

Tutorial de X_Y-pic

Carlos A. P. Campani
campani@ufpel.edu.br

14 de abril de 2006

1 Introdução

X_Y-pic é um pacote para tipografar gráficos e diagramas em T_EX. O pacote X_Y-pic pode ser usado com T_EX e L^AT_EX e permite desenhar diversos tipos diferentes de gráficos e diagramas, incluindo polígonos, nós e diagramas em matriz. Ele é implementado em torno de um *kernel de linguagem gráfica*, que fornece uma notação mnemônica e consistente, baseada na composição lógica de componentes visuais.

Este tutorial tem o objetivo de ser uma introdução breve e acessível ao uso do X_Y-pic. Estamos longe de pretender apresentar todos os recursos disponíveis. Ele complementa o *X_Y-pic Reference Manual*, de Kristoffer H. Rose e Ross Moore [2], e o *X_Y-pic User's Guide*, de Kristoffer H. Rose [1], que podem ser obtidos em <http://www.tug.org/applications/Xy-pic/>. Ainda há o ótimo livro *The L^AT_EX Graphics Companion*, de Goossens, Rahtz, e Mittelbach [3]. Recomendamos a leitura de todos estes textos para aqueles que desejam usar intensamente o X_Y-pic.

Para carregar o pacote X_Y-pic no T_EX usam-se os comandos `\input xy` e `\xyoption{all}`, que carrega todos os recursos, o que pode tornar a execução do T_EX lenta. Para aumentar o desempenho do T_EX recomenda-se carregar apenas os recursos que serão usados. Da mesma forma que no T_EX, para carregar o X_Y-pic no L^AT_EX usa-se o comando `\usepackage[all]{xy}` no cabeçalho do arquivo.

Caso se queira produzir gráficos de nós e arcos, deve-se incluir adicionalmente as opções `knot` e `arc`. Também precisamos declarar as opções `import` e `poly` se quisermos importar imagens *postscript* e desenhar polígonos nos

diagramas. Para isto basta usar, no cabeçalho do arquivo \LaTeX , o comando `\usepackage[all,knot,arc,import,poly]{xy}`.

Problemas podem ocorrer devido a flexibilidade do formato de entrada do \TeX . Isto causa algumas situações complicadas de conflito. Um exemplo é o uso do XY -pic junto com o pacote *babel*, em português e outras línguas que redefinem as aspas, como é o caso também do alemão, o que povoca conflito quando se deseja salvar posições em um diagrama. Este problema pode ser resolvido ao usarmos `$$ \shorthandoff{"} \xy ... \endxy $$`, protegendo os comandos XY -pic definidos dentro de `\xy ... \endxy`.

Outro problema é o conflito do caracter `&` quando usado em um diagrama do XY -pic dentro de um ambiente *tabular*. Neste caso, é possível resolver o problema protegendo os comandos XY -pic dentro de um par `{ e }`.

2 Conceitos Básicos

A estrutura geral de uma XY -figura é `\xy ... \endxy`, que constrói uma caixa (*box*) com uma XY -figura (usuários \LaTeX podem substituir este comando por `\begin{xy} ... \end{xy}`). Nesta estrutura podem ser declarados comandos da “linguagem gráfica” do XY -pic.

Não há necessidade de colocar a XY -figura explicitamente em modo matemático, pois a declaração `\xy ... \endxy` já o faz. Caso haja necessidade de apresentar texto dentro da XY -figura, basta usar o comando `\txt{ ... }`.

Os elementos que formam a linguagem do XY -pic são:

Posições Representam coordenadas de pontos dentro da *caixa* da XY -figura;

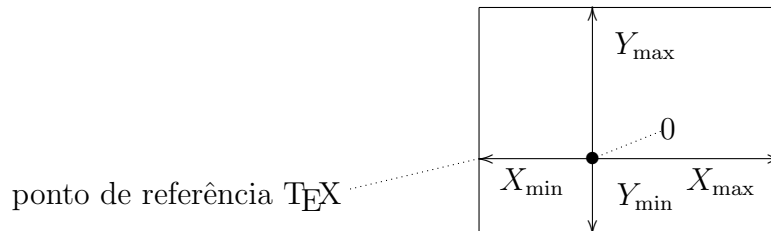
Objetos Um objeto é como uma caixa (*box*) do \TeX que pode ser posto em uma posição, exceto que ele possui uma *borda* (*edge*);

Conexões Junto com a capacidade de colocar objetos em posições, todos os objetos podem ser usados para conectar duas posições;

Decorações Sempre que o XY -pic encontra algo que não pode ser interpretado como uma posição, ele interpreta o que se segue como uma *decoração*, ou seja, um conjunto restrito de comandos a ser adicionado à figura.

Posições podem ser representadas por pares (x, y) , cujos valores x crescem da esquerda para a direita, e os valores y de baixo para cima. Assim,

a origem do sistema de coordenadas é o ponto $(0, 0)$ (também representado como 0), e a XY -figura está contida no seguinte retângulo:



A forma mais simples de colocar coisas em uma XY -figura é “largar” um objeto em uma posição. Para definir posições e “largar” objetos usa-se o operador $*$. Por exemplo,

```
(0,0)*{A}
```

coloca o rótulo A na posição $(0, 0)$.

Além de poder “largar” objetos em uma posição da XY -figura, podemos *conectar* os dois objetos correntes do *estado*, formado pelas posições p (posição prévia) e c (posição corrente). Para definir conexões usa-se o operador $**$. Assim,

```
\xy
(0,0)*{};(10,0)*{} **\dir{-}
\endxy
```

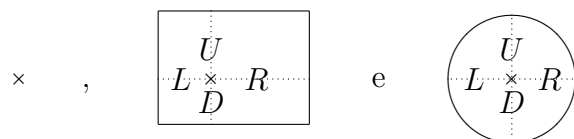
define a posição prévia $p = (0, 0)$ e a posição corrente $c = (10, 0)$ e conecta ambas com um *directional* definido por $**\dir{-}$. Qualquer objeto pode ser usado como conector e, neste caso, foi usado o $-$, indicando que as duas posições devem ser conectadas por uma linha simples. Observe que $(0,0)*\{\}$ e $(10,0)*\{\}$ define as posições prévia e corrente, sem “largar” nenhum objeto nelas.

