Trabalho Final INF1022 2025.1

Profs. Vitor Pinheiro e Edward Hermann

25 de maio de 2025

1 Enunciado

O trabalho pode ser feito em dupla ou de forma individual. Neste trabalho deve ser desenvolvido um analisador sintático para a linguagem ObsAct. O analisador sintático deve ser capaz de compilar programas escritos utilizando a linguagem ObsAct para uma outra linguagem a sua escolha (como ilustrado na Figura 1). Ou seja, o analisador sintático recebe como entrada um programa na linguagem ObsAct e produz como saída um outro programa escrito em uma outra linguagem. A linguagem do programa de saída pode ser escolhida por você, ela pode ser qualquer linguagem a sua escolha. Por exemplo, mas não restrita a essas: C, C++, Java, Lua, Pyhton ou Assembly.

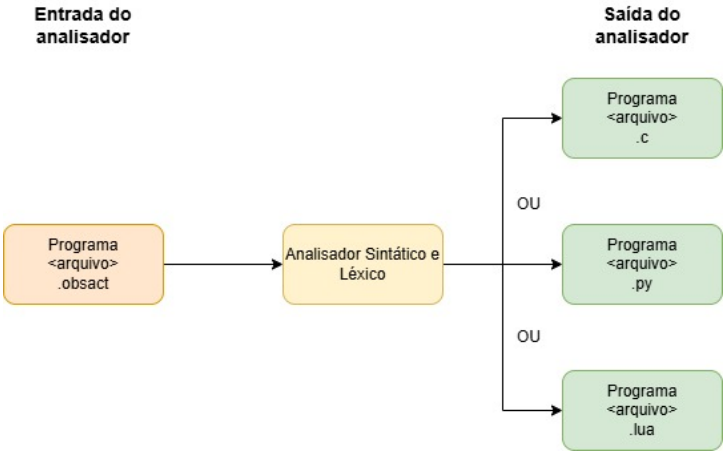

Figura 1: Analisador sintático para a linguagem ObsAct.

Para a implementação do analisador sintático, deve-se usar um gerador de analisador sintático que implemente o método LaLR(1) ou outro ascendente, por exemplo, Yacc/Lex,Bison/Flex (usa LaLR(1)), JavaCC (que usa o descendente LL(1)), etc.

1.1 Desenvolvimento

A sintaxe da linguagem *ObsAct* é dada pela gramática abaixo:

$$
\begin{array}{lcl}
PROGRAM & \longrightarrow & DEVICES\ \ CMDS \\
DEVICES & \longrightarrow & DEVICE\ \ DEVICES \mid DEVICE \\
DEVICE & \longrightarrow & dispositivo : \{namedevice\} \\
DEVICE & \longrightarrow & dispositivo : \{namedevice, observation\} \\
CMDS & \longrightarrow & CMD.\ \ CMDS \mid CMD. \\
CMD & \longrightarrow & ATTRIB \mid OBSACT \mid ACT \\
ATTRIB & \longrightarrow & set\ \ observation\ \ =\ \ VAR \\
OBSACT & \longrightarrow & se\ \ OBS\ \ entao\ \ ACT \\
OBSACT & \longrightarrow & se\ \ OBS\ \ entao\ \ ACT\ \ senao\ \ ACT \\
OBS & \longrightarrow & observation\ \ oplogic\ \ VAR \\
OBS & \longrightarrow & observation\ \ oplogic\ \ VAR\ \ \&\&\ \ OBS \\
VAR & \longrightarrow & num\ \ bool \\
ACT & \longrightarrow & ACTION\ \ namedevice \\
ACT & \longrightarrow & enviar\ \ alerta\ \ (msg)\ \ namedevice \\
ACT & \longrightarrow & enviar\ \ alerta\ \ (msg, observation)\ \ namedevice \\
ACTION & \longrightarrow & ligar\ \ desligar
\end{array}
$$

