Estructuras Discretas 2017-1
Panorama general de LaTeX

Favio E. Miranda Perea
Victor Zamora Gutiérrez
Fernando A. Galicia Mendoza

Facultad de ciencias, UNAM

Viernes, 30 de septiembre de 2016

1. Aspectos básicos

Para generar un `PDF` ingresar en terminal: `pdflatex NombreArchivo.tex`
Un sitio para buscar símbolos: `http://detexify.kirelabs.org/`

Para generar portada: `\maketitle`
Para salto de línea: `\\`
Para **negritas**: `\textbf{texto}`
Para *itlaicas*: `\textit{texto}`
Para `máquina de escribir`: `\texttt{texto}`

2. Secciones, capítulos, etc

En caso de los ambientes `article` o bien `report`, pueden definir:

Sección Utilizar comando `\section{Nombre_sección}`

Subsección Utilizar comando `\subsection{Nombre_subsección}`

En el caso del ambiente `book`, pueden definir capítulos:
Utilizar el comando `\chapter{Nombre_capítulo}`

3. Listas enumeradas, no enumeradas y descriptivas

Para crear una lista enumerada utilizar el ambiente `itemize`.
Por ejemplo la siguiente lista:

1. H
2. O
3. L
4. A

Se escribe en latex como:

```
\begin{enumerate}
\item H
\item O
\item L
\item A
\end{enumerate}
```

Para listas no enumeradas:

- H
- O
- L
- A

Se escribe en latex como:

```
\begin{itemize}
\item H
\item O
\item L
\item A
\end{itemize}
```

El ambiente `enumerate` predefinido tiene un problema a poner listas con incisos, para arreglar esto, importar el paquete `enumerate`.
Para listas descriptivas:

descripcion1 H

descripcion2 O

descripcion3 L

descripcion4 A

Se escribe en latex como:

```
\begin{description}
\item[descripcion1] H
\item[descripcion2] O
\item[descripcion3] L
\item[descripcion4] A
\end{description}
```

4. Ambiente matemático

El ambiente matemático se puede poner de dos formas:

- `$expresionMatematica$`
- `\[expresionMatematica\]`

La diferencia es que la primera se puede poner sobre la misma línea que el ambiente de texto y la segunda hace un salto de línea y lo centra.
Por ejemplo:
$ax^2 + bx + c \rightarrow$ `$ax^2+bx+c$`

$$ax^2 + bx + c \rightarrow$$

`\[ax^2+bx+c\]`

5. Símbolos lógicos

Para utilizar símbolos lógicos deben ponerlos en ambiente matemático, a continuación se muestra una lista con los posibles símbolos:

- Conectivos:
  - Negación ($\neg$): `\lnot`
  - Conjunción ($\wedge$): `\land`
  - Disyunción ($\vee$): `\lor`
  - Implicación ($\rightarrow$): `\to`
  - Doble condicional ($\leftrightarrow$): `\leftrightarrow`
  - Disyunción exclusiva ($\not\equiv$): `\not\equiv`
  - Conjunción negada ($\uparrow$): `\uparrow`
  - Disyunción negada ($\downarrow$): `\downarrow`
- Cuantificadores:
  - Universal ($\forall$): `\forall`
  - Existencial ($\exists$): `\exists`
- Universo de discurso ($\mathcal{U}$): `\mathcal{U}`
- Lenguaje ($\mathcal{L}$): `\mathcal{L}`
- Derivación semántica ($\vDash$): `\vDash`
- Derivación sintactica ($\vdash$): `\vdash`

6. Relaciones

Para utilizar símbolos de relación deben ponerlos en ambiente matemático, a continuación se muestra una lista con las relaciones usuales:

- Relación de igualdad ($=$): `=`
- Relación menor estricta ($<$): `<`
- Relación menor o igual ($\leq$): `\leq`

- Relación mayor estricta ($>$): `>`
- Relación mayor o igual ($\geq$): `\geq`

7. Definiciones, teoremas y demás

Para definir un teorema debemos definir un nuevo estilo de teorema, a través de la orden `\newtheorem{etiqueta}{Nombre}[Numeracion]`, donde `etiqueta` es donde se creará el ambiente, `Nombre` es el nombre de la etiqueta (defi a Definición, por ejemplo) y `Numeración` es la opción para indicarle a LaTeXque la numeración de los teoremas sea por sección, capítulo, etc.
Por ejemplo:
`\newtheorem{defi}{Definición}[section]` indica que el ambiente `defi`, se llamará `Definición` y será enumerado de acuerdo a la sección.
Para crear una definición creamos un ambiente con `defi` y podemos poner entre corchetes un nombre propio a la definición.
Por ejemplo:

Definición 1 (Longitud de lista)
Sea ℓ una lista de tipo a, definimos la longitud de ℓ como el total de elementos que contiene la lista, es decir, si $\ell = [x_1, \ldots, x_n]$, entonces la longitud de ℓ es n.

