Processamento de Linguagens e Compiladores
LCC (3ºano)

Trabalho Prático nº 2 (GIC/Yacc)

Ano lectivo 16/17

1 Objectivos e Organização

Este trabalho prático tem como principais **objectivos**:

- (genericamente) aumentar a experiência em *engenharia de linguagens* e em *programação generativa (gramatical)*;
- (especificamente) desenvolver processadores de linguagens segundo o método da *tradução dirigida pela sintaxe*, suportado numa gramática tradutora;
- (especificamente) utilizar *geradores de compiladores* baseados em *gramáticas tradutoras*, como o Yacc.

e como **objectivos** secundários:

- aumentar a experiência de uso do ambiente Linux, da linguagem imperativa C para codificação das estruturas de dados e respectivos algoritmos de manipulação, e de algumas ferramentas de apoio à programação;
- rever e aumentar a capacidade de escrever *gramáticas independentes de contexto* (GIC) que satisfaçam a condição LR() usando BNF-puro.

O programa desenvolvido será apresentado aos membros da equipa docente, totalmente pronto e a funcionar (acompanhado do respectivo relatório de desenvolvimento) e será defendido por todos os elementos do grupo, no mês de Janeiro, em dia a combinar.
O **relatório** a elaborar, deve ser claro e, além do respectivo enunciado, da descrição do problema, das decisões que lideraram o desenho da linguagem/gramática e as regras de tradução para Assembly da VM (incluir as especificações Yacc), deverá conter exemplos de utilização (textos fontes diversos e respectivo resultado produzido). Como é de tradição, o relatório será escrito em LaTeX.

2 Enunciado a

Numa 1ª fase deste TP, pretende-se que defina uma linguagem de programação imperativa simples (LPIS), a seu gosto. Deve ter em consideração que a LPIS terá de permitir declarar e manusear variáveis de vários tipos escalares ou estruturados e realizar as operações básicas como atribuições de expressões a variáveis, ler do *standard input* e escrever no *standard output*.
As instruções vulgares para controlo do fluxo de execução—*condicional* e *cíclica*—devem estar também previstas.
Como é da praxe neste tipo de linguagens, as variáveis deverão ser declaradas no início do programa e não pode haver re-declarações, nem utilizações sem declaração prévia; se nada for explicitado, o valor da variável após a declaração é 0 (zero).
Note que neste projeto, muitas tarefas são deixadas à criatividade do grupos de trabalho.
Escreva a sua GIC em notação do Yaccpara que possa gerar de imediato um parser e validar (pelo menos 6) programas-exemplo ou contra-exemplo.

3 Enunciado b

Numa 2ª fase deste TP, pretende-se que transforme uma GICdada numa *gramática tradutora* (GT) para resolver um problema concreto e produção automática de um resultado pedido com base na linguagem definida. Para o efeito são propostos abaixo 4 enunciados alternativos (a distribuir pelos grupos usando a fórmula dos TPs anteriores).

3.1 Stock e Faturas

Considere uma linguagem específica para descrever as Vendas duma empresa comercial, em que a ideia principal é descrever as várias Faturas sendo para isso necessário declarar inicialmente o Stock existente à partida (referência, descrição, preço unitário e quantidade em stock).
Sabe-se que uma Fatura é composta por um cabeçalho e um corpo, e este é composto por um ou mais movimentos. No cabeçalho deve aparecer o Número da Fatura, a Identificação do Fornecedor (nome, NIF, morada e NIB) e a Identificação do Cliente (nome, NIF, morada). Em cada linha do corpo da Fatura pretende-se, apenas incluir a Referência, e a Quantidade vendida.
A GIC abaixo define formalmente a linguagem Vendas, de acordo com a descrição dada:

```
T = { id, str, num, STOCK, FATURA, VENDAS, NIF:, NIB:, '.', ';', '-', '|'}
N = { Vendas, Stock, Entrada, Faturas, Fatura, Cabec, Corpo, IdFact, IdForn, IdClie, ..... }
S = Vendas
P = { Vendas   --> ´STOCK' Stock Faturas
      Stock    --> Entrada
                |  Stock ´;´ Entrada
      Entrada  --> RefProd ´-´ Desc '-' ValUnit '-' Quant
      Faturas  --> Fatura
                |  Faturas Fatura
      Fatura   --> 'FATURA´ Cabec 'VENDAS' Corpo
      Cabec    --> IdFat IdForn ´CLIENTE' IdClie
      IdFat    --> id
      IdForn   --> Nome Morada 'NIF:' NIF  'NIB:' NIB
      IdClie   --> Nome Morada 'NIF:' NIF
      Nome     --> str
      NIF      --> str
      Morada   --> str
      NIB      --> str
      Corpo    --> Linha '.'
                |  Corpo Linha '.'
      Linha    --> RefProd ´|´ Quant
      RefProd  --> id
      Desc     --> str
      ValUnit  --> num
      Quant    --> num
    }
```

Pretende-se então escreva uma *gramática tradutora* para :

a) por cada linha de uma fatura validar se o produto existe em stock e se a quantidade disponível é suficiente para a venda.

b) calcular o total por linha e total geral da fatura, bem como o total das vendas.

c) actualizar, por cada venda, a quantidade em stock.

Note que no fim do processamento o seu sistema deve listar o Stock final e uma listagem resumo das faturas.

