Dicas para a preparação de relatórios em formato digital utilizando LaTeX

MAC0239 – Introdução à Lógica e Verificação de Programas

criado por Luiz Carlos Vieira (outubro de 2015)

1 Introdução

Você pode fazer seus relatórios em formato digital utilizando qualquer editor. Porém, há muitas vantagens em usar o LaTeX pra isso.

Primeriamente, você não precisa se preocupar com a formatação, pois o LaTeX faz isso sozinho pra você. Você apenas constrói o texto e usa marcadores (comandos) especiais para indicar o que de especial quer construir. Por exemplo, a notação matemática permite definir equações em um parágrafo separado usando dois cifrões (`$$`) para abrir e fechar o ambiente da equação.

Por exemplo, o código a seguir:

```
$$
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
$$
```

Gera o seguinte resultado:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Também se pode adicionar equações no meio do texto usando apenas um cifrão (`$`) como delimitador. Por exemplo, a fórmula $a^2 = b^2 + c^2 - 2b \cdot c \cdot \cos\theta$ é produzida pelo código:
`$a^2 = b^2 + c^2 - 2b \cdot c \cdot \cos{\theta}$`

No nosso caso, a notação matemática é utilizada para escrever as fórmulas lógicas estudadas. Alguns exemplos:

O código:

```
$$
\lnot p \to q
$$
```

Gera o seguinte resultado:

$$\neg p \to q$$

O código:

```
$$
\{ \phi_1, \phi_2, \dots, \phi_n \} \vdash \psi
$$
```

Gera o seguinte resultado:

$$\{\phi_1, \phi_2, \dots, \phi_n\} \vdash \psi$$

O código:

```
$$
\Delta \models \psi
$$
```

Gera o seguinte resultado:

$$\Delta \models \psi$$

Há um pacote que estende muitíssimo as capacidades de representação matemática do LaTeX chamado "amsmath". Para utilizá-lo, inclua o comando `\usepackage{amsmath}` no preâmbulo do seu documento. Uma ajuda completa sobre a notação matemática (do pacote "amsmath"), com os símbolos e seus respectivos comandos, está disponível neste guia (PDF em inglês): ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf.

Em segundo lugar, o LaTeX conta com um bom número de pacotes que facilitam enormemente a criação de provas de dedução natural, arvores sintáticas e diagramas de decisão binária. Esses serão explorados a seguir neste documento.

1.1 Criando os documentos online com o Overleaf

A ferramenta Overleaf (https://www.overleaf.com) é gratuita (com alguma limitação de espaço para os projetos) e permite criar os documentos online. É uma ótima alternativa, principalmente para trabalhos colaborativos (o link do projeto pode ser compatilhado). Mas, para quem preferir trabalhar off-line, basta fazer o download do LaTeX para o sistema operacional de sua preferência em http://latex-project.org/ftp.html.

O código deste documento também está acessível no Overleaf, neste link: https://www.overleaf.com/read/prbcswrntmtt.

1.2 Material de apoio sobre o LaTeX

A lista a seguir reune algumas fontes que podem ser utilizadas para aprender mais sobre o LaTeX:

- Tutorial de LaTeX para escrita científica: http://sbi.iqsc.usp.br/files/Manual-SBI_LATEX_2013-.pdf

- Introdução ao LaTeX: http://www.mat.ufmg.br/~regi/topicos/intlat.pdf
- LaTeX wikibooks (em inglês): https://en.wikibooks.org/wiki/LaTeX
- LaTeX for logicians (em inglês): http://www.logicmatters.net/latex-for-logicians/
- TextStudio (editor multiplataforma popular): http://www.texstudio.org/

2 Criando o Processo de Inferência

Usando apenas a notação matemática do LaTeX é possível construir fórmulas da lógica proposicional. Mas pode ser muito útil usar também o pacote de macros "logicproof" (inclua o comando `\usepackage{logicproof}` no preâmbulo do documento), que permite fazer processos de dedução no mesmo estilo apresentado em sala de aula (com endentações e quadros).