O operador $;$ indica que deve-se atualizar as posições prévia e corrente, trocando a corrente anterior pela prévia e fazendo da última posição inserida a nova corrente. Assim, em

```
\xy
(0,0)*{};(10,0)*{} **\dir{-};
(10,10)*{} **\dir{-}
\endxy
```

O primeiro ; define as posições $p = (0, 0)$ e $c = (10, 0)$. O segundo ; atribui a p o valor anterior de c , $(10, 0)$, e faz $c = (10, 10)$. Então são traçadas duas linhas, uma entre $(0, 0)$ e $(10, 0)$ e outra entre $(10, 0)$ e $(10, 10)$.

Objetos possuem uma *borda* (*edge*). Assim, um objeto pode ser entendido como uma *caixa* (*box*) T_EX, com uma *forma* (*shape*), e com *dimensões* L , U , R e D . A forma do objeto força a forma de sua borda. O kernel do X_Y-pic fornece três formas (*shapes*), nomeadas `[.]`, `[]` e `[o]`, correspondendo a:



A forma (*shape*) *default* dos objetos é `[]`.

O X_Y-pic fornece um conjunto de *direccionais*, como no exemplo anterior o `**\dir{-}`. Os direcionais são elementos gráficos que podem ser tanto *conectores* quanto *pontas* (que terminam as extremidades de uma conexão).

Os conceitos apresentados de forma breve nesta seção serão melhor desenvolvidos nas próximas seções. Particularmente a seção seguinte tratará dos recursos do kernel do X_Y-pic, e mostrará por meio de exemplos o uso de posições, objetos e conexões.

3 Usando o Kernel do X_Y-pic

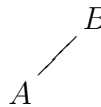
Nesta seção mostraremos o uso do kernel do X_Y-pic por meio de exemplos comentados. Serão introduzidos os recursos básicos disponíveis no kernel, e nas seções seguintes serão explorados aspectos mais avançados.

A coisa mais simples que podemos fazer com o X_Y-pic é definir duas posições e conecta-las. Isto é mostrado no exemplo seguinte, onde é produzida uma linha simples conectando as posições $(0, 0)$ e $(10, 0)$:

```
\xy
(0,0)*{};(10,0)*{} **\dir{-} _____
\endxy
```

Podemos também “largar” objetos nas posições. Isto é feito neste outro exemplo, em que definimos dois objetos com rótulos A e B , e traçamos uma linha na diagonal ligando estes dois rótulos:

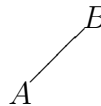
```
\xy
(0,0)*+{A};(10,10)*+{B} **\dir{-}
\endxy
```



O operador `*` é usado para definir posições e “largar” objetos, e o operador `**` é usado para definir conexões.

Neste último exemplo, observa-se o *modificador +* usado em $(0,0)*+{A}$ e $(10,10)*+{B}$. Este modificador serve para obter espaço adicional em torno do objeto, evitando que o conector fique muito próximo ao objeto, como seria o caso de:

```
\xy
(0,0)*{A};(10,10)*{B} **\dir{-}
\endxy
```



Podemos usar qualquer objeto como conector, como vemos no exemplo a seguir:

```
\xy
(0,0)*+{A};(10,10)*+{B} **\dir{>}
\endxy
```



Podemos definir três posições em seqüência e conecta-las. Como se explicou na seção anterior, o direcional `**\dir{-}` conecta as posições p e c do estado do Xy-pic . O operador `;` é usado para mudar o estado, trocando as posições p e c e atualizando a c . Mostramos isso no seguinte exemplo:

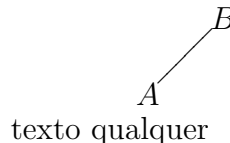
```
\xy
(0,0)*{};(10,0)*{} **\dir{-};
(10,10)*{} **\dir{-}
\endxy
```



Neste exemplo, a seqüência de mudanças de estado e ações do Xy-pic , associada aos comandos que as executam, é apresentada na Tabela 1.

Textos podem ser postos em uma Xy -figura usando o comando `\txt`:

```
\xy
(5,5)*{A};(15,15)*{B} **\dir{-};
(0,0)*{\txt{texto qualquer}}
\endxy
```

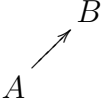


	Ação	Comando
1	$c \leftarrow (0, 0)$	<code>(0,0)*{}</code>
2	$p \leftarrow c$	<code>;</code>
3	$c \leftarrow (10, 0)$	<code>(10,0)*{}</code>
4	traça linha entre (0, 0) e (10, 0)	<code>**\dir{-}</code>
5	$p \leftarrow c$	<code>;</code>
6	$c \leftarrow (10, 10)$	<code>(10,10)*{}</code>
7	traça linha entre (10, 0) e (10, 10)	<code>**\dir{-}</code>

Tabela 1: Exemplo de execução do \Xy-pic

Este novo exemplo mostra o uso de conexões com *pontas*:

```
\xy
(0,0)*+{A};(10,10)*+{B} **\dir{-} ?>* \dir{>}
\endxy
```

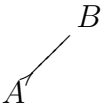


Direcionais podem ser do tipo *conectores* ou *pontas*. No exemplo dado, `**\dir{-}` é um *direcional conector*, e `\dir{>}` é um *direcional ponta*.

O `?>*` serve para indicar a posição da ponta no conector. O operador `?` serve para “pegar” o lugar da conexão mais recente definida por um `**`. O modificador `>` move posições, neste caso para o extremo final da conexão.

