Vol. 6 Núm. 6 Novembro 87

LINGUAGENS DE PROGRAMAÇÃO DE BRAÇOS ROBOT

Autores:
JOÃO JOSÉ DOS SANTOS SENTIEIRO
- Professor Responsável pela cadeira de Robótica no IST. Orientador científico da linha 6 do CAPS - CONTROLO.
RUBEN ANTÓNIO VARÃO COSTA
JOÃO FERNANDO CARDOSO SILVA SEQUEIRA
PAULO JORGE COELHO RAMALHO OLIVEIRA
- Alunos do mestrado de Robótica e Controlo no IST. Bolseiros da JNICT.

INTRODUÇÃO

Com a existência de uma tão vasta gama de aplicações e de forma dos robots, será muito difícil uma verdadeira linguagem de programação de robots (LPR) universal. Uma LPR deve conter um conjunto de instruções que permitam uma manipulação eficiente do robot tal qual as linguagens standart permitem uma manipulação eficiente do hardware do computador.

Numa LPR as instruções devem ser de tal modo que permitam uma manipulação do robot abstraindo-se da existência de motores, sensores, etc. instruções que permitem a deslocação do manipulador para determinado ponto, ou que se faça uma soldadura através de comandos simples.

A existência de uma LPR universal implicava uma infinidade de instruções visto existir uma quase infinidade de instruções possíveis (Go, Solda, Pinta, Aperta, Aparafusa, etc), e com certeza cada robot só poderia utilizar uma pequena fracção de todas essas instruções (não faria sentido um robot pintor ter uma instrução de aparafusa).

Assim os esforços produzidos para a uniformização das linguagens standart que tando contribuiram para uma manipulação fácil e eficiente dos robots, não serão válidos no caso das LPRs. E então necessário munir os robots com linguagens (ou apenas instruções) específicas para a sua forma ou aplicação. Essas "linguagens" variam desde o fornecimento pelos fabricantes de um conjunto de rotinas que serão adicionadas a uma linguagem standart ou mesmo uma mutação de uma linguagem standart até linguagens novas.

O que irá ser aqui mostrado e que o FORTH será por ventura uma linguagem ideal para este tipo de aplicações, visto ser uma linguagem óptima para a criação de novas linguagens.

AS LINGUAGENS DE PROGRAMAÇÃO DE ROBOTS

As LPR podem ser divididas em 3 grandes grupos

I - Linguagens ponto-a-ponto.

A linguagens deste tipo é duvidoso chamar-se linguagens de programação visto que elas apenas permitem a recolha de uma série de pontos posições das várias juntas), e que posteriormente serão reproduzidas (linguagens de programação e muito mais). Estas linguagens funcionam como um gravador de cassetes: existe uma fase de recolha (gravação) e depois pode-se reproduzir andar para traz para a frente, extrair pontos do percurso, acrescentar pontos.

Estas linguagens não têm qualquer semelhança com as linguagens de programação de autómatos programáveis. Enquanto estas últimas têm primitivas do tipo IF... THEN... ELSE o que permite que online seja decidido qual o ramo do programa que irá ser executado (faz sentido falar em ramos dos programas) neste tipo de LPR não existe a possibilidade de escrever programas. Apenas primitivas de sincronização tipo WAIT (por ex. um robot poderá ter que esperar por um sinal de READY de um sensor para começar a passar pelos pontos previstos ou a dado passo do percurso, ter que esperar por um sinal para continuar, mas isto só em linguagens ponto-a-ponto mais evoluídas).

Este método de manipular robots tem a grande vantagem de ser de simples utilização (por exemplo os vários pontos podem ser adquiridos através de um pendente), mas só tem aplicação em ambientes de trabalho bastante estruturados (nada corre mal) e além disso não permitem uma tomada de decisão online.

II - Linguagens de programação explícita.

A este tipo de linguagem já se pode chamar linguagem de programação. São na maioria dos casos baseados nas linguagens de programação convencionais. Tem tipos de instruções específicas para a manipulação de robots e permitem uma programação avançada recorrendo as primitivas de controlo de alto nível bem como o recurso a rotinas. Neste caso já se consegue trabalhar em ambientes pouco precisos, visto que podem interactuar com sensores avançados e o programa pode prever incertezas e corrigi-las.

A grande desvantagem deste tipo de linguagens é que para a sua manipulação é necessário saber programar o que se pode tornar complicado visto que o operador de um robot será por excelência alguém ligado à Produção.