- Todas as variáveis númericas são do tipo inteiro e não negativo.
- Todas os terminais (por exemplo: *num*, *namedevice*, *namesensor* e *msg*) não podem ser vazios.
- *num* representa um número.
- *bool* representa TRUE ou FALSE.
- *observation* é uma string que representa o nome do sensor a ser observado.
- O terminal *namedevice* só pode conter letras.
- O terminal *namesensor* pode conter letras e números, porém só pode começar com letra.
- O terminal && é o operador lógico somente utilizado entre não terminais OBS. Significa que duas OBS precisam ser verdadeiras ao mesmo tempo. Por exemplo: se movimento == True && luz $<$ 100 entao ligar lampada.
- Perceba que o terminal "." sinaliza o fim dos comandos $ATTRIB$ e $OBSACT$.
- *oplogic*: é um terminal que representa um operador lógico. Eles podem ser: $>$, $<$, $>=$, $<=$, == ou ! =

A regra:

$$ACT \longrightarrow enviar\ \ alerta\ \ (msg, observation)\ \ namedevice$$

diz para enviar um alerta (msg, observation) para o device com nome *namedevice*. A mensagem de alerta contém duas strings que devem ser concatenadas: msg e observation. A concatenação resultante deve ser msg + (espaço em branco) + observation. Por exemplo, a linha de código:

```
enviar alerta ("Temperatura medida esta em", temperatura)
    Termometro .
```

vai concatenar a string "Temperatura medida está em"com a variável *temperatura* e vai resultar na string "Temperatura medida está em 37", sendo que 37 seria o valor da variável *temperatura*.

A gramática acima não esta completa, neste trabalho você vai precisar definir algumas regras a mais para incluir algumas funcionalidades. **Mais importante que isso, fique a vontade para alterar a gramática caso ache necessário para poder gerar a linguagem que esta sendo pedida.** Por exemplo, você pode retirar recursão a esquerda, fatorar ou até alterar as regras acima. Desde que a linguagem final gerada seja a mesma.

Além disso, a gramática descrita acima deve ser complementada ou corrigida para que a linguagem ObsAct seja capaz de executar comandos do tipo:

- Envio broadcast: será uma forma que o programador vai ter para enviar um alerta para um conjunto de dispositivos. Por exemplo:

```
    enviar alerta ("mensagem") para todos: namedevice,
        namedevice .
```

  Neste caso pode-se colocar quantos *namedevice* desejar e o alerta deve ser enviado para todos eles.

1.2 Exemplos de código na linguagem Obs

Alguns exemplos de programas em ObsAct são:

```
dispositivo: {Termometro, temperatura}
dispositivo: {ventilador, potencia}
set temperatura = 40.
set potencia = 90.
se temperatura > 30 entao ligar ventilador.
```

```
dispositivo {monitor}
dispositivo: {celular}
dispositivo: {Termometro, temperatura}
se temperatura > 30 entao .
enviar alerta ("Temperatura em ", temperatura) para todos:
    monitor, celular .
```

```
dispositivo: {celular, movimento}
dispositivo: {higr metro, umidade}
dispositivo: {lampada, potencia}
```

```
dispositivo: {Monitor}
set potencia = 100 .
se umidade < 40 entao enviar alerta "Ar seco detectado"
    Monitor .
se movimento == True entao ligar lampada senao desligar
    lampada .
```

```
dispositivo: {umidade}
dispositivo {Monitor}
se umidade < 40 entao enviar alerta "Ar seco detectado"
    Monitor .
```

```
dispositivo: {lampada, potencia}
set potencia = 100
ligar lampada
```

```
dispositivo: {ventilador}
desligar ventilador
```

```
dispositivo: {Celular}
enviar alerta ("Hora de acordar!") Celular
```

```
dispositivo: {Termometro, temperatura}
enviar alerta ("Temperatura esta em", temperatura)
    Termometro
```