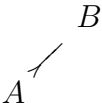
Poderíamos posicionar a ponta no outro extremo do conector usando `?<*`:

```
\xy
(0,0)*+{A};(10,10)*+{B} **\dir{-} ?<* \dir{>}
\endxy
```



Para melhorar o exemplo anterior poderíamos usar espaço adicional em torno dos objetos:

```
\xy
(0,0)+++{A};(10,10)+++{B} **\dir{-} ?<* \dir{>}
\endxy
```



Cada modificador `+` dobra o valor do espaço em torno de um objeto. Assim, ao usar, por exemplo, `(0,0)+++{A}` estamos introduzindo um espaço

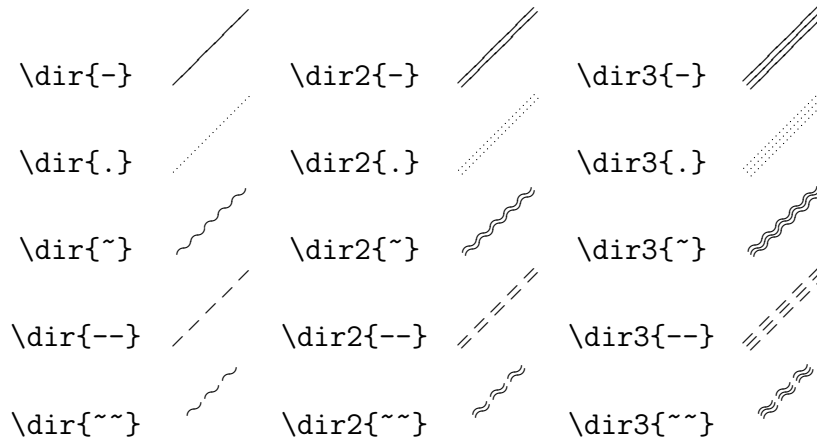



Figura 1: Direcionais (conectores)

4 vezes maior.

Agora podemos inverter a ponta, como fazemos no exemplo seguinte:


```
\xy
(0,0)*+{A};(10,10)*+{B} **\dir{-} ?<* \dir{<}
\endxy
```



As Figuras 1 e 2 apresentam os direcionais (conectores e pontas). Observe-se o recurso de fazer o conector duplo ou triplo por meio de `\dir2` e `\dir3`, e as variações de pontas usando-se `\dir~` ou `\dir_`.

Podemos produzir setas com o `Xy-pic`. Para isto usamos o comando `\ar`:

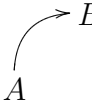
```
\xy
{\ar (0,0)*+{A}; (10,10)*+{B}}
\endxy
```



A Figura 3 apresenta as setas que podem ser usadas em uma `Xy`-figura. Devemos observar que `\ar@{=>}` e `\ar@{:>}` são abreviaturas de `\ar@2{->}` e `\ar@2{.>}`.

Podemos curvar uma seta, como por exemplo em:

```
\xy
{\ar@/^1pc/ (0,0)*+{A}; (10,10)*+{B}}
\endxy
```












































<code>\dir{>}</code>		<code>\dir^{>}</code>		<code>\dir_{>}</code>	
<code>\dir{<}</code>		<code>\dir^{<}</code>		<code>\dir_{<}</code>	
<code>\dir{ }</code>		<code>\dir^{ }</code>		<code>\dir_{ }</code>	
<code>\dir{(}</code>		<code>\dir^{(}</code>		<code>\dir_{(}</code>	
<code>\dir{)}</code>		<code>\dir^{)}</code>		<code>\dir_{)}</code>	
		<code>\dir^{'}</code>		<code>\dir_{'}</code>	
		<code>\dir^{'}</code>		<code>\dir_{'}</code>	
<code>\dir{>>}</code>		<code>\dir^{>>}</code>		<code>\dir_{>>}</code>	
<code>\dir{<<}</code>		<code>\dir^{<<}</code>		<code>\dir_{<<}</code>	
<code>\dir{ }</code>		<code>\dir^{ }</code>		<code>\dir_{ }</code>	
<code>\dir{ -}</code>		<code>\dir^{ -}</code>		<code>\dir_{ -}</code>	
<code>\dir{> }</code>		<code>\dir^{>> }</code>		<code>\dir_{ <}</code>	
<code>\dir{ <<}</code>		<code>\dir{*}</code>		<code>\dir{o}</code>	
<code>\dir{+}</code>		<code>\dir{x}</code>		<code>\dir{/}</code>	
<code>\dir{//}</code>					

Figura 2: Direcionais (pontas)

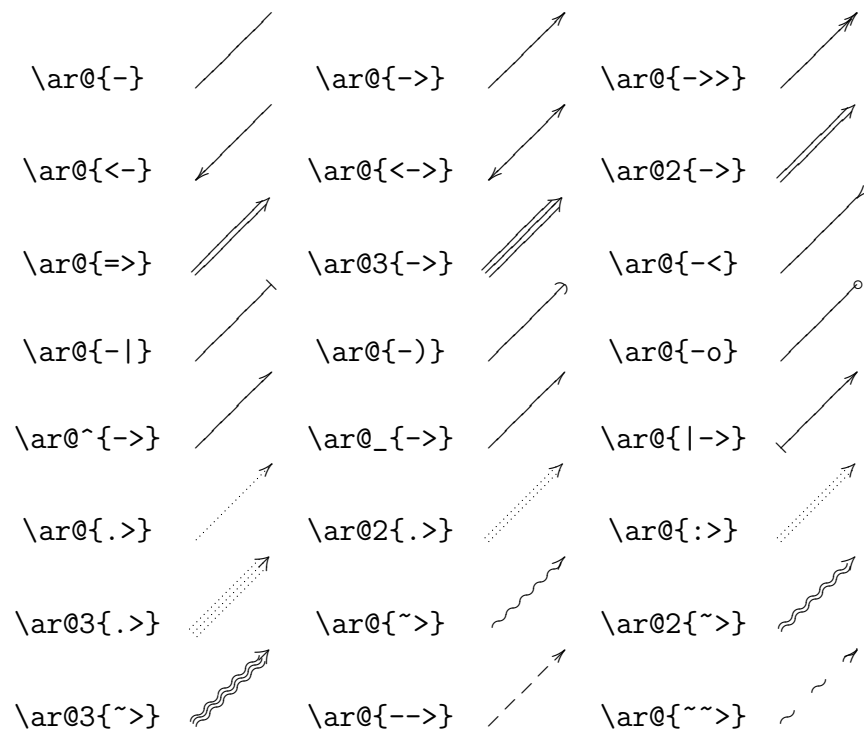
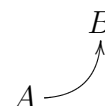


Figura 3: Setas

O `/^1pc/` é um *vetor*, normalmente usado para denotar um deslocamento, e que neste caso é usado para especificar a curvatura da seta. A curvatura especificada no exemplo é de `1pc=12pt` (pontos). Outras unidades de medida usadas pelo `TEX` são *ex* (correspondendo à altura da letra “x”), *mm*, *cm* e *in* (polegadas). O `^` indica a direção da curvatura (para cima).

