CLOUSOT

Language-Agnostic Contracts for .NET

Programming Languages and Methods

Contracts in C#

```
public class ArrayList {

  State

  public virtual int Add(object value)
  {
    Contract.Requires(value != null);
    Contract.Ensures(Count == Contract.Old(Count) + 1);
    Contract.Ensures(this[Contract.Result<int>()] == value);
    Contract.Ensures(Contract.Result<int>() == Contract.Old(Count));
    Contract.EnsuresOnThrow<ObjectDisposedException>(this.IsDisposed);

    if (_size == _items.Length) { EnsureCapacity(_size + 1); }

    _items[_size] = value;
    return _size++;
  }

  void ObjectInvariant()
  {
    Contract.Invariant(_items != null);
    Contract.Invariant(_size >= 0);
    Contract.Invariant(_size <= _items.Length);
  }
```

Contracts are:
- Pre-conditions
- Post-conditions
  - Refer to method result
  - Refer to pre-state values
- Object invariants

Logically part of method signatures

Contracts in F#

```
open Microsoft.Contracts

let inc x =
  Contract.Requires(x > 0);
  x + 1

let bad = inc 0

let good = inc 1
```

Contracts about Unsafe code

```
static unsafe private void CopyMemoryProxy(char* pdst, char* psrc,
                                           int size_dst, int size_src)
{
  Contract.Requires(size_dst >= 0);
  Contract.Requires(size_src >= 0);
  Contract.Requires(size_dst >= size_src);
  Contract.Requires(Contract.WritableBytes(pdst) >= (uint)size_dst * sizeof(char));
  Contract.Requires(Contract.WritableBytes(psrc) >= (uint)size_src * sizeof(char));

  // ...
}

public unsafe static void FastCopy(char[] d, char[] s)
{
  if (d == null) throw new NullReferenceException();
  if (s == null) throw new NullReferenceException();
  if (!(d.Length >= s.Length)) throw new ArgumentOutOfRangeException();

  fixed (char* pdst = d)
    fixed(char *psrc = s)
  {
    CopyMemoryProxy(pdst, psrc, d.Length, s.Length);
  }
}
```

Contracts in managed C++

```
int Sum(int* vec, int length) {
  Contract::Requires(vec != nullptr);
  Contract::Requires(Contract::WritableBytes(vec) >= length*sizeof(int));

  int result = 0;
  for (int i = 0; i <length; i++) {
    result += vec[i];
  }
  return result;
}
```

Contracts in VB

```
Public Function CallBinarySearch(ByVal where As Integer(), ByVal value As Integer) As Integer

  Contract.Requires(where IsNot Nothing)

  Dim v As Integer = TechFest.Demo.BinarySearch(where, value)
  If (v <> -1) Then
    Return where(v)
  Else
    Return -22
  End If
```

Contracts …
- make Design Decisions *explicit*
- verify boundary between safe/unsafe code
- generate better API documentation
- enable better VS intellisense tooltips
- amplify testing via runtime checking
- enable static verification and bug finding