Esse pacote pode já estar disponível na sua instalação do LaTeX, mas caso não esteja ele pode ser baixado daqui: http://www.ctan.org/tex-archive/macros/latex/contrib/logicproof. A documentação pode ser encontrada aqui (em inglês): http://repositorios.cpai.unb.br/ctan/macros/latex/contrib/logicproof/logicproof.pdf.

2.1 Exemplos utilizando o pacote "logicproof"

2.1.1 Exemplo 1

Prove a validade do argumento: $p \land q \to r \ \vdash \ p \to (q \to r)$

Detalhamento da prova utilizando o pacote "logicproof":

1.	$p \land q \to r$	premissa
2.	p	hipótese
3.	q	hipótese
4.	$p \land q$	$\land_i (2)(3)$
5.	r	$\to_e (1)(4)$
6.	$q \to r$	$\to_i (3-5)$
7.	$p \to (q \to r)$	$\to_i (2-6)$

Abaixo se encontra o código utilizado para criar essa prova:

```
\begin{logicproof}{2}
    p \land q \to r & premissa\\
    \begin{subproof}
        p & hipótese\\
        \begin{subproof}
            q & hipótese\\
            p \land q & $\land_i (2) (3)$\\
            r & $\to_e (1) (4)$
        \end{subproof}
        q \to r & $\to_i (3-5)$
    \end{subproof}
    p \to (q \to r) & $\to_i (2-6)$
\end{logicproof}
```

Nesse código, as linhas são construídas utilizando um & para separar a expressão lógica do texto da justificativa, assim:

```
<expressão> & <justificativa>
```

E cada quadro é construído com um ou mais blocos *subproof* aninhados, assim:

```
\begin{subproof}
    <expressão> & <justificativa>\\
    \begin{subproof}
        <expressão> & <justificativa>\\
        <expressão> & <justificativa>
    \end{subproof}
    <expressão> & <justificativa>\\
    <expressão> & <justificativa>
\end{subproof}
```

Observações importantes:

- O número 2 na linha com o comando `\begin{logicproof}{2}` é o número máximo de quadros (*subproofs*) aninhados. Mude esse número se necessário, sempre de acordo com o número de quadros aninhados na prova que quer representar.
- O comando `\\`, que indica uma quebra de linha forçada, não deve ser utilizado na última linha de um bloco *subproof*.
- O comando `\begin{logicproof}` não precisa estar dentro de um ambiente matemático (isto é, entre $$ e $$), pois ele já cria um ambiente matemático automaticamente. Porém, a justificativa em uma linha é texto puro! Então, se precisar usar símbolos matemáticos nela (como neste exemplo), escreva entre $ e $ (ambiente matemático na linha).
- Note o uso do *underline* (_) para fazer subscritos (o comando matemático para fazer sobrescritos é o acento circunflexo: `^`). A notação matemática utiliza o próximo símbolo ao *underline*, então `$\land_i$` resulta em $\land_i$. Para evitar confusão, quando precisar utilizar mais símbolos dentro de um subscrito (ou fazer subscritos aninhados), use chaves ({ e }) para delimitar o conteúdo. Exemplo: `$\lor_i2$` resulta em $\lor_i 2$ (só o i foi considerado no subscrito); `$\lor_i_2$` resulta em erro de compilação (*double subscript*), e `$\lor_{i_2}$` resulta em $\lor_{i_2}$ (o que é a notação intencionada).

2.1.2 Exemplo 2

Prove a validade do argumento: $\vdash \ (q \to r) \to ((\neg q \to \neg p) \to (p \to r))$

Detalhamento da prova utilizando o pacote "*logicproof*":

1.	$q \to r$	hipótese
2.	$\neg q \to \neg p$	hipótese
3.	p	hipótese
4.	$\neg\neg p$	$\neg\neg_i(3)$
5.	$\neg\neg q$	$MT(2)(4)$
6.	q	$\neg\neg_e(5)$
7.	r	$\to_e\ (1)(6)$
8.	$p \to r$	$\to_i\ (3-7)$
9.	$(\neg q \to \neg p) \to (p \to r)$	$\to_i\ (2-8)$
10.	$(q \to r) \to ((\neg q \to \neg p) \to (p \to r))$	$\to_i\ (1-9)$