Poderíamos curvar a seta para baixo, como em:

```
\xy
{\var@/_1pc/ (0,0)*+{A}; (10,10)*+{B}}
\endxy
```



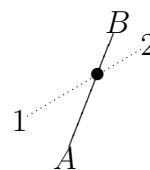
`XY-pic` fornece a facilidade de rótulos para indicar posições. Assim, com `(0,0)*{)="A"` podemos criar o rótulo A para indicar a posição (0,0). No seguinte exemplo definimos três rótulos para indicar posições dos vértices do triângulo que será traçado:

```
\xy
(0,0)*{)="A"; (10,0)*{)="B"; (10,10)*{)="C";
"A";"B" **\dir{-};
"A";"C" **\dir{-};
"B";"C" **\dir{-};
\endxy
```



A operação `!{pos1,pos2}` permite encontrar o ponto em que a última conexão intercepta uma linha definida pelas posições `pos1` e `pos2`. Por exemplo:

```
\xy
(0,5)*{1}="1"; (17,15)*{2}="2" **\dir{.};
(6,0)*{A}="A"; (13,18)*{B}="B" **\dir{-}
?!{1;2} *{\bullet}
\endxy
```

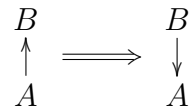


No comando `?!{1;2} *{\bullet}`, o operador `?` “pega” a posição da última conexão definida (neste caso a que liga as posições “A” e “B”), e a seguir a operação `!{1;2}` encontra o ponto de interceptação desta conexão com a linha que liga as posições “1” e “2”. Então é posto um `•` (“bullet”) no ponto de interceptação.

Podemos definir diagramas aninhados, ou seja, diagramas dentro de outros diagramas. Para isto basta usar um diagrama como se fosse um objeto

ou um direcional. Neste exemplo ilustramos o uso deste recurso:

```
\xy
(0,0)*++{
\xy
(0,0)*+{A}; (0,10)*+{B} **\dir{-} ?>* \dir{>}
\endxy
}="x";
(20,0)*++{
\xy
(0,0)*+{A}; (0,10)*+{B} **\dir{-} ?<* \dir{<}
\endxy
}="y";
{\ar@{=>} "x";"y"};
\endxy
```



Observe que

```
\xy
(0,0)*+{A}; (0,10)*+{B} **\dir{-} ?>* \dir{>}
\endxy
```

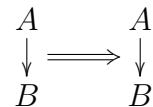
é atribuído a "x" e usado como objeto posicionado em (0,0) e

```
\xy
(0,0)*+{A}; (0,10)*+{B} **\dir{-} ?<* \dir{<}
\endxy
```

é atribuído a "y" e usado como objeto posicionado em (20,0). Ambos os objetos são conectados pela seta dupla definida por `{\ar@{=>} "x";"y"}`.

Podemos usar macros \TeX em Xy -figuras. Por exemplo:

```
\def\grafo{\xy (0,10)*+{A}; (0,0)*+{B} **\dir{-}
?>* \dir{>} \endxy}
\xy
{\ar@{=>} (0,0)*{\grafo};(15,0)*{\grafo}}
\endxy
```



Neste diagrama,

```
\def\grafo{\xy (0,10)*+{A}; (0,0)*+{B} **\dir{-}
?>* \dir{>} \endxy}
```

define uma macro $\text{T}_{\text{E}}\text{X}$, referenciada como `\grafo`, que é usada duas vezes aninhada no diagrama.

4 Extensões

Nesta seção são descritas algumas extensões ao kernel do Xy-pic . Apresentaremos curvas, círculos, frames e importação de gráficos externos.

Usando-se o comando `\crv` podemos criar curvas com múltiplos pontos tangentes:

```
\xy
(0,0)*{}="A";
(10,0)*{}="B";
"A"; "B" **\crv{(5,5)};
\endxy
```



Neste exemplo, a curva foi definida tendo apenas um ponto tangente, o $(5, 5)$.

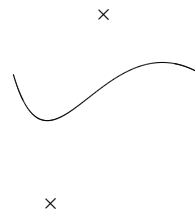
Podemos definir curvas com mais pontos tangentes:

```
\xy
(0,0)*{}="A";
(25,0)*{}="B";
"A"; "B" **\crv{(5,-17) & (12,8)};
\endxy
```



Uma facilidade para desenvolver curvas é tornar os pontos tangentes visíveis. Para isto usa-se `\pC`:

```
\xy
(0,0)*{}="A";
(25,0)*{}="B";
"A"; "B" **\crv~pC{(5,-17) & (12,8)};
\endxy
```



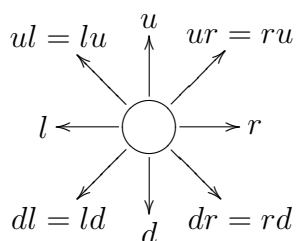
Para produzir círculos usamos o comando `\cir`. O tamanho *default* do círculo é o tamanho do objeto que ele envolverá. Este exemplo ilustra isto:

```
\xy
(0,0)*+{A}; (10,0)*+{B}* \cir{} **\dir{-}
\endxy
```

$A \text{---} \textcircled{B}$

Podemos especificar um raio para o círculo. Por exemplo, se o círculo deve ter 20pt de raio, usamos `\cir<20pt>{}`.

Segmentos de círculo podem ser obtidos especificando-se as direções dos vetores tangentes e um giro em sentido horário (usando `_`) ou sentido anti-horário (usando `^`). As direções que podem ser especificadas são:



Exemplos de segmentos de círculo:

```
\xy *\cir<5pt>{l^r} \endxy (
\xy *\cir<5pt>{dl_u} \endxy (
\xy *\cir<5pt>{dr^ur} \endxy )
\xy *\cir<5pt>{dr_ur} \endxy )
\xy *\cir<5pt>{ur^dr} \endxy )
```

Se são dadas a mesma diagonal duas vezes, então nada é produzido, como em `\xy *\cir<5pt>{d^d} \endxy`, que produz “ ”.

