## Logic in proofs

<table>
<thead>
<tr>
<th>Rule of Inference</th>
<th>Tautology</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>( p \rightarrow (p \lor q) )</td>
<td>Addition</td>
</tr>
<tr>
<td>( p \land q )</td>
<td>( (p \land q) \rightarrow p )</td>
<td>Simplification</td>
</tr>
<tr>
<td>( p \rightarrow q )</td>
<td>( p \land q \rightarrow q )</td>
<td>Conjunction</td>
</tr>
<tr>
<td>( q )</td>
<td>( p \rightarrow q \rightarrow r \rightarrow (p \rightarrow r) )</td>
<td>Hypothetical Syllogism</td>
</tr>
<tr>
<td>( p \lor q, \neg p \rightarrow q )</td>
<td>( ((p \lor q) \land (\neg p) \rightarrow q )</td>
<td>Disjunctive Syllogism</td>
</tr>
<tr>
<td>( p \lor q, \neg p \lor r \rightarrow q )</td>
<td>( (p \lor q) \land (\neg p \lor r) \rightarrow q )</td>
<td>Resolution</td>
</tr>
</tbody>
</table>
Rules of Inference

• Addition

\[
\begin{array}{c}
p \\
\hline \\
\vdots \\
p \lor q
\end{array}
\]

- RedSox will win \( p \)
- RedSox or the Mets will win \( p \lor q \)

• Simplification

\[
\begin{array}{c}
p \land q \\
\hline \\
\vdots \\
p
\end{array}
\]

- RedSox will win and The Yankees will not lose \( p \land q \)
- RedSox will win \( p \)

Rules of Inference

• Conjunction

\[
\begin{array}{c}
p \\
\hline \\
q \\
\vdots \\
p \land q
\end{array}
\]

- RedSox will win \( p \)
- The Yankees will loose \( q \)
- The RedSox will win and the Yankees will lose \( p \land q \)
Rules of Inference

- **Modus Ponens**

  \[
  \begin{align*}
  & p \rightarrow q \\
  & p \\
  \hline
  & q
  \end{align*}
  \]

  - If it is raining or snowing, the ground is wet \((R \lor S) \rightarrow W\)
  - It is raining or snowing \((R \lor S)\)
  - The ground is wet \(W\)

- **Modus Tollens**

  \[
  \begin{align*}
  & p \rightarrow q \\
  & \neg q \\
  \hline
  & \neg p
  \end{align*}
  \]

  - If it is raining or snowing, the ground is wet \((R \lor S) \rightarrow W\)
  - The ground is not wet \(\neg W\)
  - It is not raining nor snowing \(\neg (R \lor S)\)
Rules of Inference

• Hypothetical syllogism

\[
\begin{align*}
p &\rightarrow q \\
q &\rightarrow r \\
\therefore p &\rightarrow r
\end{align*}
\]

- If it is raining, the ground is wet \hspace{1cm} p \rightarrow q
- If the ground is wet, use an umbrella \hspace{1cm} q \rightarrow r
- If it is raining, use an umbrella

Rules of Inference

• Disjunctive syllogism

\[
\begin{align*}
p &\lor q \\
\neg p &
\end{align*}
\]

\[
\therefore q
\]

- It is either snowing or raining \hspace{1cm} p \lor q
- It is not snowing \hspace{1cm} \neg p
- It is raining \hspace{1cm} q
Rules of Inference

• Resolution

\[
\begin{array}{c}
\text{p} \lor \text{q} \\
\neg \text{p} \lor \text{r} \\
\hline
\therefore \text{q} \lor \text{r}
\end{array}
\]

- It is snowing or raining
- It is not snowing or hale
- It is raining or hale

\[
P \lor Q \\
\neg P \lor R \\
Q \lor R
\]

• Constructive dilemma

\[
\begin{array}{c}
\text{p} \lor \text{q} \\
\text{p} \rightarrow \text{r} \\
\text{q} \rightarrow \text{s} \\
\hline
\therefore \text{r} \lor \text{s}
\end{array}
\]