Código dessa prova:

```
\begin{logicproof}{3}
    \begin{subproof}
        q \to r & hipótese\\
        \begin{subproof}
            \lnot q \to \lnot p & hipótese\\
            \begin{subproof}
                p & hipótese\\
                \lnot\lnot p & $\lnot\lnot_i (3)$\\
                \lnot\lnot q & $MT (2) (4)$\\
                q & $\lnot\lnot_e (5)$\\
                r & $\to_e (1) (6)$
            \end{subproof}
            p \to r & $\to_i (3-7)$
        \end{subproof}
        (\lnot q \to \lnot p) \to (p \to r) & $\to_i (2-8)$
    \end{subproof}
    (q \to r) \to ((\lnot q \to \lnot p) \to (p \to r)) &
                                                $\to_i (1-9)$
\end{logicproof}
```

3 Criando árvores e diagramas de decisão binária

O "TikZ" é um dos pacotes mais poderosos do LaTeX, pois permite criar gráficos, figuras e diagramas de todo o tipo. O guia "A very minimal introduction to TikZ" (em inglês) serve como uma ótima introdução para quem desejar explorar esse ótimo pacote mais a fundo. Há também este outro site com inúmeros exemplos, incluindo desde grafos e mapas meitais a calendários e tablaturas musicais. O pacote também já deve estar disponível na sua instalação do LaTeX, mas pode ser baixado daqui: `https://www.ctan.org/pkg/pgf`.

Os exemplos desse documento utilizam configurações específicas (árvores e posicionamento de nós), então a declaração no preâmbulo é feita assim:

```
\usepackage{tikz}
\usetikzlibrary{trees,positioning}
```

3.1 Exemplos utilizando o pacote "TikZ"

3.1.1 Exemplo 1 - Árvore Sintática

Considere a função $\phi \equiv (p \wedge r) \vee (\neg p \wedge q \wedge r)$. Sua árvore sintática é a seguinte:

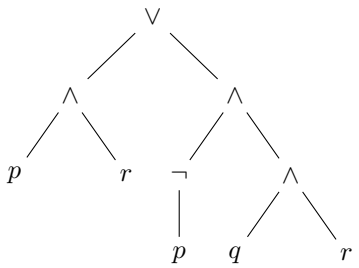

Figura 1: Árvore sintática de $\phi \equiv (p \wedge r) \vee (\neg p \wedge q \wedge r)$

O código para a construção dessa árvore utiliza um bloco `tikzpicture` para desenho, e vários blocos do tipo `node` (para desenhar um nó da árvore) e do tipo `child` (para estabelecer a relação hierárquica em árvores). O bloco `tikzpicture` recebe também alguns parâmetros que permitem configurar a distância entre os nós em um mesmo nível (*siblings*), a escala do desenho, e detalhes do estilo de desenho geral dos nós (configuração `every node`).

Usando os comandos `node` e `child` faz com que o próprio pacote organize os nós em uma estrutura de árvore pra você. Entretanto, alguns nós podem acabar sobrepostos, o que pode ser configurado ajustando a distância entre os nós de mesmo nível (parâmetro `sibling distance`) ou simplesmente movendo um nó no eixo x com o parâmetro `xshift` no próprio nó (há também o `yshift`) para forçar posicionamento de um nó no eixo y).

Observe que o texto de um nó (colocado entre { e }) também pode ser formatado em notação matemática utilizando $ e $. O código da árvore sintática apresentada anteriormente é o seguinte:

```
\begin{figure}[h!]
    \centering
    \begin{tikzpicture}[sibling distance=6em, scale=0.7,
                            every node/.style = {
                                draw=none, align=center
                            }
                        ]
        \node {$\lor$}
            child { node[xshift=-1em] {$\land$}
                child { node {$p$} }
                child { node {$r$} }
            }
            child { node[xshift=1em] {$\land$}
                child { node {$\lnot$}
                    child { node {$p$} }
                }
                child { node {$\land$}
                    child { node {$q$} }
                    child { node {$r$} }
                }
            };
    \end{tikzpicture}
    \caption{Árvore sintática de $\phi \equiv (p \land r) \lor
                                    (\lnot{p} \land q \land r)$}
\end{figure}
```

Para detalhes sobre os elementos da linguagem, por favor consulte a documentação do pacote.