No próximo exemplo produziremos um “*smile*” usando círculo, vetores, e os operadores `?`, `_` e `!`:

```
\xy
(0,0)*{}; (4,0)*{} **\dir{} ? *_!/3pt/\dir{)}
*_!/7pt/\dir{:}; (2,2)* \cir<5pt>{};
\endxy
```

$\textcircled{\text{smile}}$

O operador `?`, que já explicamos anteriormente, serve para “pegar” a posição da última conexão. O operador `_` serve para girar um objeto 90° em sentido horário (para o sentido anti-horário usaríamos o operador `^`). O operador `!` serve para tornar a direção oblíqua ao direcional usado (*skew*). Finalmente, os vetores `/3pt/` e `/7pt/` servem para deslocar os objetos “)” e “:” sobre a direção.

Vamos explicar passo a passo a construção do smile do nosso exemplo.

Em primeiro lugar, usamos o operador `?` para “pegar” a posição do direcional “vazio” (*dummy*) que conecta $(0, 0)$ e $(4, 0)$. Sobre esta direção será posto o “)”. Ilustramos isto, mostrando o direcional vazio como uma linha pontilhada para melhor visualização:



Para produzirmos a boca do smile devemos girar o “)” em um ângulo de 90° em sentido horário, e para isto usamos o operador `_`, resultando em:



Usamos o operador `!` para indicar a direção oblíqua ao direcional (para que possamos depois deslocar). Ilustramos com uma seta pontilhada esta nova direção:



As mesmas operações são feitas sobre o “:”, que formará os olhos do smile.


Finalmente, deslocamos o “)” e o “:” na nova direção, usando os vetores `/3pt/` e `/7pt/`, respectivamente:



O círculo foi usado como “toque final” para completar o smile.

Poderíamos também desenhar um smile usando um segmento de círculo, como em:

```
\xy
(0,0)*{};(4,0)*{} **\dir{} ? *_!/7pt/\dir{:};
(2,2)*\cir<5pt>{};
(2,2)*\cir<3pt>{dr^ur};
\endxy
```



Observe que a boca, tendo sido feita com um segmento de círculo, resultou em um smile um pouco diferente ao do exemplo anterior.

Frames são molduras que podem ser postas em XY -figuras. Uma moldura (frame) é um objeto XY -pic na forma `\frm{ ... }`. Na Figura 4 são mostrados alguns tipos de molduras disponíveis (para mais veja o *X_Y-pic Reference Manual*).

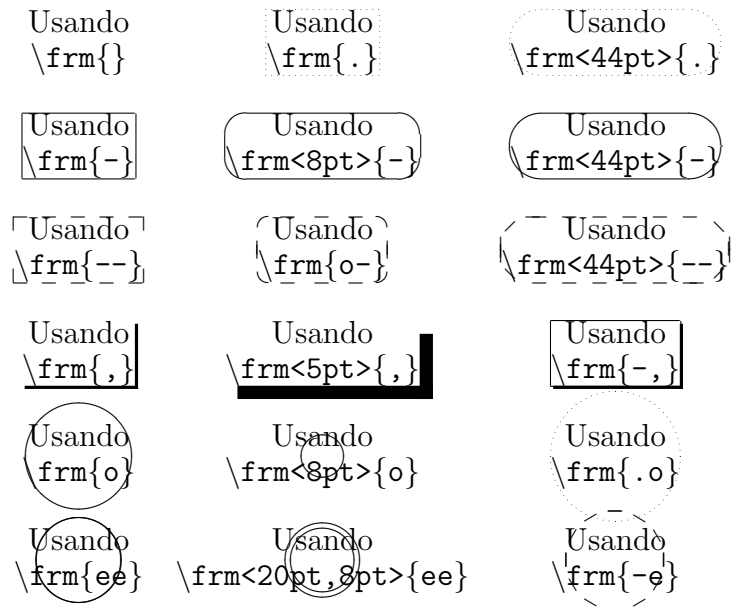
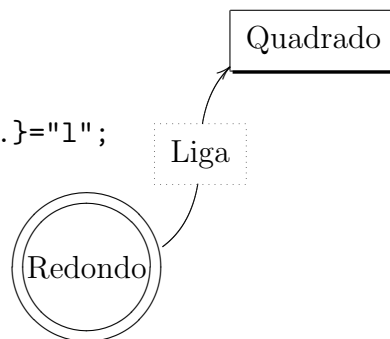


Figura 4: Molduras (frames)

Podemos agora, usando curvas e molduras (frames), construir o seguinte diagrama:

```
\xy
(0,0)+++{\txt{Redondo}}*\frm{oo}="r";
(30,30)+++{\txt{Quadrado}}*\frm{-,}="q";
"r";"q" **\dir{} ? +++{\txt{Liga}}*\frm{.}="l";
"r";"l" **\crv{(15,0)};
"l";"q" **\crv{(15,30)} ?>* \dir{>};
\endxy
```



Observe neste exemplo como o Liga é posicionado usando-se o operador ? para obter a posição da ligação entre "r" e "q".

Para importar imagens *postscript* devemos declarar a opção `import` na declaração `\usepackage[all,import]{xy}`. Podemos usar qualquer pacote para importar a imagem, como por exemplo, `graphicx`, `graphics`, `epsf` ou `epsfig`. Neste caso, usamos o `graphicx`, e para isto devemos declarar o uso

do pacote com o comando `\usepackage{graphicx}`.

Usamos `\xyimport` para estabelecer um sistema de coordenadas para uma imagem em particular, permitindo que qualquer comando do `Xy-pic` seja usado, com as posições relativas ao sistema de coordenadas definido. Para isto, o comando `\xyimport(larg,alt){imagem}` exige que se defina uma largura e uma altura, que fornece uma distância em unidades de coordenadas, iniciando no canto inferior esquerdo, onde o sistema de coordenadas usualmente deve estar localizado.

