PROGRAMAÇÃO LÓGICA NA ELABORAÇÃO DE DIAGRAMAS UNIFILARES

Alexandre Cardoso
Universidade Federal de Uberlândia
Departamento de Eng. Elétrica
38400 Uberlândia - Minas Gerais

Julio Cesar Portela Silveira
Universidade Federal de Uberlândia
Departamento de Engenharia Elétrica
384000 Uberlândia - Minas Gerais

Sérgio de Mello Schneider
Universidade Federal de Uberlândia
Departamento de Informática
38400 Uberlândia - Minas Gerais

RESUMO

Este trabalho apresenta uma utilização de linguagem lógica (PROLOG) em computação gráfica. O PROLOG foi utilizado na elaboração de rotinas gráficas que permitem mostrar o diagrama unifilar de sistemas elétricos industriais em tela de monitores de equipamentos de controle em tempo real.

1. INTRODUÇÃO

Em âmbito industrial, os diagramas unifilares são de suma importância para visualização global do sistema de energia elétrica [03]. Atualmente, com implantação de sistemas informatizados em tais setores faz-se necessário a implementação de utilitários com recursos de computação gráfica no sentido de melhorar a interpretação e gerenciamento do sistema como um todo.

Para a elaboração de diagramas unifilares é necessário criar uma simbologia específica tais como símbolos de barramentos, transformadores, capacitores, indutores e geradores que compõem o

alfabeto básico. Tais símbolos têm significado como componentes físicos existentes na rede de energia elétrica. A convenção seguida na concepção gráfica (a nível de implementação) segue normas específicas da ABNT (Associação Brasileira de Normas Técnicas).

Optou-se por aplicar os recursos de linguagem lógica nos aplicativos de computação gráfica por permitir uma grande redução no tamanho do programa computacional e no número de instruções de busca nos arquivos de informações. A utilização de linguagem de progamação lógica (PROLOG) difere de implementações em linguagens procedurais (C,PASCAL,etc), onde a programação é feita por sequência precisa de instruções.

2. PROGAMAÇÃO LÓGICA E ELABORAÇÃO DE DIAGRAMAS UNIFILARES

A linguagem de progamação PROLOG é largamente aceita para propósitos de computação simbólica [04]. Um programa escrito em PROLOG é uma associação de fatos e regras que levam a uma dedução lógica da solução do problema. O progamador faz uma descrição do problema através de cláusulas e a execução se encarrega de encontrar a melhor solução

No problema em questão, é necessário que se faça uma abstração do comportamento de um desenhista na elaboração do "layout" do sistema elétrico industrial. O auxílio dado pela linguagem PROLOG é exatamente na abstração das soluções e opções desejáveis durante elaboração do diagrama unifilar. Neste aspecto, encontramos diversas opções e possibilidades como:

a) a seleção da direção da pena ao desenhar;

b)o tipo do equipamento (e suas dimensões) que será desenhado;

c)a disposição do símbolo de cada equipamento dentro do diagrama unifilar;

d)a necessidade de descontinuidade do desenho a nível de implementação.

Tais problemas são solucionados através de claúsulas que são opções para o operador. Cada claúsula elaborada está associada a fatos que são considerados como verdade em âmbito local.

Um exemplo da facilidade de implementação é um caso dos possíveis movimentos da pena na mão do desenhista. O predicado implementado é o predicado **"move"** com suas diversas opções:

```
move(desce,X,Yv,X,Yn):-!,Yn=Yv+1,putpixel(X,Yv,63).
move(sobe,X,Yv,X,Yn):-!,Yn=Yv-1,putpixel(X,Yv,63).
move(esquerda,Xv,Y,Xn,Y):-!,Xn=Xv-1,putpixel(Xv,Y,63).
move(direita,Xv,Y,Xn,Y):-!,Xn=Xv+1,putpixel(Xv,Y,63).
move(f1,X,Y,X,Y):-dbarr(X,Y,20).
move(f2,X,Y,X,Y):-dbarr(X,Y,30).
move(f3,X,Y,X,Y):-dbarr(X,Y,40).
move(f4,X,Y,X1,Y):-X1=X+40,dtrafo(X,Y,X1,Y).
move(f5,X,Y,X,Y1):-Y1=Y+20,dtrafo(X,Y,X,Y1).
move(f6,X,Y,X1,Y):-X1=X+20,dcapacitor(X,Y,X1,Y).
```

```
move(f7,X,Y,X,Y1):-Y1=Y+20,dcapacitor(X,Y,X,Y1).
```

Ao chamar tal predicado, é utilizado uma clausula le_tecla que busca no teclado a tecla digitada pelo operador. A partir daí, a mesma ganha um nome em outra cláusula do programa. De posse deste nome, a linguagem de programação procura uma das opções do predicado "move" e a executa. Observa-se a completa ausência dos desvios que apareceriam numa implementação em linguagem procedural(aumentando a quantidade de linhas elaboradas , o tempo de programação e a ilegibilidade do programa).

