In OCL, the truth value set consists of the Boolean true and false.

OCL expressions can be composed of:

- literal values
- variables
- relational operators
- arithmetic operators
- set operations
- logical operators
- conditional expressions
- function calls
- method invocations

OCL expressions are evaluated according to the following rules:

1. **Order of Evaluation**: Expressions are evaluated from left to right, with the exception of function calls, where the function name is evaluated first.
2. **Priority of Operators**: Operators are evaluated based on their precedence, with function calls having the highest precedence.
3. **Evaluation Context**: The evaluation context determines the meaning of variables and expressions.
4. **Interpretation of Expressions**: Expressions are interpreted based on their domain type.
UML Notation for Object Diagrams

Object Diagrams are a type of diagram used to represent the structure and behavior of a system. They consist of nodes and edges, where nodes represent objects and edges represent relationships between objects.

Definition.

A node labelled graph is a triple $G = (N, E, f)$ where $N$ is a set of node labels, $E$ is a set of edges, and $f: E \to N \times N$ is a function that assigns to each edge a pair of node labels.

We can equivalently represent $G$ graphically as follows:

Consider $\sigma$ of finite system state.

Node labels are identities (not necessarily alive), i.e., $\sigma \subseteq \mathbb{N}$.

Edges correspond to "links" of objects, i.e., $\sigma \cap \mathbb{N} \subseteq \mathbb{N} \times \mathbb{N}$.

Nodes are "compartment", i.e., $\sigma \cap \mathbb{N} \subseteq |\mathbb{N}| \times \mathbb{N}$.

Graphically as follows:

We may equivalently

\[ \sigma \subseteq \mathbb{N} \times \mathbb{N} \]

An object-labelled graph $\Sigma \in \sigma$ is a system state.

\[ \sigma \in \sigma \]

In which each node is a compartment.

\[ \sigma \in \sigma \]

Groups are objects of $\Sigma$.

\[ \sigma \in \sigma \]

Each group is the object diagram of $\sigma$.

\[ \sigma \in \sigma \]
A finite system state can have:

- Each finite system state has a value comprises an object which is not alive in system state.

If somebody...

\[ \text{A finite system state can have:} \]
\[ \text{each finite system state has} \]
\[ \text{a value comprises an object which is not alive in system state.} \]

Let \( \sigma \in \text{system state}. \)

\[ \text{If somebody} \]
\[ \text{a value comprises an object which is not alive in system state.} \]

\[ \text{A finite system state can have:} \]
\[ \text{each finite system state has} \]
\[ \text{a value comprises an object which is not alive in system state.} \]

Let \( \sigma \in \text{system state}. \)

\[ \text{If somebody} \]
\[ \text{a value comprises an object which is not alive in system state.} \]
SpecialNotation

\(\frac{CB}{\{\text{Int}\}, \{\text{C}\}, \{n,p:\text{C}\}, \text{C} \mapsto \{n,p\}}\).

Instead of \(\frac{1}{\text{C}}\), we want to write \(1\text{C} = \emptyset\).

We slightly deviate from the standard (for reasons):

- In the course, \(\text{C}0, 1\text{C}\) and \(\text{C}\) - typed attributes only have sets as values.
  UML also considers multisets, that is, they can have values like \(u_1 : \text{C} u_2 : \text{C}\).
  (This is not an object diagram in the sense of our definition because of the requirement on the edges \(E\). Extension is straightforward but tedious.)

- We allow to give the evaluation of \(\text{C}0, 1\text{C}\)- or \(\text{C}\) - typed attributes in the values compartment.
  - Allows us to indicate that a certain \(r\) is not referring to another object.
  - Allows us to represent "dangling references", i.e. references to objects which are not alive in the current system state.
  - We introduce a graphical representation of \(\emptyset\) values.

References