- Either RedSox or Yankees will win
- If RedSox wins, then Boston goes wild
- If Yankees wins, then NYC goes wild
- Boston or NYC goes wild

\[
P \lor Q \\
P \rightarrow R \\
Q \rightarrow S \\
R \lor S
\]
Example

- Prove that

- \([ (P \lor Q) \rightarrow R] \land [R \rightarrow (S \rightarrow T)] \land [P \land S] \rightarrow T \)
- \([ (A \land B) \lor \neg C] \land [(A \land B) \rightarrow D] \land [E \lor \neg D] \land \neg E \rightarrow \neg C \)
- \([ (\neg I \land J) \rightarrow K] \land [\neg L \rightarrow J] \land [\neg L \land \neg I] \rightarrow K \lor M \)

Simple Exception Handling

```
function Tan (  
    X : Float )  
return Float is
begin  
    return Sin(X) / Cos(X);  
exception  
    when Numeric_Error =>  
        if (Sin(X)>=0.0 and Cos(X)>= 0.0) or (Sin(X)< 0.0 and Cos(X)<= 0.0) then  
            return Float'Last;  
        else  
            return -Float'Last;  
        end if;  
end Tan;  
```
**Exception Handling**

```plaintext
procedure Safe_Get_Float(
    Out_Float : out Float;
    Min, Max : in Float ) is
    Local_Float : Float;
    Good_One : Boolean := False;

begin -- Safe_Get_Float
    while not Good_One loop
        begin
            Put("Enter a float in range ");
            Put( Min, Exp => 0 );
            Put( " to ");
            Put( Max, Exp => 0 );
            Put( ");
            Get( Local_Float );
            -- this point can only be
            -- reached if the get
            -- did not raise the exception
            -- now tested against limits
            -- specified
            Good_One:=((Local_Float)>=Min) and
            (Local_Float<=Max));

            if not Good_One then
                raise Data_Error;
                -- Local_Float < Min OR
                -- Local_Float > Max
            end if;
        end loop;

    skip_Line;

    Out_Float := Local_Float;
    -- export input value
    end Safe_Get_Float;

    Conventional Execution
```

**Exception**

```plaintext
exception
    when Data_Error =>
        Put_Line("DATA ERROR. Invalid input, pls try again ");
        new_line;
        Skip_Line;
end; -- protected block of code

end loop;
    -- this point can only be reached
    -- when valid value input
    Skip_Line;

    Out_Float := Local_Float;
    -- export input value
    end Safe_Get_Float;

    Conventional Execution
```
Infix Evaluation

- Check if the parentheses are balanced
- Parse the input string from left to right
  - If Input(I) is an operand, push it on operand stack
  - If Input(I) is an operator, push it on operator stack
  - If Input(I) = ‘)’
    - Pop two elements from the operand stack
    - Pop the operator from the operator stack
    - Perform computation and Push result back onto operator stack
- The value of the expression is now on top of operand stack

Binary Tree

A binary tree is a tree that is
1. Empty
2. Has two children left, right which are themselves binary trees

Prove that the height of a non-empty binary tree is at least ⌊lg(n)⌋, where n is the number of nodes in the tree.
Proof

• Given:
  – height = 1 + max (height of subtrees)
  – ⌊lg(n)⌋ = ⌊lg(n)-1⌋ if n>2 and n odd

• To prove:

  \[ \text{height(Tree)} \geq \lceil \lg(\text{num}\_\text{nodes}) \rceil \]

Base Case

• For a tree with just the root node (n=1), the theorem holds \( \lg(1) = 0 \)
Inductive Step

- Assume \( n \geq 2 \), theorem holds for \( 1 \leq j < n \)
- Prove that theorem holds for \( j = n \)

Given that \( n \geq 2 \), the tree \( T \) can be split into two subtrees \( T_L \) and \( T_R \)

Assume that both \( T_L \) and \( T_R \) have equal number of nodes.

\[
\left\lfloor \frac{n-1}{2} \right\rfloor \leq T_L \leq \left\lceil n-1 \right\rceil
\]

Given that theorem holds for subtrees, \( \text{Height} \ (T_L) \geq \log \left\lfloor \frac{n-1}{2} \right\rfloor \)

\[ \Rightarrow \text{Height}(T) \geq \left\lfloor \log \left\lfloor \frac{n-1}{2} \right\rfloor + 1 \right\rfloor \]
\[ \geq \left\lfloor 1 + \log \left\lfloor \frac{n-1}{2} \right\rfloor \right\rfloor \]
\[ \geq \left\lfloor \log \left( 2 \left\lceil \frac{n-1}{2} \right\rceil \right) \right\rfloor \]
\[ \geq \left\lfloor \log \ (n) \right\rfloor \]