Na implementação, os predicados que elaboram o desenho dos elementos do diagrama unifilar são baseados em primitivas gráficas que compõem o recurso gráfico da linguagem. Por exemplo, o predicado que desenha o esboço horizontal de um transformador seria:

```
dtrafo(X,Y,X1,Y):- !,dpontos(D,X,Y,X1,Y),
                    X2 = X + 2 * D / 5,
                    R1 = D / 5,
                    X3 = X + 3 * D / 5,
                    X4 = X1 + D / 5,
                    X5 = X + 4 * D / 5,
                    circle(X2,Y,R1),
                    circle(X3,Y,R1),
                    line(X,Y,X4,Y),
                    line(X5,Y,X1,Y).
```

A conclusão de que o esboço do transformador é horizontal ou

vertical é feita no momento em que há passagem de parâmetros. A própria linguagem, através das claúsulas elaboradas, observa se valores X ou Y se mantêm constantes concluindo se o elemento deve ser desenhado em posição horizontal ou vertical.

Observamos aqui mais uma vantagem das linguagens lógicas, uma vez que numa linguagem procedural seria necessário "ensinar" como optar entre um elemento e outro.

3. GERAÇÃO DE ARQUIVOS

É necessário, durante a construção do diagrama unifilar, arquivar todos os passos descritos no esboço. Para tal, contamos com recursos de PROLOG de criação, manuseio e manutenção de arquivos. Este arquivo conterá todas informações de elementos do diagrama unifilar.

A criação desta base de dados é feita passo a passo, armazenando cada linha inserida pelo operador. Para tal fim, criaram-se predicados que identificam o tipo, as coordenadas e as demais informações do elemento do sistema, armazenando-os sequencialmente em estrutura de árvore, cada ramo representando uma espécime :

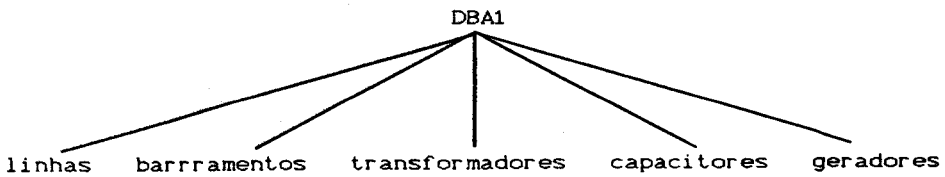

Figura 1 - Estrutura do arquivo de dados .....

vertical é feita no momento em que há passagem de parâmetros. A própria linguagem, através das claúsulas elaboradas, observa se valores X ou Y se mantêm constantes concluindo se o elemento deve ser desenhado em posição horizontal ou vertical.

Observamos aqui mais uma vantagem das linguagens lógicas, uma vez que numa linguagem procedural seria necessário "ensinar" como optar entre um elemento e outro.

3. GERAÇÃO DE ARQUIVOS

É necessário, durante a construção do diagrama unifilar, arquivar todos os passos descritos no esboço. Para tal, contamos com recursos de PROLOG de criação, manuseio e manutenção de arquivos. Este arquivo conterá todas informações de elementos do diagrama unifilar.

A criação desta base de dados é feita passo a passo, armazenando cada linha inserida pelo operador. Para tal fim, criaram-se predicados que identificam o tipo, as coordenadas e as demais informações do elemento do sistema, armazenando-os sequencialmente em estrutura de árvore, cada ramo representando uma espécime :

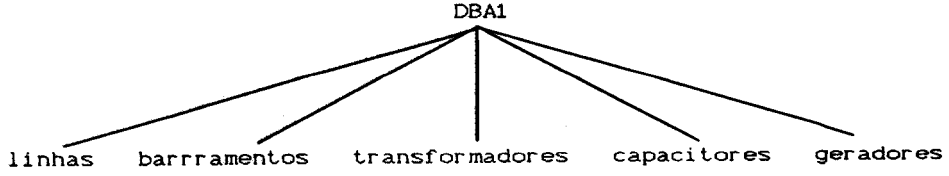

Figura 1 - Estrutura do arquivo de dados .....

