Robot-L: Compilador para Simulação de Robô Móvel[1]

Objetivo

Verificar e fortalecer os conhecimentos relacionados à técnica e à teoria para projeto e construção de compiladores, através do desenvolvimento de um compilador para uma linguagem de alto nível focada no controle de robô móvel.

Descrição

A avaliação consiste no desenvolvimento de um compilador capaz de realizar a transformação de um programa fonte, escrito usando uma linguagem de alto nível, para um programa em linguagem de montagem.

Cenário para uma linguagem de uso específico

A linguagem de alto nível foi especialmente projetada para um cenário muito específico, em que um robô móvel atua em um ambiente no qual precisa desviar de obstáculos e controlar a iluminação acendendo ou apagando lâmpadas.

Descrição geral do compilador

O compilador a ser construído deve possuir os seguintes blocos: analisador léxico, analisador sintático, analisador semântico, e gerador de código em linguagem de montagem.

Os blocos analisadores devem ser construídos seguindo os requisitos apresentados na

Tabela 1.

Tabela 1 - Requisitos mínimos para os blocos analisadores

Analisador	Descrição	Observação
Léxico	• Deve ler a sequência de caracteres que compõe o código fonte do programa, identificando-os e agrupando-os em uma seqüência de *tokens* válidos da linguagem. • Deve ser capaz de identificar e reportar os erros léxicos encontrados no código fonte (e.g. símbolos desconhecidos ou identificador mal-formado). Para cada erro encontrado, deve-se informar o posicionamento (linha e coluna) no arquivo fonte de entrada em que o erro ocorreu.	Não é permitido o uso de qualquer processador léxico (e.g. ***Lex***).
Sintático	• Deve ser capaz de agrupar, hierarquicamente, a sequência de *tokens* válidos em frases gramaticais e representá-la através de árvores semânticas. • Deve validar se as sentenças (frases gramaticais) estão de acordo com a gramática especificada para a linguagem. • Deve estar apto para identificar e reportar os erros sintáticos encontrados no código fonte. Para cada erro encontrado, deve-se informar o posicionamento (linha) no arquivo fonte de entrada em que o erro ocorreu.	Não é permitido o uso de qualquer processador sintático (e.g. ***Yacc***).
Semântico	• Uma vez que, a lexicografia e a sintaxe do código fonte estão corretas, deve ser capaz de avaliar se a semântica traduzida no programa segue a semântica especificada para a linguagem. • Deve estar apto para reportar todos os erros semânticos encontrados no arquivo fonte.	

[1] *Especificação adaptada dos Profs. Luciano Barreto e Alírio Sá.*

Descrição geral da linguagem Robot-L

A linguagem de programação Robot-L tem como propósito prover mecanismos simples para a manipulação de robôs móveis. Para tanto, a mesma é composta por uma sintaxe simples e declarações de alto nível. A especificação realizada é baseada na linguagem *Karel*[2] e é apresentada na Tabela 2.

- Os terminais estão descritos entre aspas duplas e em negrito.
- O símbolo * representa zero ou mais ocorrências do não-terminal à esquerda deste símbolo.
- Produções opcionais estão entre colchetes.
- O compilador deve aceitar palavras escritas em minúsculas ou maiúsculas.
- Comentários são definidos em linhas iniciadas com o símbolo "#"

Tabela 2 – Especificação da gramática da Robot-L

Token	Expressão Regular
Programa	::= **"programainicio"** Declaracao* **"execucaoinicio"** Comando **"fimexecucao" "fimprograma"**
Declaracao	::= **"definainstrucao"** identificador **"como"** Comando
Bloco	::= **"inicio"** Comando* **"fim"**
Comando	::= Bloco | Iteracao | Laco | Condicional | Instrução
Iteracao	::= **"repita"** Numero **"vezes"** Comando **"fimrepita"**
Laço	::= **"enquanto"** Condicao **"faca"** Comando **"fimpara"**
Condicional	::= **"se"** Condicao **"entao"** Comando **"fimse" ["senao"** Comando **"fimsenao"]**
Instrucao	::= **"mova"** Numero* **["passos"] | "Vire Para"** Sentido **|** Identificador **| "Pare" | "Finalize" | "Apague Lampada"| "Acenda Lampada" | "Aguarde Ate"** Condição
Condicao	::= **"Robo Pronto" | "Robo Ocupado" | "Robo Parado" | "Robo Movimentando" | "Frente Robo Bloqueada" | "Direita Robo Bloqueada" | "Esquerda Robo Bloqueada" | "Lampada Acessa a Frente" |"Lampada Apagada a Frente" | "Lampada Acessa A Esquerda" | "Lampada Apagada A Esquerda" | "Lampada Acessa A Direita" | "Lampada Apagada A Direita"**
Identificador	::= Letra(Letra|Digito)*
Numero	::= Digito*
Letra	::= **"A" | "a" | "B" | "b" | ... | "z"**
Digito	::= **"0" | ... | "9"**
Sentido	::= **"esquerda" | "direita"**

