COMP LabBook 2024 2 - E2

Lucas M. Schnorr

October 24, 2024

Contents

1 Logs de testes, tabelas de decisão e entradas de teste

Os arquivos fornecidos no `e2_CSV_LOG.zip`.

- `e2_make.log`: o arquivo contém o log da compilação
- `e2.log`: o arquivo contém o log cru dos testes
- `e2.csv`: após parsing para obter apenas a saída, as colunas são as seguintes:
  - Grupo: identificador do grupo
  - Teste: identificador do teste
  - Result: resultado da solução do grupo
    - 0, aceitou a entrada
    - 1, refugou a entrada
  - Razao
    - Vazio se aceitou a entrada
    - Erro sintático (possivelmente) informando o erro
- `e2_esperado.csv`: informada se a entrada é correta ou incorreta sintaticamente de acordo com a especificação da E2
  - Teste: identificador do teste
  - Esperado
    - 0, entrada sintaticamente correta de acordo com a E2
    - 1, entrada sintaticamente **incorreta** de acordo com a E2
- `e2_output_objetivo.csv`
  - E2.O: nota final do teste objetivo (sobre 10)
- `e2_output_quais_os_testes_mais_errados.csv`
  - Estatística que ilustra os testes que mais deram errado
- `e2_output_quem_errou_o_que.csv`
  - Indica apenas os testes que falharam, para cada grupo
  - Grupo, Test, Result, Esperado, Correto, Razao
    - Result: resultado da solução do grupo

    - Esperado: resultado esperado para aquele teste
  - Por exemplo:
    - Temos a linha `GrupoA,asl031,1,0,FALSE,line 4:  syntax error`
      - Esse teste foi considerado errado pois o GrupoA informou erro sintático na entrada asl031 (1 na coluna Result) quando na realidade sintaticamente o teste está correto (0 na coluna Esperado).
    - Para fazer essas verificações, consulte o TGZ (listagem abaixo).

Os arquivos fornecidos no `E2.tgz`.

- Cada arquivo tem um comentário //CORRECT ou //INCORRECT para facilitar a identificação se o teste está correto ou incorreto do ponto de vista da E2. Estes comentários são utilizados para definir o conteúdo de `e2_esperado.csv`.

2 Para todos: Entradas corretas identificada como erradas

As entradas abaixo estão sintaticamente corretas (de acordo com a E2). Verifique porque sua gramática considera elas falhas e faça as alterações para aceitá-las como sintaticamente corretas.

Grupo	Teste
GrupoC	asl014
GrupoC	asl015
GrupoC	asl037
GrupoC	asl038
GrupoF	asl006
GrupoF	asl014
GrupoF	asl015
GrupoF	asl017
GrupoF	asl027
GrupoF	asl029
GrupoF	asl037
GrupoF	asl038
GrupoF	asl044
GrupoF	asl056
GrupoF	asl058
GrupoF	asl062
GrupoF	asl081
GrupoK	asl100
GrupoK	asl101
GrupoL	asl100
GrupoL	asl101
GrupoT	asl100
GrupoT	asl101
GrupoZ	asl005
GrupoZ	asl006

3 Para todos: Entradas incorretas mas aceitas

As entradas abaixo estão sintaticamente incorretas (de acordo com a E2). Verifique porque sua gramática considera elas corretas e faça as alterações para não aceitá-las como sintaticamente corretas.

Grupo	Teste
GrupoB	asl047
GrupoF	asl020
GrupoF	asl047
GrupoF	asl076
GrupoI	asl047
GrupoK	asl047
GrupoM	asl047
GrupoN	asl047
GrupoP	asl047
GrupoR	asl047
GrupoZ	asl047

4 Comentários Gerais

- Em todas as regras com recursão gramatical (listas de comandos, listas de funções, etc), dar preferência para a recursão à esquerda pois isso facilita as regras de associatividade (em expressões, por exemplo).

- Usar os contra-exemplos (rodar o `bison` com o parâmetro `-Wcounterexamples`) que demonstram as situações dos conflitos e auxiliam o grupo a entender como modificar a gramática para evitar aquele conflito.
  - Os contra-exemplos indicam os caminhos equivalentes para chegar no conflito. Deve-se editar a gramática então para que tenha um caminho único de derivações para chegar até a situação do contra-exemplo.

Os comentários abaixos focam na observação do código e a experiência do professor da execução dos testes com o código do grupo. Os comentários abaixo **não** mencionam os erros detectados através dos testes automáticos. Para estes, a sugestão é olhar o ZIP e o TGZ e a introdução deste relatório para decodificar os ajustes que devem ser feitos. Caso houverem dúvidas, entre em contato.