1.3 Sobre os dispositivos

Veja que no inicio de todo programa ObsAct o programador precisa primeiro declarar todos os dispositivos que serão utilizados ao longo do programa. Como por exemplo:

```
dispositivo: {Termometro, temperatura}
dispositivo: {Ventilador, potencia}
dispositivo: {Celular}
dispositivo: {Monitor}
```

Estas regras estão em algum dos seguintes formatos:

$$\begin{aligned} DEVICE &\longrightarrow dispositivo : \{namedevice\} \\ DEVICE &\longrightarrow dispositivo : \{namedevice,\ \ observation\} \end{aligned}$$

Todo programa gerado a partir de um programa escrito em ObsAct vai conter 4 funções: uma para ligar o dispositivo, outra para desligar o dispositivo, outras duas para enviar um alerta para o dispositivo. Estas funções devem ser implementadas como a seguir:

```
function ligar(namedevice)
{
    print(namedevice+" ligado!")
}

function desligar(namedevice)
{
    print(namedevice+" desligado!")
}

function alerta(namedevice, msg)
{
    print(namedevice+" recebeu o alerta:\n")
    print(msg)
}

function alerta(namedevice, msg, var)
{
    print(namedevice+" recebeu o alerta:\n")
    print(msg + " "+ var)
}
```

Veja que estas 4 funções definidas acima estão em um pseudocódigo. *namedevice* é a variavel que identifica o dispositivo.

Elas devem ser implementadas na linguagem de programação escolhida pelo aluno. Por exemplo, caso o aluno queria traduzir a linguagem ObsAct para Python, as funções acima devem ser geradas em Python.

Desta maneira, toda vez que alguma ação de *ligar*, *desligar* ou *alerta* for executada para algum *namedevice*, são estas funções que devem ser chamadas passando como parametro o *namedevice*.

Para facilitar o trabalho, o aluno pode escolher também já deixar estas 4 funções implementadas no código. Por exemplo, se o aluno for gerar código em Python, ele já pode ter um arquivo .py que vai conter estas 4 funções (ligar, desligar e as duas de alerta).

As variaveis de *msg* e *namedevice* podem conter no máximo 100 caracteres.

1.4 Suposições

Todo valor de *observation* que não for definido pelo programador através da linha de código:

```
set temperatura = 40 .
```

deve ser definido para zero.

Perceba que a regra da gramática responsável por definir os valores dos sensores é:

$$ATTRIB \longrightarrow set \;\; observation \;\; = \;\; VAR.$$

1.5 Observação

Você tem a permissão de expandir a gramática para permitir comandos adicionais ou demais operações que julgar interessantes e/ou necessárias.

Deixe explícitas as alterações realizadas na gramática descrita neste documento (documente no relatório do trabalho). Tanto as alterações obrigatórias (solicitadas neste trabalho) quanto alterações que você julgar interessantes.

2 Entrega

Você deve entregar um arquivo contendo seu relatório, no qual estarão descritos seu trabalho - o que foi implementado, como foi implementado, o que funciona, o que não funciona, quais os testes utilizados etc. - e quaisquer outras informações que você considere relevantes, como a maneira de executá-lo.

No seu relatório final deve estar contido a gramática final utilizada no trabalho. A gramática deve estar descrita de maneira similar ao que a gramatica inicial foi descrita neste documento. Deixe claro quais regras foram adicionadas na gramática descrita neste relatório para que seja possível permitir todas as funcionalidades solicitadas. Caso tenha implementado alguma funcionalidade adicional, ou seja, uma que não foi solicitada, por favor deixe explícito.

Além disso, entregue um .zip com os casos de teste utilizados.

Por fim, entregue o código do seu analisador bem como quaisquer outros arquivos/módulos auxiliares que tenha criado para a execução do trabalho.

Indique nome e matrícula de **ambos os componentes da dupla**, se for o caso.

Caso tenha arguições, a ordem das arguições será definida com base na ordem das entregas no EAD.

Dica

- Comece pequeno: garanta que seu analisador funcione para um fragmento da linguagem, depois vá incrementando.