III - Linguagem tipo TASK

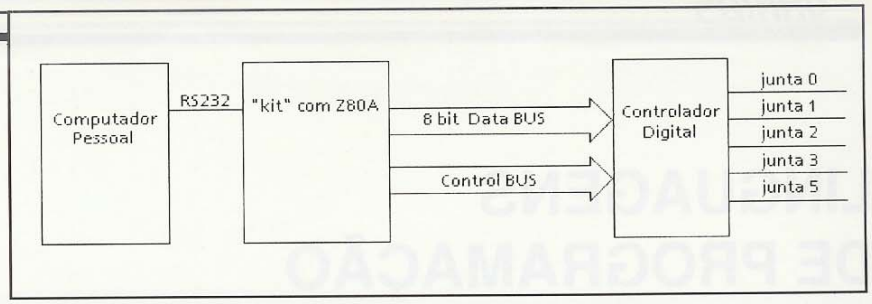
Este tipo de linguagem permite ao programador estar a um nível bastante elevado em relação ao robot. O programador não necessita de lidar com imprecisões, obstáculos, falhas, etc.. A programação é feita ao nível de TASK. É mandado fazer algo ao robot e o robot descobre a maneira de o fazer estando o programador por cima disto.

Estas linguagens têm todas as vantagens dos tipos anteriores ou seja é de fácil programação e trabalham em ambientes imprecisos.

No entanto necessitam de recursos computacionais bastante grandes como sejam mini-computadores sistemas de visão, bases de dados, CAD, sistemas periciais, etc.

Ainda não existem linguagens de programação deste tipo num estado final, no entanto já se obtiveram resultados satisfatórios neste domínio (em Portugal neste momento existe um grupo de trabalho neste domínio na Universidade Nova).

Após uma breve exposição sobre o estado actual dos vários tipos de LPR vamos ver em seguida como com o FORTH se pode criar uma LPR do tipo I, II e talvez III (?).



(Fig. 1) - Arqitectura do sistema

O FORTH

Antes de apresentar o FORTH devemos dizer que as aplicações privilegiadas do FORTH são no campo do controlo em tempo real, o que implica para já que o FORTH seja uma linguagem rápida.

A maioria das linguagens para trabalhar em tempo real são linguagens compiladas. Estas muito mais rápidas que as linguagens interpretadas tem como desvantagem o não permitirem uma interactividade com o exterior. O factor interactividade é muito importante em aplicações de robótica visto que por um lado pode haver a necessidade de dar comandos directos ao robot, por outro lado permite uma detecção de erros. Por isto a maioria das LPR são interpretadas sendo nalguns casos fornecido adicionalmente um compilador. O FORTH sendo uma linguagem compilada, apesar de a compilação neste caso não significar uma tradução de FORTH para código máquina, mas sim para um código intermédio que será interpretado por um interpretador FORTH (note-se que este interpretador não interpretará o código escrito pelo programador), permite que mal uma WORD (o conceito de WORD é semelhante ao conceito de rotina nas linguagens comuns) seja definida possa ser executada, e como possue para além de um editor de texto primitivas para manipulação de disco o FORTH poderá quase ser considerado um sistema operativo.

APLICAÇÕES DO FORTH

Como já foi dito anteriormente a grande parte das aplicações do FORTH são aplicações em tempo real. isto deve-se ao facto de o FORTH ser rápido e necessitar de pouca memória. Assim eis algumas das aplicações:

- Na astronomia nomeadamente no controlo de periféricos.
- Na medicina de vigilância a doentes quer nos hospitais quer em sistemas portáteis.
- Em controlo de periféricos de computador (CRT, impressoras, floppy disk drive, etc.).
- Na produção de máquinas de jogos tipo mete moeda.
- No controlo de câmaras de cinema para obtenção de efeitos especiais.
- Em satélites.

CONCEITOS FUNDAMENTAIS

1 - STACK.

O stack é o lugar privilegiado para a passagem de parâmetros entre as várias rotinas. O FORTH utiliza uma notação POSFIXA ou seja primeiro escrevem-se os argumentos e em seguida a operação em causa. Isto quer dizer que antes de se envocar uma WORD se colocam os parâmetros na stack e quando a WORD for executada ela vai retirar os parâmetros a stack.

Ex: Numa linguagem convencional para escrever um número no ecrã faz-se WRITE (6) ou seja primeiro escreve-se a instrução e em seguida o seu argumento, em FORTH seria 6 em que o 6 seria colocado no stack e a WORD iria buscar um número ao stack e imprimi-lo no ecrã.

