Boolean algebra

Variable of boolean type can have only two values

- true (Matlab uses 1 to indicate it, actually everything but zero)
- false (Matlab uses 0)
Boolean algebra

Variable of boolean type can have only two values
- true (Matlab uses 1 to indicate it, actually everything but zero)
- false (Matlab uses 0)

There are three logical operators which are used in boolean algebra
- ¬ - logic not, Matlab
  - ¬true = false
  - ¬false = true
- ∧ - logic and, Matlab
  \[ A \land B = \begin{cases} 
  \text{true, if } A=\text{true and } B=\text{true}, \\
  \text{false, otherwise} 
\end{cases} \]
- ∨ - logic or, Matlab
  \[ A \lor B = \begin{cases} 
  \text{false, if } A=\text{false and } B=\text{false}, \\
  \text{true, otherwise} 
\end{cases} \]
Boolean operators precedence in Matlab

If $A = \text{false}$, $B = \text{true}$, $C = \text{true}$

$A | \neg B \& C$

\neg has highest precedence, then \&, and then |

Thus

$A | \neg B \& C = \text{false}$
If $A = \text{false}$, $B = \text{true}$, $C = \text{true}$

\[ A \neg B \land C \]

$\neg$ has highest precedence, then $\land$, and then $|$  

\[ A \neg ((\neg B) \land C) \]

Thus

\[ A \neg B \land C = \text{false} \]

“Cat is an animal and cat is not an animal”

is false statement
There is an island, which is populated by two kind of people: liars and truthlovers.

- Liars always lie and never speak a word of truth.
- Truthlovers always speak only truth.

Suppose, you are landed on this island and met a person. What will be the answer to your question “Who are you?”

- The answer always will be “Truthlover”.

Now you see a person who answers to your question. “I am a liar.” Is it possible?

This makes a paradox and should not ever happen on this island.
Matlab boolean logic examples

- `123.3 & 12 = 1`
- `1232e-6 = 0`

```matlab
>> B = [1.22312, 0; 34.343, 12]
B =
    1.2231   0
    34.3430  12.0000
```

To be or not to be

The answer is to be
Matlab boolean logic examples

1.23.3 \& 12 = 1
\sim 1232e-6 = 0

>> B = [1.22312, 0; 34.343, 12]
B =
1.2231 0
34.3430 12.0000

~B

ans =
0 1
0 0

B \| ~B

"To be or not to be"
The answer is to be
Matlab boolean logic examples

```matlab
>> B=[1.22312, 0; 34.343, 12]
B =
  1.2231    0
  34.3430  12.0000
>> A=[56, 655; 0, 24.4]
A =
  56.0000  655.0000
  0       24.4000
```

```matlab
B&A
ans =
  1  0
  0  1
A|~B
ans =
  1  1
  0  1
```
Matlab boolean logic examples

>> B=[1.22312, 0; 34.343, 12]
B =
1.2231 0
34.3430 12.0000

>> A=[56, 655; 0, 24.4]
A =
56.0000 655.0000
0 24.4000

B&A
ans =
1 0
0 1

A|~B
ans =
1 1
0 1

Comparison operators

<table>
<thead>
<tr>
<th>Math</th>
<th>Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>double equal sign!</td>
</tr>
<tr>
<td>~</td>
<td>=</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;=</td>
<td>&gt;=</td>
</tr>
</tbody>
</table>

x=[1,2,3,4,5]
x =
1 2 3 4 5

x >= 3
ans =
0 0 1 1 1

% chose such 'x' where x>=3
x(x >= 3)
ans =
3 4 5

Notes
Comparison operators

<table>
<thead>
<tr>
<th>Math</th>
<th>Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>(=)</td>
<td>(==) double equal sign!</td>
</tr>
<tr>
<td>(&lt;)</td>
<td>(\sim)</td>
</tr>
<tr>
<td>(\leq)</td>
<td>(\leq)</td>
</tr>
<tr>
<td>(\geq)</td>
<td>(\geq)</td>
</tr>
</tbody>
</table>

\[ x = [1, 2, 3, 4, 5] \]
\[
x = \\
1 \quad 2 \quad 3 \quad 4 \quad 5
\]
\[ x \geq 3 \]
\[
\text{ans} = \\
0 \quad 0 \quad 1 \quad 1 \quad 1
\]