Programa exemplo

Usando a gramática apresentada na seção anterior é possível compor o programa apresentado no Exemplo 1. O objetivo desse programa é acender uma lâmpada supondo a existência/conhecimento do mapa do ambiente no qual se encontra o robô.

```
1   PROGRAMAINICIO
2     DEFINAINSTRUCAO Trilha COMO
3     INICIO
4           Mova 3 Passos
5           Aguarde Até Robô Pronto
6           Vire Para ESQUERDA
7           Apagar Lâmpada
8           Vire Para DIREITA
9           Mova 1 Passo
10          Aguarde Até Robô Pronto
11    FIM
12    EXECUCAOINICIO
13          Repita 3 VEZES Trilha
14          Vire Para DIREITA
15    FIMEXECUCAO
16  FIMPROGRAMA
```

Exemplo 1 - Exemplo de Programa Usando a Robot-L

[2] Disponível em: http://mormegil.wz.cz/prog/karel/prog_doc.htm.

Descrição das declarações

Maiores detalhes sobre as declarações da linguagem Robot-L podem ser encontrados na Tabela 3.

Tabela 3 - Detalhes sobre as declarações da linguagem

Declaração	Descrição	Exemplo
Programainicio, fimprograma	Delimitam o corpo (inicio e o final) de um programa. Pode conter declarações ou um bloco de execução.	Ver Exemplo 1.
Execucaoinicio, fimexecucao	Delimitam o corpo (inicio e o final) do bloco principal do programa. Pode conter uma ou mais instruções.	Ver Exemplo 1.
Definainstrucao <instnome> como <comando>	Define uma macro instrução. É similar a definição de um procedimento nas linguagens tradicionais, contudo não recebe qualquer tipo de parâmetro. Pode ser composto por instruções bases ou por outras macro instruções.	Ver Exemplo 1
Repita n VEZES <comando> fimrepita	Repete uma instrução ou bloco de instruções n vezes. Equivale a um laço incondicional em linguagens tradicionais.	Ver Exemplo 1
Enquanto <condicao> faca <comando> fimpara	Repete uma instrução ou um bloco de instruções enquanto a condição é verdadeira.	**Enquanto** Robo Pronto **Faça** ... **Fimpara**
se <condicao> entao <comando> fimse [senao <comando> fimsenao]	Estrutura condicional.	**SE** Robo Pronto **Entao** ...
Mova [n] ["passos"]	Coloca o robô em movimento. Quando declarado sem parâmetro, o robô será posto em movimento contínuo. O parâmetro n indica o número de passos a serem realizados pelo robô.	Ver Exemplo 1
Vire Para <sentido>	Faz com que o robô vire para um dos sentidos (DIREITA ou ESQUERDA)	Ver Exemplo 1
Pare	Faz com que o robô pare, se o mesmo estiver em movimento.	
Finalize	Aborta a execução do programa.	
Apague lampada	Aciona o atuador do robô para que este acenda uma lâmpada.	
Acenda lampada	Aciona o atuador do robô para que este apague uma lâmpada.	
Aguarde ate <condicao>	Bloqueia a execução do programa até a condição seja verdadeira.	

Regras semânticas

As regras semânticas definidas para a linguagem não permite que:

- Existam duas declarações de instrução com o mesmo nome;
- Declarações ***Vire Para*** imediatamente subseqüentes tenham sentidos diferentes. Exemplos:

Vire Para ESQUERDA (Permitido)
Vire Para ESQUERDA

Exemplo 2 – Comando Mova: Esquerda e Esquerda.