Dentro del archivo `.tex` debe ir:

```
\begin{defi}[Longitud de lista]
  Sea $\ell$ una lista de tipo $a$, definimos la longitud de $\ell$
  como el total de elementos que contiene la lista, es decir,
  si $\ell = [x_1,\dots,x_n]$, entonces la longitud de $\ell$ es $n$.
\end{defi}
```

8. Arreglos y sistemas de ecuaciones

Código LaTeX:

```
\[
\begin{array}{|l|cr}
```

```
  left1 & center1 & right1\\
  \hline
  d & e & f
\end{array}
\]
```

Resultado:

$left1$	$center1$	$right1$
d	e	f

Código LaTeX:

```
\[
\begin{array}{lcl}
  z & = & a \\
  & = & a \\
  f(x,y,z) & = & x + y + z
\end{array}
\]
```

Resultado:

$$\begin{array}{lcl} z & = & a \\ & = & a \\ f(x,y,z) & = & x + y + z \end{array}$$

Código LaTeX:

```
\begin{align*}
  z & = & a \\
  & = & a \\
  f(x,y,z) & = & x + y + z
\end{align*}
```

Resultado:

$$\begin{aligned} z &= a \\ &= a \\ f(x,y,z) &= x + y + z \end{aligned}$$

Código LaTeX:

```
\begin{align}
  z & = & a \\
  & = & a \\
  f(x,y,z) & = & x + y + z
\end{align}
```

Resultado:

$$\begin{aligned} z &= a && (1)\\ &= a && (2)\\ f(x,y,z) &= x+y+z && (3) \end{aligned}$$

9. Algoritmos

Código LaTeX:

```
\begin{algorithm}[H]
  \KwData{Entradas}
  \KwResult{Salida del algoritmo}
  print "Hello world''\;
  \While{guardia}{
    instruccion\;
    \eIf{guardia}{
      instruccion true\;
    }{
      instruccion else\;
    }
  }
  \caption{Como escribir algoritmos}
\end{algorithm}
```

Resultado:
Data: Entradas
Result: Salida del algoritmo
print "Hello world";
while *guardia* **do**
  instruccion;
  if *guardia* **then**
    instruccion true;
  else
    instruccion else;
  end
end

Algoritmo 1: Como escribir algoritmos

10. Código

Código LaTeX:

```
\begin{code}
```

```
--Poner codigo Haskell

module PostFix where

--Palabra reservada postfix
data PF = POSTFIX deriving(Show,Eq)

--Tipo que define un comando postfix
data Comando = L Int | EXEC | ADD | SUB | MUL | DIV | REM | Eq | Gt | Lt |

--La pila es una lista de comandos
type Pila = [Comando]

--Un programa es una tupla la cual tiene la palabra reservada postfix,
--un numero natural y una lista de comandos
type Programa = (PF,Int,[Comando])

{-
  - Verificacion de la sintaxis de un programa
  -}

--Funcion que determina que un porgrama este sintacticamente bien formado
bienFormado :: Programa -> Pila -> Bool
bienFormado (p,n,l) s = (prPF (p,n,l)) && (n == length s)
```

\end{code}

Resultado:

```
--Poner codigo Haskell

module PostFix where

--Palabra reservada postfix
data PF = POSTFIX deriving(Show,Eq)

--Tipo que define un comando postfix
data Comando = L Int | EXEC | ADD | SUB | MUL | DIV | REM | Eq | Gt |
Lt | SEL | NGET | POP | SWAP | SEC [Comando]  deriving(Show,Eq)
```

```
--La pila es una lista de comandos
type Pila = [Comando]

--Un programa es una tupla la cual tiene la palabra reservada postfix,
--un numero natural y una lista de comandos
type Programa = (PF,Int,[Comando])

{-
  - Verificacion de la sintaxis de un programa
  -}

--Funcion que determina que un porgrama este sintacticamente bien formado
bienFormado :: Programa → Pila → Bool
bienFormado (p,n,l) s = (prPF (p,n,l)) && (n == length s)
```

Observación: Aunque parezcan, observen que el ambiente `code` evita que salga del documento y lo pone en un estilo completamente parecido a un editor de texto.