Comparison with matrices

\[
\text{>> } A = [1, 2; 3, 4] \\
A = \\
1 \quad 2 \\
3 \quad 4
\]
\[
\text{>> } B = [33, 11; 53, 42] \\
B = \\
33 \quad 11 \\
53 \quad 42
\]

\[ x(x \geq 3) \]
\[
\text{ans} = \\
3 \quad 4 \quad 5
\]

\[ x(x \leq 3) \]
\[
\text{ans} = \\
0 \quad 0 \quad 1 \quad 1 \quad 1
\]
### Comparison with matrices

```matlab
>> A = [1, 2; 3, 4]
A =
1  2
3  4

>> B = [33, 11; 53, 42]
B =
33 11
53 42

A >= 2
ans =
0  1
1  1

A(A >= 2)
ans =
3  2
4

B(A >= 2)
ans =
53 11
42
```

Chose such elements of B where elements of A \( \geq 2 \):

ans =
53 11
42
Comparison with matrices

```matlab
>> A=[1,2;3,4]
A =
 1 2
 3 4

>> B=[33,11;53,42]
B =
 33 11
 53 42

A>=2
ans =
0 1
1 1

A(A>=2)
ans =
3
2
4

B(A>=2)
ans =
53
11
42

Chose such elements of B where elements of A ≥ 2
```

if-else-end statement

```matlab
if hungry
buy some food
else
keep working
end
```

Notes
### if-else-end statement

```
if expression
  this part is executed only if expression is true
else
  this part is executed only if expression is false
end
```

### Common mistake in the 'if' statement

```
if (x=y)
  D=4;
  Z=45;
  C=12;
else
  D=2;
end
```

The value of 'D' is always 4, except the case when y=0

Someone used assignment operator (=) instead of comparison (==)
Short form of 'if-end' statement

```
if expression
this part is executed
only if expression is true
end
```

if expression
if won a million
  go party
end

if (deviation<=0)
  exit;
end

---

The 'while' statement

```
while expression
this part is executed
while expression is true
end
```

while hungry
  keep eating
end

```
i=1;
while (i<=10)
  c=a+b;
  z=c*4+5;
  i=i+2;
end
```

while loop is extremely useful but they are not guaranteed to finish.

For a bit more complicated conditional statement and loop it is impossible to predict if the loop will finish.

Yet another common mistake is:

```
i=1;
while (i<=10)
  c=a+b;
end
```

not updating the term leading to fulfillment of the while condition.
The 'while' statement

while expression
this part is executed
while expression is true
end

while hungry
keep eating
end

while loop is extremely useful but they are not guaranteed to finish.
For a bit more complicated conditional statement and loop it is impossible to predict if the loop will finish.

Yet another common mistake is

i=1;
while (i<=10)
c=a+b;
z=c*4+5;
i=i+2;
end

while loop is extremely useful but they are not guaranteed to finish.
For a bit more complicated conditional statement and loop it is impossible to predict if the loop will finish.

Yet another common mistake is

i=1;
while (i<=10)
c=a+b;
end
The 'while' statement

while expression this part is executed while expression is true end

while loop is extremely useful but they are not guaranteed to finish.
For a bit more complicated conditional statement and loop it is impossible to predict if the loop will finish.

