenas aconteçam em literais, por exemplo: ( 'not', ( 'and', 'A' , 'B' ) ) é o mesmo que dizer ( 'or' , ( 'not' , 'A' ) , ( 'not' , 'B') ). Por ultimo é feito a transformação de distribuição de conjunções sobre disjunções de forma a que a frase resultante seja uma conjunção de disjunções, exemplo: ( 'or' , ( 'and', 'A' , 'B' ) , 'C') é o mesmo que dizer ( 'and', ( 'or', 'A' , 'C' ) , ( 'or', 'B' , 'C' ) ).

Estando a frase passada para a forma CNF, é desseguida feita uma transformação interna dos dados de forma a que a frase em vez de estar no formato de input passe a ser representada por um set de frozensets e strings (elementos que são separados por conjunções) que representam as clauses (disjunções) e literais (simples e negados). Para este efeito trabalhámos de novo recursivamente passando todos os literais de uma clause ou um literal negado para um frozenset enquanto que um literal simples é introduzido no set como uma string.

Tendo os dados neste formato vai ser muito mais fácil aplicar as transformações de simplificação. Em relação a estas criámos uma função para cada uma (funções disponíveis no ficheiro simplifications.py), exceto para a simplificação da factorização que nos diz que se um literal ocorrer mais do que uma vez numa clause a sua duplicação pode ser eleminada, que conseguimos garantir através da utilização dos frozensets para a representação de clauses. Na primeira simplificação removemos as clauses que contem literais que não são complementares com qualquer outra clause restante. Na segunda simplificação removemos todas as clauses que contenham um literal e a sua negação (tautologias). E por fim na ultima função de simplificação removemos todas as clauses que são implícitas por outras, isto é, caso uma clause for subset de outra a maior é eliminada.

Ficamos então com a frase CNF na forma mais simplificada possível e por fim esta é retornada pelo stdout no formato pretendido designado no enunciado do projecto.