2 R.P.N.

A notação para toda a aritmética do FORTH e um caso particular do que foi dito atrás. Assim se quisermos fazer:
2*3 em FORTH 2 3*
(6+5) * (2+3) em FORTH será 65 + 23 +*

3 WORD

Uma WORD é o nome pela qual se evoca uma definição. Será o correspondente nas linguagens de programação a uma rotina. Em FORTH para correr o que nas linguagens convencionais será um programa e em FORTH evocar uma WORD.

4 DICIONÁRIO

Lista ligada que contém todas as definições até ao momento. O dicionário corresponde a sequência de IF que estão na secção anterior.

5 INTERPRETADOR

Parte do FORTH que quando de uma evocação se encarrega de procurar a WORD evocada no dicionário.

6 COMPILADOR

Parte do FORTH que se encarrega de juntar novas definições ao dicionário. No FORTH existem vários compiladores, visto uma variável ser colocada de maneira diferente no dicionário do que uma WORD executável. O seguinte exemplo mostra como é compilada uma variável e uma palavra executável:

3. Alguns aspectos da utilização do FORTH no controlo digital.

Conceptualmente, pode separar-se a tarefa de controlar um manipulador em 2 partes:

1. Controlo, digital ou analógico, de cada uma das juntas, utilizando hardware dedicado, ou de uso geral, combinado com software.

2. 2.1 Construção de um nível task, de fácil utilização, que permita o movimento do braço.

2.2. Construção de um nível de Inteligência Artificial utilizando a linguagem atrás criada.

No presente caso, o microcomputador e o controlador utilizam o mesmo hardware, baseado no microprocessador 280A.

A linguagem escolhida para construção do nível task foi o FORTH, dadas as suas capacidades de manipulação de I / O, a sua arquitectura "stack based" e o facto de ser interpretada. Esta última característica é necessária a arquitectura de conjunto escolhida. O FORTH é de instalação muito simples e permite a construção de primitivos tal que a sua utilização se torna bastante parecida com a linguagem comum.

O controlador digital pode ter várias formas (fig. 2).

Qualquer destas formas é facilmente implementada em FORTH. O PID ou o PD da fig. 2 podem ser descritos pelas equações as diferências:

PID $y(k) = y(k-1) + p^o.x(k) + p1.x(k-1) + p2.x(k-2)$
PD $y(k) = A.y(k-1) + B.y(k-2) + C.x(k) + D.x(k-1) + E.x(k-2)$

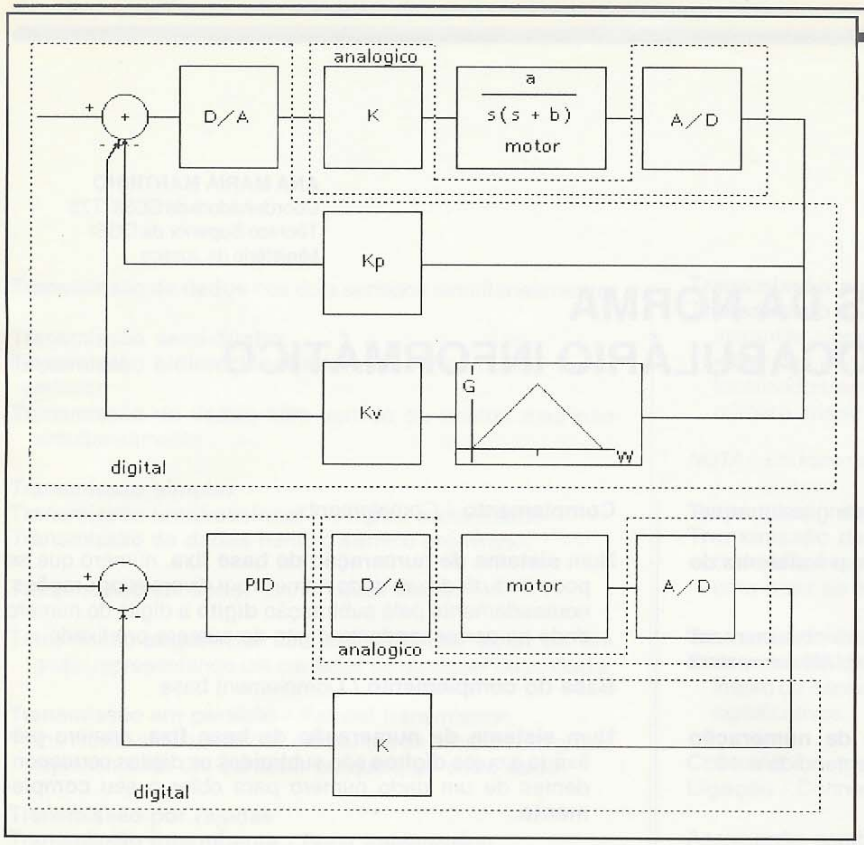
Utilizando apenas o stack, chega-se a formas muito simples para os controladores.

```
PID
SWAP p2 * OVER pi *+ O P@ DUP p° * ROT
+ SWAR ROT ROT
DUP 5 P!
```

Onde p^o e p2 são os parâmetros do PID.

```
PD
DUP A * ROT B * + ROT DUP D * ROT +
4 ROLL E * + O P@
DUP C * ROT + 4 ROLL SWAP
```

Onde A, B, C, D, e E são parâmetros do PD.