Um dos predicados implementados tem a forma:

```
salva(f3,X,Y,X1,Y):-!, ilinhas(X,Y,X1,Y),
                        entradados(barr,N),
                        chain_insertz(dba1,ramo2,barrams,barr(X,Y,40),
                        db_close(dba1),
                        numera(N,"f3").
```

4. OPERAÇÕES GRÁFICAS ELABORADAS

Uma das necessidades, quando da visualização de uma rede de energia elétrica em monitor é a aproximação da imagem de forma a observar detalhes de determinados trechos da rede. Para solucionar tal problema, implementou-se em PROLOG rotinas de visualização como o "ZOOM". Para tanto, novamente, elaboraram-se predicados que produzem o efeito desejado, mediante consulta ao arquivo. Por exemplo:

```
zoom:-
 retractall(coord(_,_,_),coords),
 retractall(ponto(_,_),ponto),
 retractall(pvirtual(_,_),pvirtual),
 assertz(pvirtual(0,0),pvirtual),
 assertz(ponto(1,1,outras,ponto),
 seta(X,Y),
 X1 = X + 5,
 Y1 = Y + 5,
 quadrar(X,Y,X1,Y1,B),
```

```
telazoom(X,Y,X1,Y1,B),
esc,
cleardevice,
setviewport(0,0,639,199),
retanova,
trafonovo,
descapc,
desbarr,!.
```

Cada claúsula leva a uma parte do processo de promover o zoom com os elementos que são selecionados na claúsula "quadrar", que estabelece os limites da tela a ser transfomada numa nova tela através das transformações sobre os elementos. Uma das transformações é feita sobre as retas, conforme:

```
retanova:-
   chain_first(dba1,ramo1,T),
   ref_term(dba1,linhas,P,linha(X,Y,_,_,V)),
   buscalinha(T,V,X,Y),!.

buscalinha(T,V,X,Y):-
   chain_next(dba1,T,P),
   ref_term(dba1,linhas,P,linha(X,Y,_,_,V1)),
   tracar(X,Y,X1,Y1,V),
   buscalinha(P,V1,X1,Y1).
buscalinha(_,_,_,_).

tracar(X,Y,X1,Y1,"sim"):-
```

```
escala(Dx,DY),
getmaxx(Xmax),
getmaxy(Ymax),
Xn=X*Xmax,
X1n=X1*Xmax,
Yn=Y*Ymax,
pvirtual(Xv,_),
coord(Xt,_,Xb,_),
Xn > (Xt+Xv),
Xn < (Xt+Xv),
desenhareta(Xn,Yn,Xn,Y1n,Dx,Dy).
```

Este trecho demonstra as transformações operadas sobre as retas que se encontram na área dada pelas coordenadas de "**quadrar**" de acordo com a escala dada pela divisão destas coordenadas por pontos máximos X e Y. Observa-se grande facilidade na descrição de processos como claúsulas, além da facilidade em manusear os dados contidos no arquivo de elementos.

Implementou-se ainda rotinas de SCROOL, permitindo maior flexibilidade ao desenhar ,criando um ambiente de uma tela de comprimento infinito, onde a tela corrente corresponde a um trecho do esboço total

5. CONCLUSÃO

A linguagem de programação lógica PROLOG apresenta certas facilidades no caso de aplicativos gráficos. conforme pudemos observar para o trabalho de elaboração de diagramas unifilares,

inclusive com operações de transformação e geração de arquivos.

Os recursos gráficos utilizados fazem parte da linguagem Turbo Prolog 2.0, podendo ser implementados em qualquer micro compatível com IBM-PC.

REFERÊNCIAS

[1]. R.Fujiwara and Y.Kohno ; "*User-Friendly Workstation for power systems analysys* " , IEEE vol PAS-104, No 6, June 1985

[2].R. P. Schulte, G. B. Sheble, S. L. Larsen and J. N. Wrubel; "*Artificial inteligence solutions to power system operating problems*" , IEEE PWRS-2, No 4, November 1987

[3].Dejan J. Sobjic and Yoj-Han Pao ; "*An artificial intelligence system for power system contygency screening*", IEEE Transations on Power System, Vol. 3, No 2, May 1988

[4]. Lamounier, Edgard A. Jr ; Hess Lilia A. e Schneidder Sérgio M. ; "Programação Lógica na Modelagem de Sólidos ", Anais do SIBCGRAPI'89

AGRADECIMENTOS

Agradecimentos ao professor Edgard Afonso Lamounier Jr pelo auxílio prestado na elaboração deste artigo e incentivo para este trabalho.