[l, gl, x, r, pr]values

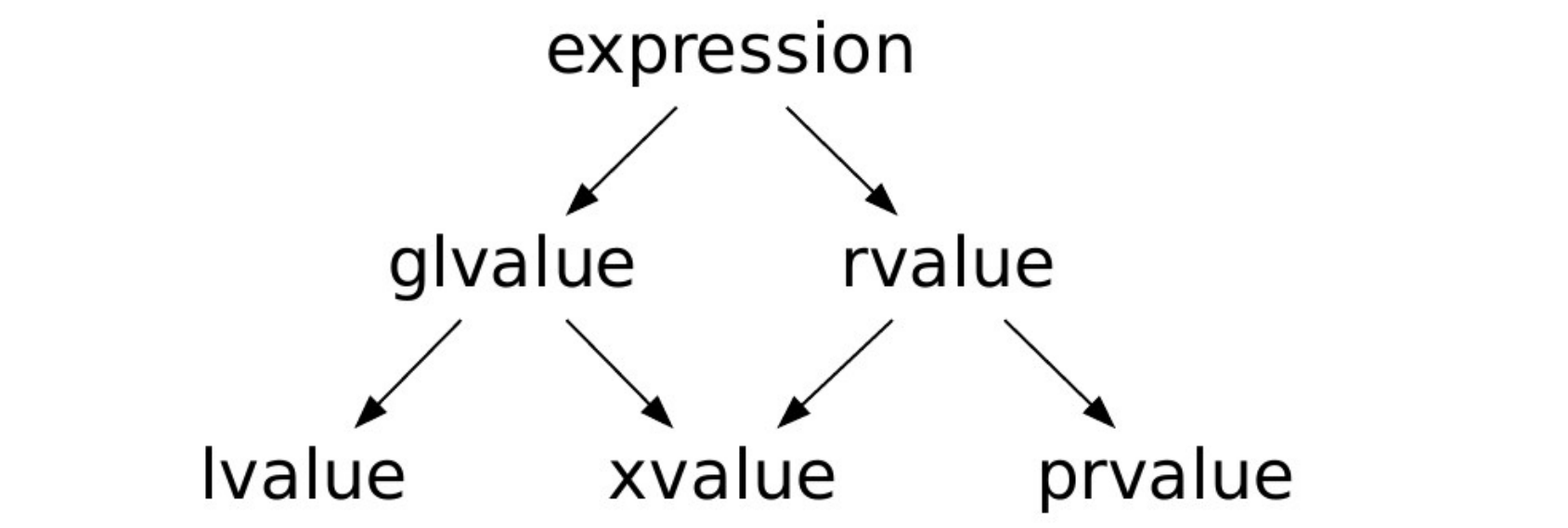
Value categories

Dawid Pilarski

dawid.pilarski@tomtom.com

Introduction

How are expressions categorized?



How to understand fundamental classifications?

- lvalue - T&

How to understand fundamental classifications?

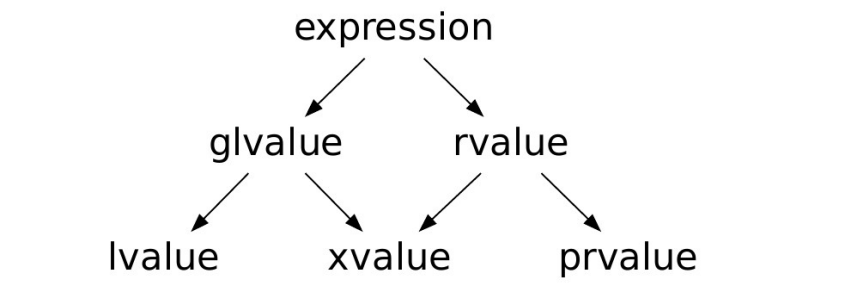
- lvalue - T&
- xvalue - T&&

How to understand fundamental classifications?

- lvalue - T&
- xvalue - T&&
- prvalue - T

The common mistake

Usually people think about expression categories:



As categories of references, which is wrong

Getting it right

$$\textit{category} <=> \textit{expression}$$

$$\textit{reference} => \textit{category}$$

$$\textit{category} \neq> \textit{reference}$$

[Note: there is no reference of type prvalue]

prvalue vs glvalues

glvalues

: Generalized lvalues. It's everything that references the *object*

prvalues

: Pure rvalues. It's a value.

Values vs Objects

Objects	Values
• many object with same value • object can be changed • many references to the same object	• value is unique • value cannot be changed • value

Into the details

Xvalues

xvalues mean:

eXpiring values

Xvalues are such kind of expressions, that its' results point to the object, which will soon expire.

Xvalues examples

There are fixed number of ways we can get xvalues:

- function call which result type is rvalue reference (T&&).
- explicit cast to rvalue reference.
- subscript operator call on the xvalue arrays.
- non reference member access to the xvalue objects (also through pointer to member).
- temporary materialization conversion.

function call which result type is rvalue reference