3.2 Gestão de Clientes numa Farmácia

Considere uma linguagem específica para descrever Receitas e as Vendas de uma farmácia. Inicialmente descreve-se o Stock, indicando para cada medicamento—identificado pelo seu nome e código (supõe-se, para simplificar, só haver 1 tipo e 1 dosagem de cada um)—as versões de vários laboratórios: preço unitário e quantidade disponível. Depois descrevem-se os Clientes (número do cartão de cidadão, nome, morada, telef e NIF). Por fim virá então a lista das Receitas a processar, sendo que uma Receita é composta por um cabeçalho e um corpo, e este é composto por um ou mais prescrições. No cabeçalho deve aparecer o Número da Receita, a Identificação do Doente (número do seu cartão de cidadão) e a Data. Em cada linha do corpo da Receita pretende-se, apenas incluir o Código do medicamento, o Laboratório, e a Quantidade de embalagens.

A GIC abaixo define formalmente a linguagem Farma, de acordo com a descrição dada:

```
T = { str, num, data, STOCK, CLIENTES, RECEITA, MEDICACAO, LAB:, ';', '-', '+', ',', '.', '|'}
N = { Farma, Stock, Medicam, Labs, Lab, Clies, Clie, Receitas, Receita, Cabec, ..... }
S = Farma
P = { Farma   --> ´STOCK' Stock 'CLIENTES' Clies Receitas
      Stock    --> Medica
                |  Stock ´;´ Medica
      Medicam  --> CodMedica ´-´ Nome '-' Labs
      Labs     --> Lab
                |  Labs '+' Lab
      Lab      --> 'LAB:' CodLab ',' Preco ',' Qt
      Clies    --> Clie
                |  Clies ';' Clie
      Clie     --> CC ','  Nome ',' Morada ',' Telef ',' NIF
      Receitas --> Receita
                |  Receitas Receita
      Receita   --> 'RECEITA´ Cabec 'MEDICACAO' Corpo
      Cabec    --> NRec CC data
      Corpo    --> Linha '.'
                |  Corpo Linha '.'
      Linha    --> CodMedica Laborat ´|´ Qt
      Laborat  --> &
                |  '|' CodLab

      CodMedica--> str
      CodLab   --> str
      CC       --> str
      NRec     --> str
      Nome     --> str
      Telef    --> str
      Morada   --> str
      NIF      --> str
      Preco  --> num
      Qt       --> num
    }
```

Pretende-se então escreva uma *gramática tradutora* para :

a) por cada linha de uma receita validar se o cliente existe, se o medicamento existe em stock (atendendo ao laboratório, se indicado) e se a quantidade disponível é suficiente para a venda.

b) calcular o total por linha[1] e total geral da receita, criando uma conta-corrente para o cliente em que se regista para cada movimento apenas o código da receita, a data e o valor total.

[1] Note que não se o laboratório não for especificado, deve fornecer-se o mais barato.

c) actualizar, por cada venda, a quantidade em stock.

Note que no fim do processamento o seu sistema deve listar o Stock final e a conta-corrente de cada Cliente.

3.3 Planeamento de Espetáculos numa Escola de Música

Considere uma linguagem específica para descrever as Turmas de uma Escola de Música (EM) e o Programa de um ou mais Espetáculos. Inicialmente descrevem-se as Turmas e depois a data/hora do espetáculo e o Programa proposto. Por cada Turma será indicado o seu número de código, o nome do professor e o instrumento em causa, seguindo-se a lista de alunos (por cada um, o número de aluno, nome, contacto e idade).
Quanto ao Programa, indicam-se as várias sessões que o compõem, referindo para cada uma a hora de início e fim, o código da turma, o tipo instrumental e os alunos (cada qual identificado pelo seu número) que a vão integrar.
A GIC abaixo define formalmente a linguagem PrgEM, de acordo com a descrição dada:

```
T = { id, str, num, data, hr, TURMA:, ALUNOS, ESPETACULO, PROGRAMA, ';', '-', ',', '.', ':'}
N = { PrgEM, Turmas, Turma, Als, Alu, Al, Espetacs, Espetac, ..... }
S = PrgEM
P = { PrgEM    --> Turmas Espetacs
      Turmas   --> Turma
                |  Turmas ´;´ Turma
      Turma    --> ´TURMA:' CodT ´-´ NProf '-' NInst 'ALUNOS' Als
      Als      -->  Alu
                |   Als ';' Alu
      Alu      --> NuAl ',' NoAl ',' ContAl ',' IdAl
      Espetacs --> 'ESPETACULO' Espetac
                |  Espetacs 'ESPETACULO'  Espetac
      Espetac  --> Nome '-' data '/' hora '-' Local 'PROGRAMA' Sessoes
      Sessoes  --> Sessao '.'
                |  Sessoes Sessao '.'
      Sessao   --> HI '-' HF '-' NInst '-' CodT ':' Tocas
      Tocas    --> Al
                |  Tocas ',' Al
      Al       --> NuAl

      CodT     --> id
      NProf    --> str
      NInst    --> str
      NuAl     --> id
      NoAl     --> str
      ContAl   --> str
      IdAl     --> num
      Local    --> str
      HI       --> hora
      HF       --> hora
    }
```

Pretende-se então escreva uma *gramática tradutora* para :

a) por cada espetáculo validar se o espetáculo não começa depois das 21h nem antes das 15h, se as sessões são seguidas a partir do seu início, se a turma existe e se é dedicada ao instrumento referido e por fim se cada aluno existe nessa turma.

b) calcular o tempo total do espetáculo garantindo que não ultrapassa as 2 horas.

c) gerar um folheto em LaTeX com o programa "*bonitinho*" para anúncio de cada espetáculo.