Vire Para DIREITA (Permitido)
Vire Para DIREITA

Exemplo 3 - Comando Mova: Direita e Direita.

Vire Para ESQUERDA (Permitido)
Pare
Vire Para DIREITA

Exemplo 4 – Comando Mova: Esquerda, Pare e Direita.

Vire Para ESQUERDA (Não Permitido)
Vire Para DIREITA

Exemplo 5 – Comando Mova: Esquerda e Direita.

Vire Para DIREITA (Não Permitido)
Vire Para ESQUERDA

Exemplo 6 – Comando Mova: Direita e Esquerda.

- Uma vez que o robô é composto por dispositivos mecânicos, algumas instruções precisam ser concluídas para que novas instruções possam ser executadas, assim:

Após uma instrução ***Mova*** n, em que n representa o número de passos, deve ser precedida por uma instrução do tipo ***Aguarde Até*** *Pronto*;

Geração de Código

Uma vez que o processamento léxico, sintático e semântico do código foi bem sucedido, o compilador deve traduzir o código fonte para um código intermediário em linguagem de montagem, seguindo o padrão Intel 8086.

O código *assembly* gerado pode ser executado usando o emulador de 8086, denominado emu8086. Nesse emulador, é considerada a possibilidade de comunicação com elementos externos através de portas de E/S (comandos in e out). O emulador suporta o *interfaceamento* com dispositivos virtuais que podem ser criados usando qualquer linguagem de programação que permita manipulação de arquivos.

O *interfaceamento* com tais dispositivos é feito usando um arquivo ($emu8086.io$) para comunicação com o dispositivo virtual. A porta 0 é representa pelo *byte* zero no arquivo, a porta 1 pelo *byte* um, a porta 2 pelo *byte* dois, e assim por diante. Através do arquivo podem-se endereçar portas de 00000 a $0FFFFh$ (0 a 65535).

No emu8086 existe a possibilidade de *interfaceamento* com um robô móvel (dispositivo virtual previamente disponível como exemplo no simulador).

O robô é controlado pelo envio de dados para a porta de E/S de número 9. Considerando os comandos da Tabela 4.

Tabela 4 - Lista de comandos para o robô móvel

Valor decimal	Valor binário	Ação
0	00000000	Não executa qualquer ação.
1	00000001	Movimenta para frente.
2	00000010	Vira para esquerda.
3	00000011	Vira para direita.
4	00000100	Examina um objeto usando o sensor. Quando a tarefa é finalizada, armazena o resultado no registrador de dados e um bit 1 no registrador de estado.
5	00000101	Acende uma lâmpada
6	00000110	Apaga uma lâmpada.

Um exemplo de transformação pode ser vista no Exemplo 7.

Mova	MOV AL, 1 ;MOVA PARA FRENTE OUT 9, AL
Vire Para Direita	MOV AL, 3 ;VIRAR PARA DIREITA OUT 9, AL
Mova	MOV AL, 1 ;MOVA PARA FRENTE OUT 9, AL
Vire Para Esquerda	MOV AL, 2 ;VIRAR PARA ESQUERDA OUT 9, AL
Mova	MOV AL, 1 ;MOVA PARA FRENTE OUT 9, AL

(a) Linguagem de Alto Nível **(b) Linguagem de montagem**

Exemplo 7 - Exemplo de transformação de código

As informações sobre a execução do comando "examinar" enviado para o robô pode ser obtido através do registrador de dados (porta 10). Como pode ser visto na Tabela 5.

Tabela 5 - Registrador de dados

Valor decimal	Valor binário	Significado
255	11111111	Parede à frente
0	00000000	Nada em frente
7	00000111	Lâmpada acessa a frente
8	00001000	Lâmpada apagada a frente
9	00001001	Lâmpada acessa a esquerda
10	00001010	Lâmpada apagada a esquerda
11	00001011	Lâmpada acessa a direita
12	00001100	Lâmpada apagada a direita
240	11110000	Parede à esquerda
15	00001111	Parede à direita

O registrador de estado atual do robô pode ser obtido usando o registrador de estado (porta 11), ver Tabela 6.