3.1.2 Exemplo 2 - Árvore de Decisão Binária

Os nós e arestas nos BDDs precisam de estilos customizados para definir a aresta trecejada como "lo" (de *low*) e a aresta sólida como "hi" (de *high*). Essa definição de estilos pode ser reutilizada por todos os diagramas com o comando `tikzset`, sendo definida da seguinte forma:

```
% Configuração geral de estilos para o desenho de ROBDDs
\tikzset{%
    % Estilo para nós de decisão
    decision/.style={shape=circle, draw, solid, align=center},
    % Estilo para nós terminais
    terminal/.style={shape=rectangle, draw, solid,
                                      align=center, scale=0.7},
    % Estilo para a linha tracejada (low) reta
    lo/.style={edge from parent/.style={dashed,draw}},
    % Estilo para a linha tracejada (low) encurvada
    lo-bend/.style={dashed, bend right=15,
        edge from parent path={
            (\tikzparentnode\tikzparentanchor)
                edge [dashed, draw, bend right=15]
            (\tikzchildnode\tikzchildanchor)
        }
    },
    % Estilo para a linha solida (high) reta
    hi/.style={edge from parent/.style={solid,draw}},
    % Estilo para a linha solida (high) encurvada
    hi-bend/.style={solid, bend left=15,
        edge from parent path={
            (\tikzparentnode\tikzparentanchor)
                edge [solid, draw, bend left=15]
            (\tikzchildnode\tikzchildanchor)
        }
    }
}
```

Os nomes dos estilos estão antes de `/.style` e os parâmetros internos são os mesmos já utilizados na árvore sintática ou alguns novos (para encurvar uma linha – `bend left=15` – ou para fazer o nó ter um formato circular – `shape=circle`). Novamente, para detalhes sobre cada parâmetro, por favor consulte a documentação do pacote.

Este exemplo reutiliza a mesma função $\phi \equiv (p \wedge r) \vee (\neg p \wedge q \wedge r)$ do exemplo anterior. A árvore de decisão binária criada para ele é a seguinte:

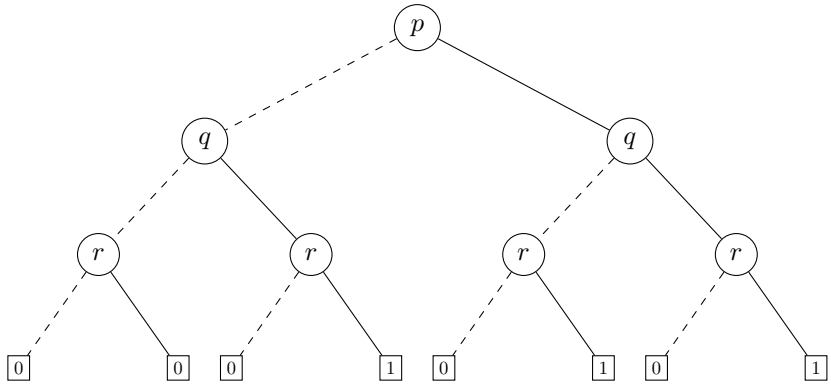

Figura 2: Árvore de decisão binária para $\phi \equiv (p \wedge r) \vee (\neg p \wedge q \wedge r)$

Apenas para mérito de comparação, essa é a tabela-verdade de ϕ – observe como a árvore de decisão tem exatamente as mesmas combinações de valorações para as variáveis, tanto que a sequência de nós terminais (da esquerda para a direita) e a mesma da valoração da fórmula (última coluna, de cima pra baixo):

Tabela 1: Tabela-verdade de $\phi \equiv (p \wedge r) \vee (\neg p \wedge q \wedge r)$

p	q	r	$\neg p$	$p \wedge r$	$\neg p \wedge q \wedge r$	$(p \wedge r) \vee (\neg p \wedge q \wedge r)$
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	1	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	1	0	1	0	1
1	1	0	0	0	0	0
1	1	1	0	1	0	1

O código para a criação dessa árvore é apresentado a seguir. Observe alguns detalhes:

- Os nós (`node`) e as arestas (`child`) utilizam os estilos definidos anteriormente ("decision" e "terminal" para os nós, e "lo" e "hi" para as arestas).
- Os nós do segundo nível são afastados (para a esquerda e para a direita, usando valores negativo e positivo no comando `xshift`) para deixar o aspecto visual mais agradável.
- O texto dos nós é definido utilizando a notação matemática (`$ <conteúdo> $`), para deixar condizente com as mesmas fontes usadas na criação das fórmulas.
- O bloco `figure` não é do pacote "TikZ", e sim do LaTeX. Ele permite colocar a figura em um *floater* (uma caixa que flutua junto com o texto, utilizada justamente para exibir figuras). Note que o bloco `tikzpicture` é colocado dentro do bloco `figure`. Para detalhes, consulte a documentação do LaTeX.

Eis o código para criação da árvore:

```
\begin{figure}[h!]
    \centering
    \begin{tikzpicture}[sibling distance=6em, scale=1.0]
        \node[decision] {$p$}
            child[lo, xshift=-5em] { node[decision] {$q$}
                child[lo, xshift=-1em] { node[decision] {$r$}
                    child[lo] { node[terminal] {$0$}}
                    child[hi] { node[terminal] {$0$} }
                }
                child[hi, xshift=1em] { node[decision] {$r$}
                    child[lo] { node[terminal] {$0$}}
                    child[hi] { node[terminal] {$1$} }
                }
            }
            child[hi, xshift=5em] { node[decision] {$q$}
                child[lo, xshift=-1em] { node[decision] {$r$}
                    child[lo] { node[terminal] {$0$}}
                    child[hi] { node[terminal] {$1$} }
                }
                child[hi, xshift=1em] { node[decision] {$r$}
                    child[lo] { node[terminal] {$0$}}
                    child[hi] { node[terminal] {$1$} }
                }
            };
    \end{tikzpicture}
    \caption{Árvore de decisão binária para $\phi \equiv
                (p \land r) \lor (\lnot{p} \land q \land r)$}
\end{figure}
```

3.1.3 Exemplo 3 - Diagrama de Decisão Binária

Se as simplificações C1-C3 forem efetuadas sobre a árvore do exemplo anterior, o seguinte diagrama de decisão binária (de ordem $[p, q, r]$ e reduzido) será obtido:

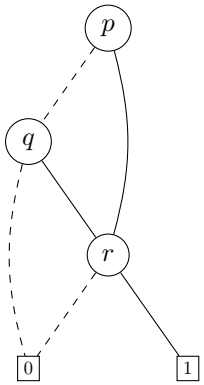

Figura 3: ROBDD para $\phi \equiv (p \wedge r) \vee (\neg p \wedge q \wedge r)$

O código para a criação desse diagrama é o seguinte:

```
\begin{figure}[h!]
    \centering
    \begin{tikzpicture}[sibling distance=6em, scale=1.0]
        \node[decision](p) {$p$}
            child[lo, xshift=-3em] { node[decision](q) {$q$}
                child[hi, xshift=3em] { node[decision](r) {$r$}
                    child[lo] { node[terminal](0) {$0$} }
                    child[hi] { node[terminal](1) {$1$} }
                }
            };

        \draw[hi-bend] (p) to (r);
        \draw[lo-bend] (q) to (0);
    \end{tikzpicture}
    \caption{ROBDD para $\phi \equiv (p \land r) \lor
                                (\lnot{p} \land q \land r)$}
\end{figure}
```

Observe que como esse diagrama não é mais uma árvore e sim um grafo, há a necessidade de fazer ligações entre nós *que não são filhos*. Por exemplo, o nó p agora tem uma ligação direta com o nó r (que na verdade é filho de q). Essas ligações diretas são efetuadas com o comando `draw`. Ele também permite utilizar os mesmos estilos já criados (e no exemplo usamos as arestas curvas apenas para dar um visual bacana ao diagrama). Observe também que ser possível ligar os nós, eles precisam ser *nomeados*. Isso é feito nomeando os nós. Por exemplo, no código `child[lo] { node[terminal](0) {$0$} }` o 0 entre parênteses (em vermelho) é o nome desse nó terminal, utilizado para fazer a conexão com o nó q no código: `\draw[lo-bend] (q) to (0);`.