Yet another common mistake is

i=1;
while (i<=10)
c=a+b;
z=c*4+5;
i=i+2;
end

not updating the term leading to fulfillment of the while condition

The 'for' statement

for variable = expression do something end

In this case variable is assigned consequently with columns of the expression, and then statements inside of the loop are executed

sum=0;
x=[1,3,5,6]
for v=x
    sum=sum+v;
end

>> sum
sum =
15

for loops are guaranteed to complete after predictable number of iterations (the amount of columns in expression).
Example

\[ S = \sum_{i=1}^{100} i = 1 + 2 + 3 + \cdots + 99 + 100 \]

\[
S=0; \ i=1; \\
\text{while}(i<=100) \\
\quad S=S+i; \\
\quad i=i+1; \\
\text{end}
\]

Example

\[ S = \sum_{i=1}^{100} i = 1 + 2 + 3 + \cdots + 99 + 100 \]

\[
S=0; \ i=1; \\
\text{while}(i<=100) \\
\quad S=S+i; \\
\quad i=i+1; \\
\text{end}
\]

Example

\[ S = \sum_{k=1}^{100} a_k \]

While \( k \leq 100 \) and \( a_k \geq 10^{-5} \), where \( a_k = k^{-k} \).
Example

\[ S = \sum_{k=1}^{100} a_k \]

While \( k \leq 100 \) and \( a_k \geq 10^{-5} \), where \( a_k = k^{-k} \).

\[
S=0; \quad k=1;
\text{while} \ ( (k\leq100) \& \ (k^{-k} \geq 1e-5)) \text{ }
S=S+k^{-k};
\quad k=k+1;
\text{end}
\]

\[ S = 1.2913 \]
Same example with 'for' loop and use of matrix ops

\[ S = \sum_{k=1}^{100} a_k \]

While \( k \leq 100 \) and \( a_k \geq 10^{-5} \), where \( a_k = k^{-k} \).

```matlab
S=0;
for k=1:100
    a_k=k^-k;
    if (a_k < 1e-5)
        break;
    end
    S=S+a_k;
end
>> S
S =
1.2913
```

Often it is more elegant to use built in Matlab matrix operators

```matlab
k=1:100;
a_k=k.^-k;
S=sum(a_k(a_k>=1e-5))
S =
1.2913
```

Note use of the choose elements construct built in sum function

Interest rate related example

Suppose bank gave you 50% interest rate (let's call it 'x'), and you put one dollar in.

How much would you get at the end of the year?

- one payment at the end of the year
  \[ M_1 = 1 \times (1 + x) = 1 \times (1 + 0.5) = 1.5 \]
Interest rate related example

Suppose bank gave you 50% interest rate (let’s call it ‘x’), and you put one dollar in.
How much would you get at the end of the year?

- one payment at the end of the year
  \[ M_1 = 1 \times (1 + x) = 1 \times (1 + .5) = 1.5 \]

- interest payment every half a year
  \[ M_2 = 1 \times (1 + x/2) \times (1 + x/2) = 1 \times (1 + .5/2)^2 = 1.5625 \]

- interest payment every month
  \[ M_{12} = 1 \times (1 + x/12)^{12} = 1.6321 \]

Now let’s find how your return on investment \( M_N \) depends on the number of payments per year

```matlab
x=.5; N_max=100; N=1:N_max;
M=0*(N); % since N is vector M will be a vector too
for i=N
M(i)=(1+x/i)^i;
end
plot(N,M,'-'); set(gca,'FontSize',24);
xlabel('N, number of payments per year');
ylabel('M_n, return on investment'); % note M_n use title('Return on investment vs number of payments');
```

Of course we do not need computer to show that 
\[ M_\infty = e^x = 1.6487 \]
but we need it to calculate something like 
\[ M_{1001} - M_{1000} = 2.0572 \times 10^{-7} \]

Bonus question: can you calculate \( M \) without use of loops?
Interest rate related example

Return on investment vs number of payments

$M_n$, return on investment

$N$, number of payments per year

Eugeny Mikhailov (W&M) — Practical Computing — Lecture 03 19 / 19