Tabela 6 - Registrador de estado

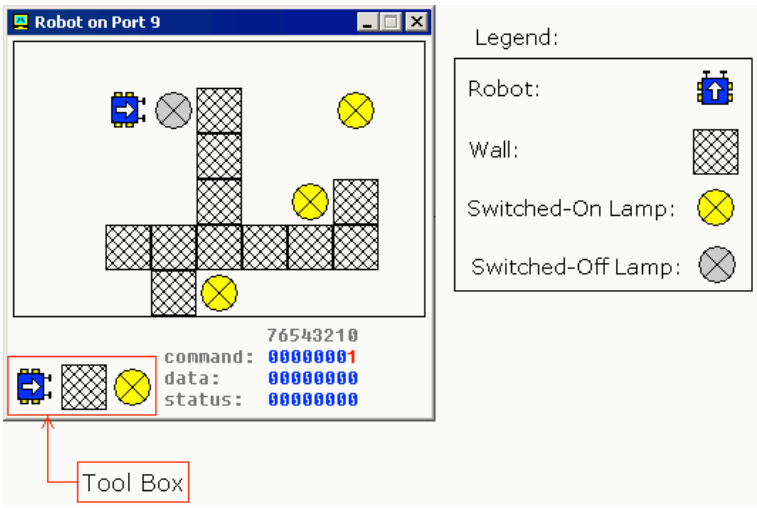
Número do bit	Descrição
$bit\,\#0$	Zero quando não existem novos dados no registrador de dados e um caso contrário.
$bit\,\#1$	Zero quando o robô está pronto para receber novos comandos e um caso contrário.
$bit\,\#2$	Zero quando não existem não ocorreram problemas na execução do último comando e um caso contrário.

Maiores detalhes sobre o funcionamento do emulador e da biblioteca de instruções 8086 disponível para uso podem ser encontrados no manual[3] de funcionamento que vem junto com o programa instalador do emu8086.

[3] http://jbwyatt.com/253/emu/io.html

Considerações

- Os detalhes da simulação de robô móvel podem ser encontrados na seção "*I/O ports and Hardware Interrupts*" do manual do emu8086.
- Um código exemplo para a manipulação do robô pode ser encontrado no arquivo *robot.asm*, na pasta *examples* do simulador.
- Imagem do robô no simulador pode ser vista na figura abaixo



Entrega e Pontuação

Entrega do Código Final + Artigo: 20/11/2019 (para todas as equipes).

Apresentações: nas aulas dos dias 25/11, 27/11 e 02/12 (ordem definida por sorteio no dia 20/11/2019).

Documentação: Todo código gerado deve estar devidamente documentado. Deve ser criado um Artigo em formato de artigos da SBC[4] relatando todo o projeto, as decisões tomadas pela equipe, e os trechos do código fonte que ilustrem as decisões. Devem ser incluídos exemplos dos erros considerados e como o compilador responde a cada tipo de erro.

Critérios de Avaliação

- considerará o uso das técnicas de projeto e desenvolvimento de compiladores vistos em sala de aula;
- Para cumprir com a avaliação considera-se a entrega dos produtos descritos na Tabela 7.
- **Atenção:** Programas que não compilam ou com cópias totais /parciais (plágio) terão nota **zero.**
- Entregas fora do prazo estabelecido terão desconto de 20% da nota por dia de atraso.
- As notas serão atribuídas de forma totalizada para todo o grupo, e o grupo decidirá como será feita a redistribuição interna. Assim, um grupo de 3 pessoas pode receber a nota 24, e decidir que a nota do Aluno 1 é 10; Aluno 2 é 8 e Aluno 3 é 6.

Tabela 7 - Produtos e Pontuação

Produto	Subproduto	Pontuação Máxima	Entrega
Analisadores (e correspondentes **tratamentos de erros**)	Analisador Léxico	1,0	30/09
	Analisador Sintático	1,5	04/11
	Analisador Semântico	1,5	04/11
	Gerador de Código	1,5	20/11
Documentação	Artigo + Código Fonte	1,5	20/11
Apresentação (Grupo + Individual)		3,0	25/11, 27/11 e 02/12 (sorteio)

[4] https://www.sbc.org.br/documentos-da-sbc/category/169-templates-para-artigos-e-capitulos-de-livros