5 Histórico de comentários

5.1 GrupoA

5.1.1 E2

- Preliminar: arquivo parser.y não submetido
  - Submetido na versão definitiva
- Poderiam documentar o comando `%locations` e o que ele faz

5.1.2 E1

☐ Para os tokens especiais, pode-se colocá-los todos na mesma regra
  - E usar yytext[0] ao invés de explicitamente usar o literal

☐ Quebra de linha é considerado espaço em branco (pode-se colocar na mesma regra)

5.2 GrupoB

5.2.1 E2

- É necessário atualizar o arquivo `README.md` na medida que o projeto avança
  - Por exemplo, a listagem da estrutura do projeto

5.2.2 E1

- Usando Makefile como um "wrapper" do CMake (só para deixar anotado)
- Boa documentação do projeto com o arquivo `README.md`

☒ Espaços ignorados podem ser aglutinados em uma única regra
  - Quebras de linha são considerados espaços

☐ Na hora de montar o pacote tgz, por favor, remover todos os arquivos "ocultos" que começam por ".". Eles não aparecem na saída do comando `ls`, mas o `tar` os vê e os inclui. Informar ao `tar` para não inclui-los.

5.3 GrupoC

5.3.1 E2

- Funções "estranhas" (não usadas) no final do parser.y
  - Exemplo, `parse_string`. Imagino que sejam para um uso alternativo de funcionamento do parser, visto que envolvem uma chamada à `yyparse`.
- No arquivo main.c, o include do `parser.tab.h` possui um caminho relativo. Em geral isso não é uma boa prática, sendo preferível informar ao compilador onde procurar arquivos de cabeçalhos via o parâmetro `-I` (neste caso, poderia ser algo como `-I./obj/` em algum lugar do `Makefile`).

5.3.2 E1

☒ O arquivo scanner.l, contendo código, normalmente fica em diretórios `src`

☒ O arquivo Makefile não segue a ideia de compilação separada por arquivo, algo em geral benéfico até para projetos pequenos.
  - O alvo `test` não funciona, provavelmente porque o conteúdo de `src/testing` foi removido do pacote
  - Pode-se criar uma arquivo `Makefile.alt` somente para testes
    * Removê-lo do tgz na hora de submeter

5.4 GrupoD

5.4.1 E2

- A regra `expressao` não segue a especificação da E2 pois não implementa precedência de operadores (Sec 3.4 da E2). Para que a precedência possa ser implementada, precisas criar "níveis", por exemplo, no primeiro nível é o or, depois o and, e assim por diante. Seria algo assim:

  ```
  expressao: expressao TK_OC_OR exp1 | exp1
  exp1: exp1 TK_OC_AND exp2 | exp2
  exp2: ...
  ```

  Podemos trocar uma ideia caso ainda não tiverem entendido.

5.4.2 E1

☐ Para os tokens especiais, pode-se colocá-los todos na mesma regra

  – E usar yytext[0] ao invés de explicitamente usar o literal

☐ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

5.5 GrupoE

5.5.1 E2

- O que é a "metodologia de tdd"? (comentário no `scanner.l`)
- A regra `expressao` não segue a especificação da E2 pois não implementa precedência de operadores (Sec 3.4 da E2). Para que a precedência possa ser implementada, precisas criar "níveis", por exemplo, no primeiro nível é o or, depois o and, e assim por diante. Seria algo assim:

  ```
  expressao: expressao TK_OC_OR exp1 | exp1
  exp1: exp1 TK_OC_AND exp2 | exp2
  exp2: ...
  ```

  Podemos trocar uma ideia caso ainda não tiverem entendido.

5.5.2 E1

☐ Para os tokens especiais, pode-se colocá-los todos na mesma regra

  – E usar yytext[0] ao invés de explicitamente usar o literal

5.6 GrupoF

5.6.1 E2

- Vários testes falham. Olhando rapidamente, parece que o problema tem a ver com a `expressao` que deve virar um `literal` ou apenas com `literal`.

5.6.2 E1

☐ O Makefile não segue a filosofia geral para construção de projetos, pois possui listagens de código nas dependências e não possui uma regra de compilação intermediária de código objeto que permite a compilação parcial do projeto.

☒ Os espaços podem ser aglutinados em uma única regra

☒ Para os tokens especiais, pode-se colocá-los todos na mesma regra

  – E usar yytext[0] ao invés de explicitamente usar o literal

5.7 GrupoG

5.7.1 E2

- Nenhum comentário.