Assim, usando a imagem apresentada na Figura 5, podemos produzir o que se pode ver na Figura 6, usando o seguinte código:

```
\def\grafico{\includegraphics[width=9cm]{grafico.eps}}
\xy
\xyimport(100,100){\grafico}
{\ar (75,85)*+{\txt{Astróide}}; (60,75)*{}}
{\ar (75,25)*+{\txt{Ellipse}}; (80,37)*{}}
\endxy
```

Observe-se que

```
\def\grafico{\includegraphics[width=9cm]{grafico.eps}}
```

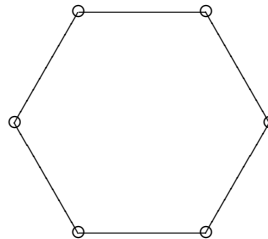
é uma macro `TeX` para definir a imagem a ser importada.

5 Usando Polígonos e Elipses

Para usar polígonos em diagramas é necessário carregar o `Xy-pic` declarando a opção `poly`, usando o comando `\usepackage[all,poly]{xy}`.

Polígonos podem ser produzidos usando-se o comando `\xypolygon`. Por exemplo, podemos criar um hexágono usando:

```
\xy
/r4pc/:{\xypolygon6{\circ}}
\endxy
```



Observe-se que `/r4pc/` especifica o tamanho do polígono em 4pc (48pt). O número de lados do polígono é declarado por meio de um valor inteiro após

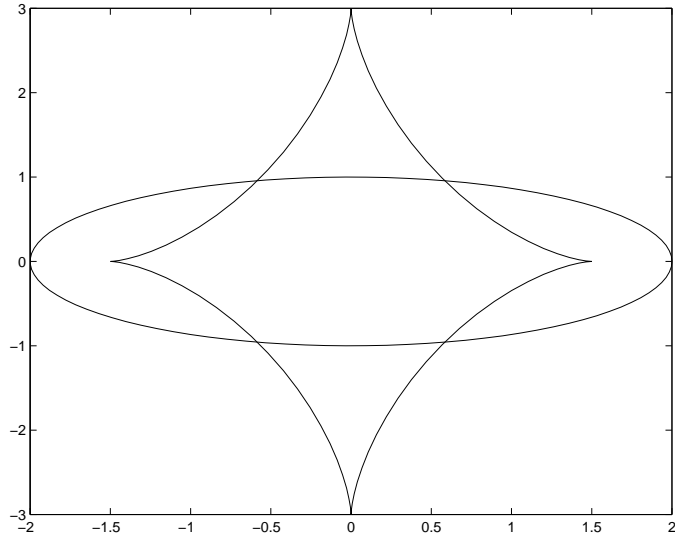


Figura 5: Imagem sem comandos X\&Y

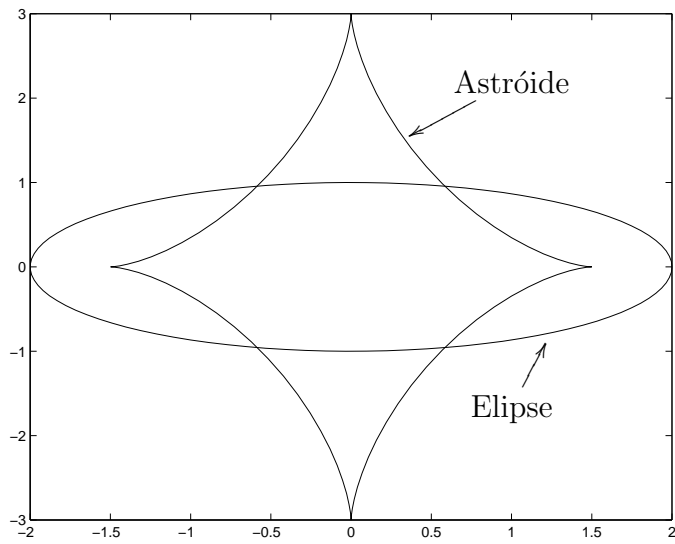
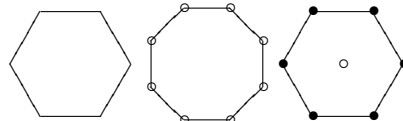


Figura 6: Importando uma imagem

o comando `\xypolygon`. Assim, especificamos o hexágono com `\xypolygon6`. Além disto, o argumento `\circ` indica que os vértices do polígono deverão ser círculos.

Alguns outros exemplos de polígonos são:

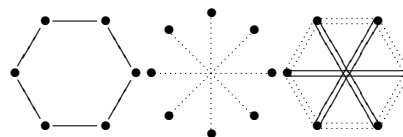
```
\xy /r8mm/
, 0 ,{\xypolygon6{}}
,+/r18mm/,{\xypolygon8{@{o}}}
,+/r18mm/,{*{@{o}}\xypolygon6{@{*}}}
\endxy
```



Nestes exemplos, 0 (origem) e `+/r18mm/` (deslocamento) especificam o posicionamento dos três polígonos, e os `@{o}` e `@{*}` especificam os vértices.

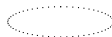
Mais exemplos:

```
\xy /r8mm/
, 0 ,{\xypolygon6{~*{\dir{*}}}}
,+/r18mm/
{\xypolygon8{~<{.}~>{}}~={45}{\dir{*}}}
,+/r18mm/
{\xypolygon6{~<{=}~>{:}{\dir{*}}}}
\endxy
```



Neste exemplo, observamos o uso de `~<`, `~>` e `~=` para indicar conexões entre os vértices do polígono.

Para produzir elipses usamos `xycircle`. Assim, `(0,0)*\xycircle(7,2){.}` produz uma elipse centrada em (0,0), com largura 7 e altura 2, e pontilhada:



6 Produzindo Diagramas em Matriz

O `Xy-pic` oferece uma facilidade (*feature*) para tipografar diagramas em forma de matriz. Este tipo de diagrama tem aplicação em diversas áreas da matemática e de ciência da computação, como por exemplo em teoria dos autômatos e teoria das categorias. Para produzir um diagrama deste tipo usamos o comando `\xymatrix{ ... }`.

O diagrama será formado pelas entradas de uma matriz, organizadas em linhas e colunas. Cada entrada pode conter uma expressão matemática (produzida usando o modo matemático).

