Compiladores - Prova 1

Prof. Fábio Macêdo Mendes (5/10/2017)

Nome:

Matrícula:

Instruções:

- A prova é individual e sem consulta.
- Serão aceitas respostas a lapis ou caneta.
- Mantenha as respostas organizadas e legíveis.
- Cada questão da prova possui uma versão "jedi" (mais difícil, mas com um valor maior) e uma versão "padawan" (mais fácil, mas valendo menos). Você deve optar e resolver apenas uma das duas versões de cada questão.
- Indique claramente se você optou pela versão "jedi" ou "padawan" de cada questão circulando o título correspondente. Faça isto ao final da prova para evitar rasuras.

Q1-jedi (2.0 pts): Gramáticas simples

A gramática abaixo define uma linguagem baseada em expressões que envolvem as letras "a" e "b".

```
E = a E b E
  | b E a E
  | a
```

a) A expressão "baa" faz parte desta linguagem? Explique.

b) Monte uma árvore sintática válida para a expressão "abaaaba".

c) Mostre por indução que todas as expressões da linguagem possuem um número de ocorrências de "a" maior que de "b".

Q1-padawan (1.5pts): Gramáticas simples

A gramática `E = E * E | a`, que representa todas as expressões do tipo `a*a*a*...`, é uma gramática ambígua.

a) Mostre que a expressão `a*a*a` pode ser derivada de duas maneiras diferentes (escreva as respectivas derivações ou árvores sintáticas).

b) Elimine a ambiguidade da gramática propondo uma nova gramática que force a associatividade à esquerda.

Q2-jedi (3.0 pts): Linguagens regulares

A expressão abaixo define uma linguagem que descreve números com casas decimais:

`N = (-|+| )(dd*)(.d*)*`

O símbolo "d", na expressão acima representa qualquer dígito de zero a nove. " " representa a string vazia e os símbolos "(", ")", "|" e "*" possuem os significados usuais em expressões regulares. Todos os outros símbolos correspo ndem a seus valores literais.

a) Marque as expressões abaixo que correspondem a sentenças válidas segundo esta linguagem regular.

[ ] `42`	[ ] `+42.`	[ ] `-42`
[ ] `3.1415`	[ ] `.15`	[ ] `1.2e10`
[ ] `--1.5`	[ ] `1.2.3`	[ ] `1..10`

(Cada item correto vale +0.2 e cada item incorreto vale -0.2)

b) Modifique a linguagem regular acima para que ela deixe de aceitar os valores inválidos na última linha.

c) Faça uma extensão para que a parte antes da vírgula também possa conter opcionalmente números descritos com separadores de milhar. Os separadores de milhar podem aparecer a cada grupo de três dígitos como em `1_234_567.89` (leia-se um milhão, duzentos e trinta e quatro mil, quinhentos e sessenta e sete ponto oitenta e nove).

Nas duas respostas, espaços e quebras de linha serão ignorados na expressão regular para que você possa organizá-la com mais clareza.

Q2-padawan (2.0 pts): Linguagens regulares

Escreva uma expressão regular que represente corretamente numeros inteiros com as seguintes características.

- Aceita somente números sem sinal como 42, mas não +42 ou -42.
- Não aceita a string vazia.
- Números grandes utilizam o underscore como separador do milhar: 1_234_567.
- Separador sempre agrupa grupos de 3.

Utilize o símbolo "d" para representar qualquer dígito válido, ou seja, assuma que d = 0 | 1 | 2 | ... | 9.

Q3-jedi (3.0 pts) Análize de código

Considere o código Python abaixo:

```python
def say_hi(x):
    print("Hello " + x)
```

a) Faça a análize léxica do código Python abaixo identificando cada token e associando um tipo com nome descritivo para cada elemento encontrado.

b) Proponha uma árvore sintática para representar o código acima. A árvore proposta não precisa refletir exatamente a árvore utilizada pelo Python, nem ser capaz de representar código Python arbitrário. Atribua nomes claros para cada tipo de nó na árvore e utilize de forma consistente os tokens propostos na primeira parte da questão.

Q3-padawan (2.0 pts) Análize de código

Responda às mesmas perguntas (a) e (b) da questão Q3-jedi utilizando o código abaixo:

```
x = 10 * b + 42
```

Q4-jedi (3.5 pts) Gramática para o JSON

Assumindo a existência dos valores terminais do tipo NUMBER e STRING, crie uma gramática BNF para representar todos os documentos JSON válidos. JSON (Javascript object notation) é um formato de listas e dicionários aninhados que podem conter valores de tipos básicos de Javascript. Segue um exemplo abaixo:

```
{
  "trues": [1.0, true],
  "falses": [0.0, false, null],
  "nested": {
    "objects": "sure!",
    "lists": "no problem!"
  }
}
```

Q4-padawan (2.5 pts) Gramática dos dicionários

Assumindo a existência dos terminais NUMBER e STRING, crie uma gramática BNF para representar dicionários válidos. Assuma que as chaves dos dicionários são strings e os valores podem ser números ou strings, como no exemplo:

```
{
  "nested": 0,
  "bools": "none",
}
```