(Fig. 2) - Algumas formas de controladores

O port tem o endereço O e a organização do stack, para os dois casos é:

PD	PID
y (k-1)	(k-1)
y (k-2)	x (k-1)
y (k-1)	x (k-2)
x (k-2)	

LINGUAGEM TIPO TASK EM FORTH

Pretende-se que um braço robot transporte uma série de caixas entre várias paletes (uma palete é um lugar físico onde podem estar caixas. Por exemplo, o terminal de um tapete rolante pode ser considerado uma palete desde que seja possível e desejável que o robot aí carregue ou descarregue caixas). Para isso dispôs-se as paletes numa coroa circular em redor do braço, de tal forma que para aceder a qualquer palete o braço apenas tem que movimentar duas juntas: uma, para se orientar em relação à palete e a outra, para tocar as caixas note-se que o braço possue um electroíman na ponta. No entanto, é indiferente o tipo de efector que o braço possue).

Assim, em FORTH foram criadas as seguintes WORDS de alto nível:

AGARRA	(agarra a carga na palete em que o braço posicionado)
LARGA	(coloca a carga que o braço transporta na palete onde o braço está posicionado
pi pd MOVE	(desloca a carga da palete pi para a palete pd)
pn GO	(coloca-se sobre a palete pn)

Ao movimentar-se por intermédido destas acções, o braço evita qualquer obstáculo que se lhe depare, e minimiza a altura a que deve subir, ou seja: vê qual a caixa mais alta que encontra no caminho, soma a essa altura uma margem de segurança e sobe até a altura calculada.

Com estas WORDS podem-se construir "programas" (em FORTH, tal como em LISP, não é muito correcto falar em programas) para os mais variados fins. Suponhamos que a estação de trabalho consiste em três paletes

- Palete 1 - terminal de um tapete rolante
- Palete 2 - zona de processamento (por ex., uma máquina de solda
- Palete 3 - o início de outro tapete rolante.

O processo de fabrico constituirá em colocar a peça que vem na palete 1 para a zona de processamento e, após esta estar processada, colocar na palete 3 a peça que irá seguir para a próxima estação de processamento. Supõe-se ainda que existem dispositivos externos que indicam quando uma peça chega a palete, quando a peça está processada e quando a peça abandona a pelete. Então, o "programa" para controlar esta estação de trabalho seria o seguinte:

```
: CICLO
BEGIN (Espera até chegar    pe.a

ONTERMINAL?
UNTIL
1 2 MOVE
BEGIN
ENDCICLE?    (Espera até fim)

UNTIL
2 3 MOVE
1 GO;
```

Note-se a simplicidade do programa" desenvolvido. Como o FORTH não distingue as WORDS de base das WORDS criadas, com o braço robot poderá ser fornecido um FORTH que contenha as primitivas básicas para a sua manipulação, sendo os "programas" específicos construidos consoante as necessidades das aplicações utilizando essas primitivas.

O NÍVEL SUPERIOR - PLANEAMENTO E INTELIGÊNCIA

No nível superior teremos então uma linguagem, que como foi dito no início não obrigará o utilizador a ter um conhecimento formal do robot. e neste nível que se está a investir mais presentemente mas não nos podemos esquecer que é aquele que necessita de recursos computacionais maiores como sejam minicomputadores sitemas de visão, bases de dados, CAD, sistemas periciais, inteligência artificial, etc.

Para resolução do problema de célula de trabalho atrás citada, poderemos ter um programa em LISP, que vai gerar o programa em FORTH a partir das posições iniciais e finais da célula. Note-se que é o LISP que vai funcionar a um nível de inteligência artificial - planeamento.

Para que seja possível teremos de formalizar a célula de trabalho. Tal foi feito recorrendo ao mundo de blocos, onde teremos que cada peça e um bloco e as posições possíveis para o tratamento das peças são as paletes.

Neste nível atingimos assim para este problema muito simples a total independência entre o planeamento (LISP) e a execução (FORTH).