Usa-se `&` para separar as colunas e `\\` para indicar nova linha, em uma notação semelhante ao ambiente `array` do modo matemático do \LaTeX . Assim, se desejamos produzir uma matriz com duas linhas e duas colunas, usamos:

```
\xymatrix{
1 & 2 \\
3 & 4
}
```

Poderíamos omitir entradas à direita que não fossem necessárias no diagrama, como em:

```
\xymatrix{
1 & 2 \\
3
}
```

Também podemos deixar entradas em branco na matriz, como em:

```
\xymatrix{
1 & 2 \\
& 4 \\
}
```

Para conectar entradas por setas usamos `\ar`. O destino da seta é definido de forma relativa à origem por meio de uma seqüência de `u` (acima), `d` (abaixo), `l` (esquerda) e `r` (direita), colocados entre colchetes. Assim, para conectar a entrada da primeira linha e coluna com a da segunda linha e coluna usamos `\ar[dr]`:

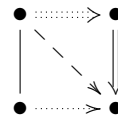
```
\xymatrix{
1 \ar[dr] & 2 \\
3 & 4
}
```

As setas da Figura 3 funcionarão também com o `\xymatrix`:

```

\matrix{
\bullet \ar@{:>}[r] \ar@{-->}[dr] & \bullet \\
\ar@{=>}[d] & \\
\bullet \ar@{-}[u] \ar@{.>}[r] & \bullet \\
}

```

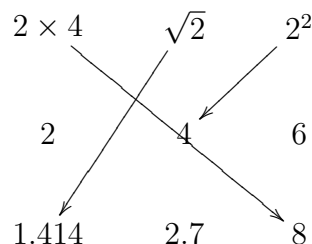


Este é um exemplo com uma matriz três por três:

```

\matrix{
2\times 4 \ar[ddrr] & \sqrt{2} \ar[ddl] \\
& 2^2 \\
2 & 4 & 6 \\
1.414 & 2.7 & 8 \\
}

```

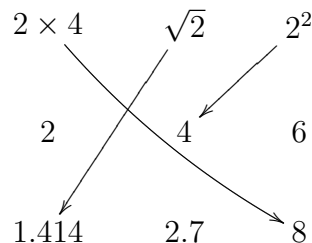


Observe que a seta que liga 2×4 e 8 passa sobre o 4 , o que pode ser inconveniente. Para evitar isto podemos curvar a seta para cima, usando “@/^/”, ou para baixo, usando “@/_/”. Neste caso, curvaremos para baixo:

```

\matrix{
2\times 4 \ar@/_/[ddrr] & \sqrt{2} \ar[ddl] \\
& 2^2 \\
2 & 4 & 6 \\
1.414 & 2.7 & 8 \\
}

```



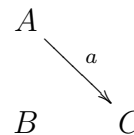
Para uma curvatura maior poderíamos usar, por exemplo, @/_1pc/.

Podemos colocar um rótulo acima (ou abaixo) de uma seta. Para isto basta usar “^” (ou “_”). Neste exemplo mostramos isto:

```

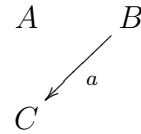
\matrix{
A \ar[dr]^a \\
B & C \\
}

```



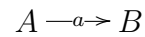
Observe que “acima” pode não significar exatamente acima da seta, se a seta está voltada para a esquerda:

```
\xymatrix{
A & B\ar[dl]^a \\
C
}
```



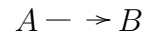
Também podemos posicionar o rótulo da seta sobre a seta, ou no “meio”, usando |:

```
\xymatrix{
A\ar[r]|a & B
}
```



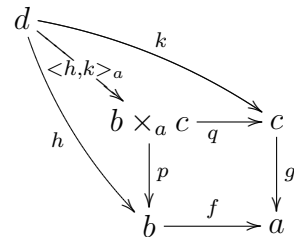
O “|” pode ser útil para fazer “buracos” nas setas (por exemplo, para passar outras setas sem que se cruzem). Para isto usamos \hole:

```
\xymatrix{
A\ar[r]|\hole & B
}
```



O seguinte diagrama é a definição de produto fibrado em teoria das categorias. Nele usamos vários dos recursos do \xymatrix já apresentados:

```
\xymatrix{
d\ar@/_/[ddr]_h\ar[dr]|{\langle h,k \rangle_a} \\
\ar@/^/[drr]^k \\
& \{b\times_a c\}\ar[d]^p\ar[r]_q & c \\
c\ar[d]^g \\
& b\ar[r]^f & a
}
```



O comando \xymatrix permite especificar a forma com que o diagrama será tipografado. A especificação é uma seqüência de @<especificação> que antecedem os comandos dentro do \xymatrix. Assim, por exemplo, \xymatrix@1{ ... } especifica que o diagrama deve ser tipografado em uma linha, como em \xymatrix@1{A\ar[r]^f & B}, que produz $A \xrightarrow{f} B$. Isto é útil para produzir pequenos diagramas que aparecerão dentro do parágrafo do texto.

Da mesma forma podemos modificar o espaçamento das linhas e das colunas por meio das especificações @R<dim> e/ou @C<dim>, como por exemplo

em `\xymatrix@R10pt@C5pt{ ... }`, que especifica 10pt para o espaçamento das linhas e 5pt para o das colunas.

Podemos explicitamente posicionar o rótulo sobre a seta:

`\xymatrix{A\ar[r]^<{f} & B}` $A \xrightarrow{f} B$

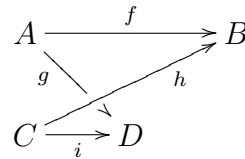
`\xymatrix{A\ar[r]^>{f} & B}` $A \xrightarrow{f} B$

`\xymatrix{A\ar[r]^(.4){f} & B}` $A \xrightarrow{f} B$

Observe-se que, no último caso, podemos usar um valor entre 0 e 1 como fator para posicionar o rótulo (foi usado 0,4 como exemplo). O fator 0 representa o início da seta, e o fator 1 representa o fim.