5.7.2 E1

☒ Arquivos devem estar na raiz

☒ Normalmente evita-se de #include com um caminho relativo (ou absoluto)

  – No caso farias somente #include "tokens.h", instruindo o compilador (com -I) a procurar os cabeçalhos em determinado lugar

☒ Para os tokens especiais, pode-se colocá-los todos na mesma regra

  – E usar yytext[0] ao invés de explicitamente usar o literal

☒ A regra do barra-enne é redundante com a classe [:space:]

☒ Boa organização em subdiretórios, mas o Makefile precisa melhorar visto que não é uma boa prática em receitas misturar entradas .c e .o como no alvo $(ETAPA). Além disso, a compilação de .o pode ser via regra única por intermédio de wildcards, conforme visto no tutorial indicado.

☒ Arquivos `deliver.sh`, `tester.py`, `tests` podem ser omitidos do tgz

5.8 GrupoH

5.8.1 E2

- Nenhum comentário.

5.8.2 E1

☐ O caractere barra-ene está incluso na classe `[:blank:]`, portanto temos uma regra redundante

☒ O Makefile não segue a filosofia geral para construção de projetos, pois possui listagens de código nas dependências e não possui uma regra de compilação intermediária de código objeto que permite a compilação parcial do projeto.

5.9 GrupoI

5.9.1 E2

- Arquivos devem estar na raiz

  – O peso deste comentário aumentou

5.9.2 E1

☐ Arquivos devem estar na raiz

☒ Deve-se evitar colocar a implementação de funções no cabeçalho do scanner, priorizando a última seção do arquivo ou em arquivos suplementos.

☒ Ao invés de implementar yywrap, pode-se usar a opção para desabilitar essa funcionalidade.

- [/] Alvos e receitas do makefile são majoritariamente manuais, sem wildcards. O makefile pode ficar bem mais sucinto se empregar os conhecimentos do tutorial indicado.

  – Além disto, possui regras específicas da etapa1, mesmo sabendo que temos outras etapas por vir.

5.10 GrupoJ

5.10.1 E2

- Preliminar: calculou a coluna do erro sintático!
- No arquivo main.c, o include do `parser.tab.h` possui um caminho relativo. Em geral isso não é uma boa prática, sendo preferível informar ao compilador onde procurar arquivos de cabeçalhos via o parâmetro `-I` (neste caso, poderia ser algo como `-I./obj/` em algum lugar do `Makefile`).
- O arquivo `Makefile` evolui bastante, mas de uma maneira geral ele ficou muito complexo (veja o tamanho). Poderia ficar bem mais simples, sobretudo pela característica ainda pequena de nosso projeto.
- Veja os pontos não marcados como feitos abaixo; entendo que eles ainda estão em aberto.

5.10.2 E1

☐ As ações associadas às regras estão com identação diferente, algumas alinhadas outras não, no arquivo scanner.l

☐ O makefile parece ter código boilerplate que não se aplica neste projeto, pois ele vê se existe um subdiretório src, adaptando-se em função. Se o grupo não tem a intenção de organizar em subdiretórios, sugiro simplificar o Makefile.

☐ Não se usa a filosofia de compilação parcial dos arquivos (-c), ou seja, sempre se compila tudo novamente.

☒ A variável tar file pode ser definida a partir do nome de binary.

5.11 GrupoK

5.11.1 E2

☒ Preliminar: arquivos devem estar na raiz

5.11.2 E1

☐ Não há necessidade de stdio.h no arquivo scanner.l

☐ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

5.12 GrupoL

5.12.1 E2

- Nenhum comentários.

5.12.2 E1

☐ Normalmente o ; nos comandos em C ficam imediatamente após a último token do comando, sem espaços.

☐ Não há necessidade de incluir stdio.h. Além disso, cabeçalhos de bibliotecas de sistema são incluídas com <stdio.h> ao invés de "stdio.h".

☐ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

☒ Evitar de empacotar o diretório `testes` (e os arquivos `quero.*\.sh`).

5.13 GrupoM

5.13.1 E2

☒ Preliminar: usou o comando %left, em desacordo com a especificação E2

- Usar wildcards do makefile para simplificar as regras

5.13.2 E1

☒ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

☒ Não há necessidade de incluir stdio.h.

☐ Os espaços ignorados podem ser aglutinados em uma única regra

5.14 GrupoN

5.14.1 E2

- Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

5.14.2 E1

☐ Na hora de montar o pacote tgz, por favor, remover todos os arquivos "ocultos" que começam por ".". Eles não aparecem na saída do comando `ls`, mas o `tar` os vê e os inclui. Informar ao `tar` para não inclui-los.

☒ Não temos aspas simples

☒ Não temos comentários multilinha portanto não há necessidade de %x

5.15 GrupoO

5.15.1 E2

- A regra `expressao` não segue a especificação da E2 pois não implementa precedência de operadores (Sec 3.4 da E2). Para que a precedência possa ser implementada, precisas criar "níveis", por exemplo, no primeiro nível é o or, depois o and, e assim por diante. Seria algo assim:

  ```
  expressao: expressao TK_OC_OR exp1 | exp1
  exp1: exp1 TK_OC_AND exp2 | exp2
  exp2: ...
  ```

  Podemos trocar uma ideia caso ainda não tiverem entendido.

- Melhorar a identação das receitas do makefile, um tab é o suficiente.

- Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

5.15.2 E1

☒ Os espaços ignorados podem ser aglutinados em uma única regra

  – Inclusive o barra-enne poderia ser aglutinado visto que trata-se de um espaço também

☐ Comentários são legais, mas melhor se estiverem não todos em maiúscula

  – Para não confundir com constantes do código

5.16 GrupoP

5.16.1 E2

- Melhor identificar que o número que aparece no relatório de erro é um número de linha

- Documentar melhor as regras da gramáticas, as agrupando e colocando espaços entre categorias de regras, de maneira a facilitar a leitura da gramática.

5.16.2 E1

☐ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

5.17 GrupoQ

5.17.1 E2

5.17.2 E1

- Não submetido

5.18 GrupoR

5.18.1 E2

- Preliminar: melhor identificar que o número que aparece no relatório de erro é um número de linha

5.18.2 E1

☒ Na função `get_line_number`, corrigir a identação.

☒ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

  – Ter um alvo que empregue o parâmetro `-c`

5.19 GrupoS

5.19.1 E2

- Nos comentários que categorizam as regras gramaticais, usar texto não em uppercase total.

5.19.2 E1

☒ Ignorar também o caractere tab com barra-t

☒ Aglutinar caracteres ignorados (espaços) em uma única regra

☒ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

5.20 GrupoT

5.20.1 E2

- Remover o arquivo tokens.h conforme recomendado na especificação E2
- No makefile, a compilação de .o pode ser via regra única por intermédio de wildcards, conforme visto no tutorial indicado.

5.20.2 E1

☒ O arquivo scanner.l, contendo código, normalmente fica em diretórios `src`

☒ O arquivo Makefile não segue a ideia de compilação separada por arquivo, algo em geral benéfico até para projetos pequenos.

☒ Para os tokens especiais, pode-se colocá-los todos na mesma regra

  – E usar yytext[0] ao invés de explicitamente usar o literal

5.21 GrupoZ

5.21.1 E2

- Retomo o comentário deixado pelo grupo "TODO: Check for precedencia"
  – Realmente ficou faltando.

5.21.2 E1

☒ Melhorar o makefile para que se possa usufruir de um sistema de compilação que permita compilação parcial dos vários fontes do projeto.

6 Subjetivo

6.1 Tabela com valores dos critérios por grupo

Grupo	Nota	Critério
GrupoA	10	Gramática
GrupoB	9	Gramática
GrupoC	9	Gramática
GrupoD	10	Gramática
GrupoE	10	Gramática
GrupoF	8	Gramática
GrupoG	10	Gramática
GrupoH	10	Gramática
GrupoI	9	Gramática
GrupoJ	10	Gramática
GrupoK	9	Gramática
GrupoL	9	Gramática
GrupoM	9	Gramática
GrupoN	9	Gramática
GrupoO	10	Gramática
GrupoP	9	Gramática
GrupoR	9	Gramática
GrupoS	10	Gramática
GrupoT	9	Gramática
GrupoZ	9	Gramática
GrupoA	10	Relatório de Erro
GrupoB	10	Relatório de Erro
GrupoC	10	Relatório de Erro
GrupoD	10	Relatório de Erro