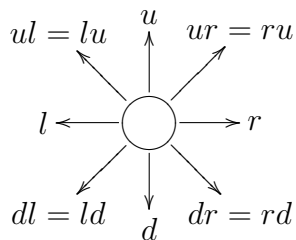
Finalmente, outra possibilidade é usar `!{t1;t2}`, que posiciona o rótulo no ponto em que a seta cruza a linha que liga os lugares `t1` e `t2`:

`\xymatrix{A \ar[rr]^f \ar[dr]_(.3)g \!{[d];[rr]}\hole & B \\ & \\ C \ar[rru]_(.7)h \ar[r]_i & D}`



Neste último exemplo, o `!{[d];[rr]}` determina o ponto em que a seta que liga `A` e `D` se cruza com a que liga `C` e `B`. Neste ponto é posto o `\hole`.

Podemos indicar a posição que a seta deve entrar ou sair de uma entrada usando as seguintes direções, que já haviam sido mostradas, e repetimos aqui para facilitar:



Isto nos permite fazer uma seta “reflexiva”, especificando a seta usando o comando `\ar@(saída,entrada) []`. O `[]` indica que a seta apontará para a própria entrada, e a especificação `@(saída,entrada)` define as direções de saída e entrada da seta. Assim,

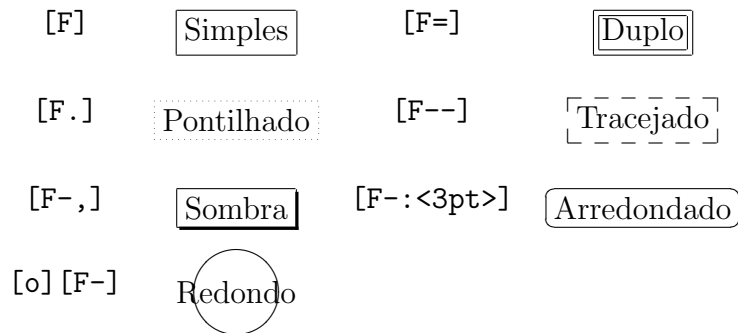
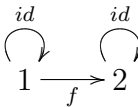


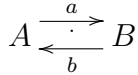
Figura 7: Frames em diagramas em matriz

```
\xymatrix{
1 \ar@{ul,ur}[]^{\text{id}} \ar[r]_f &
2 \ar@{ul,ur}[]^{\text{id}}
}
```



Podemos produzir setas paralelas, usando uma *dimensão* para separá-las, definida por @<dim>:

```
\xymatrix{
A \ar@{<1ex>[r]}^a \ar@{<1ex>[l]}_b B
}
```

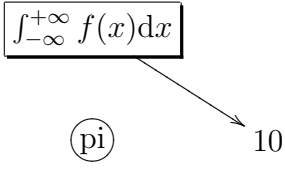


A dimensão de 1ex, adotada neste caso, é conveniente pois corresponde à altura da letra “x”.

Os recursos de frames (molduras) que podem ser usados nos objetos do diagrama são apresentados na Figura 7. Os modificadores + e - podem ser usados para aumentar ou diminuir o tamanho da moldura.

Um exemplo usando frames é:

```
\xymatrix{
*+[F-,]{\int_{-\infty}^{+\infty} f(x)\mathrm{d}x} \ar[r] &
*+[o][F-]{\text{pi}} & 10
}
```

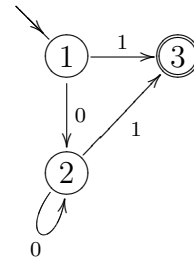


O exemplo a seguir é um diagrama que representa um autômato finito:

```

\matrix{
*+[o][F-]{1} \ar@{ul}{u}[] \ar[r]^{1}
\ar[d]^{0} & *+[o][F=]{3} \\
*+[o][F-]{2} \ar[ur]_{1} \ar@(dl,d)[]_{0}}

```

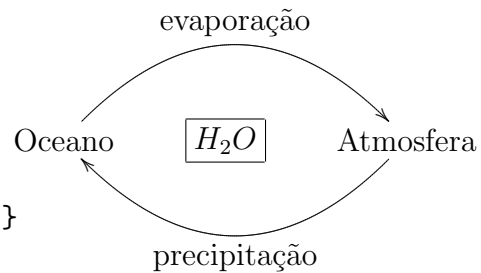


Outro exemplo:

```

\matrix{
{\text{Oceano}}
\ar@/^3pc/[rr]^{evaporação}
& *+[F-]{H_2O}
& {\text{Atmosfera}}
\ar@/_3pc/[ll]_{precipitação}
}

```

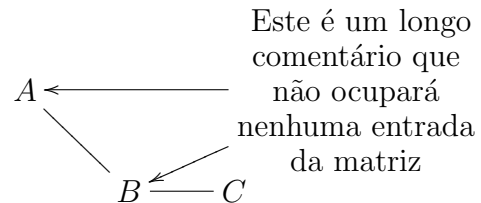


Ainda é possível colocar entradas extras, que estarão fora da matriz, usando o comando `\save ... \restore`. Neste caso, o que fica dentro do comando não fará parte de nenhuma entrada da matriz:

```

\matrix{
A \ar@{-}[dr] &
\save[]+<3cm,0cm>*\text<8pc>{
Este é um longo comentário que
não ocupará nenhuma entrada
da matriz}
\ar[l]\ar[d]
\restore \\
& B\ar@{-}[r] & C
}

```



Observe-se que a seta `\ar[d]`, que parte do comentário, não necessariamente é “para baixo”.

Referências

- [1] Rose, K. H. *Xy-pic User's Guide*. Disponível em: <http://tug.org/applications/Xy-pic/soft/xyguide.ps.gz>.
- [2] Rose, K. H. & Moore, R. *Xy-pic Reference Manual*. Disponível em: <http://tug.org/applications/Xy-pic/soft/xyrefer.ps.gz>.
- [3] Goossens, M. & Rahtz, S. & Mittelbach, F. *The L^AT_EX Graphics Companion*, Addison-Wesley, 1997.

Copyright ©2006 Carlos A. P. Campani.

É garantida a permissão para copiar, distribuir e/ou modificar este documento sob os termos da Licença de Documentação Livre GNU (GNU Free Documentation License), Versão 1.2 ou qualquer versão posterior publicada pela Free Software Foundation; sem Seções Invariantes, Textos de Capa Frontal, e sem Textos de Quarta Capa. Uma cópia da licença é incluída na seção intitulada “GNU Free Documentation License”.

veja: <http://www.ic.unicamp.br/~norton/fdl.html>.