Continued on next page

Continued from previous page

Grupo	Nota	Critério
GrupoE	10	Relatório de Erro
GrupoF	5	Relatório de Erro
GrupoG	10	Relatório de Erro
GrupoH	10	Relatório de Erro
GrupoI	10	Relatório de Erro
GrupoJ	10	Relatório de Erro
GrupoK	10	Relatório de Erro
GrupoL	10	Relatório de Erro
GrupoM	10	Relatório de Erro
GrupoN	10	Relatório de Erro
GrupoO	10	Relatório de Erro
GrupoP	7.5	Relatório de Erro
GrupoR	7.5	Relatório de Erro
GrupoS	10	Relatório de Erro
GrupoT	10	Relatório de Erro
GrupoZ	10	Relatório de Erro
GrupoA	10	Remoção de Conflitos
GrupoB	10	Remoção de Conflitos
GrupoC	10	Remoção de Conflitos
GrupoD	10	Remoção de Conflitos
GrupoE	10	Remoção de Conflitos
GrupoF	10	Remoção de Conflitos
GrupoG	10	Remoção de Conflitos
GrupoH	10	Remoção de Conflitos
GrupoI	10	Remoção de Conflitos
GrupoJ	10	Remoção de Conflitos
GrupoK	10	Remoção de Conflitos
GrupoL	10	Remoção de Conflitos
GrupoM	10	Remoção de Conflitos
GrupoN	10	Remoção de Conflitos
GrupoO	10	Remoção de Conflitos
GrupoP	10	Remoção de Conflitos
GrupoR	10	Remoção de Conflitos
GrupoS	10	Remoção de Conflitos
GrupoT	10	Remoção de Conflitos
GrupoZ	10	Remoção de Conflitos
GrupoA	9.5	Opinião
GrupoB	9.75	Opinião
GrupoC	9.75	Opinião
GrupoD	6	Opinião
GrupoE	6.5	Opinião
GrupoF	9	Opinião
GrupoG	10	Opinião
GrupoH	10	Opinião
GrupoI	9	Opinião
GrupoJ	8.5	Opinião
GrupoK	9.5	Opinião
GrupoL	9.5	Opinião
GrupoM	9.75	Opinião
GrupoN	9.75	Opinião
GrupoO	6.5	Opinião
GrupoP	9.5	Opinião
GrupoR	9.75	Opinião
GrupoS	10	Opinião
GrupoT	9.5	Opinião
GrupoZ	7	Opinião

6.2 Justificativas da tabela

6.2.1 Gramática (Sec 2.1)

Alinha-se com os resultados objetivos, com a função `trunc` (obtém o valor inteiro do ponto-flutuante).

6.2.2 Relatório de Erro (Sec 2.2)

- Nota 10: grupo apresentou solução com "mensagem de erro informando a linha do código da entrada que gerou o erro sintático e informações adicionais que auxiliem o programador que está utilizando o compilamensagem de erro informando

a linha do código da"

- Nota 5: grupo apresentou solução que apenas identifica o número da linha onde aconteceu o erro sintático, sem informar qual foi o erro especificamente OU grupo apresentou solução que apenas identifica o erro, sem mencionar o número da linha
- Nota 0: grupo apresentou solução que apenas identifica que houve um erro, sem informar o número da linha ou qual foi o erro, ou não entregou.

Sugestão:

- Ative `%define parse.error verbose` no parser.y

6.2.3 Remoção de Conflitos (Sec 2.3)

- Nota 10: sem conflitos shift/reduce ou reduce/reduce
- Nota 5: apenas um conflito shift/reduce ou reduce/reduce
- Nota 0: com pelo menos um conflito acima ou `expect` não justificado

6.2.4 Opinião

Baseada na combinação da análise objetiva, com relatório de erro, remoção de conflitos e gramática, além da revisão das regras gramaticais, adequação às regras gerais, e demais comentários ressaltados acima.

7 Final

Grupo	E2.O	E2.S	E2.P	E2
GrupoA	10	9.65	1	9.82
GrupoB	9.91	9.72	1	9.82
GrupoC	9.65	9.72	1	9.69
GrupoD	10	7.2	1	8.6
GrupoE	10	7.55	1	8.78
GrupoF	8.6	8.6	1	8.6
GrupoG	10	10	1	10
GrupoH	10	10	1	10
GrupoI	9.91	9.2	1	9.55
GrupoJ	10	8.95	1	9.47
GrupoK	9.74	9.55	1	9.64
GrupoL	9.82	9.55	1	9.69
GrupoM	9.91	9.72	1	9.82
GrupoN	9.91	9.72	1	9.82
GrupoO	10	7.55	1	8.78
GrupoP	9.91	9.3	1	9.61
GrupoR	9.91	9.47	1	9.69
GrupoS	10	10	1	10
GrupoT	9.82	9.55	1	9.69
GrupoZ	9.74	7.8	1	8.77

8 Recuperação

Nenhum grupo em recuperação.