CallSharp: Automatic Input/Output Matching in .NET

Dmitri Nesteruk

@dnesteruk

The Problem

Given input and required output data, use “artificial intelligence” to determine the call, or chain of calls, that would lead to the output.

```
"abc" ⟶ "cba"
```

```
new string(input.Reverse().ToArray())
```

abc ⟶ ABC

“abc” is clearly a String
Strings are immutable (pure functions)
Look for any member function (property) that
    Is a member of String
    Has no arguments
    Returns a String
Very simple search

String-to-String Methods

`Normalize`
`ToLower`
`ToLowerInvariant`
`ToUpper`
`ToUpperInvariant`
`ToString`
`Trim`
`TrimStart`
`TrimEnd`

Functions that take no arguments

Functions where *all* arguments have default values

Functions that take a single argument of type `params[]`

abc ⟶ ABC

```
input.ToUpper()
input.ToUpperInvariant()
```

`abc` ⟶ 3

- A problem!
- What is “3”?
  - Definitely a string (*everything* is)
  - Parses as integral type…
  - Also a floating-point type
  - Even a TimeSpan!
- Let the user choose (assume Int32)
- Search for String member functions yielding Int32

abc ⟶ 3

```
input.Length
```

`abc` ⟶ `false`

- Output is either a bool or a String
  - Let's assume bool for the time being
- String has only one bool-returning function
  - `IsNormalized() = true`
- But String also has static functions, so…
- Search all static String-returning functions

abc ⟶ false

```
string.IsNullOrEmpty(input)
string.IsNullOrWhiteSpace(input)
```

$$\texttt{abc␣␣} \longrightarrow \texttt{ABC}$$

Try all single static/non-static calls… no luck

Try a call chain

$$\text{String} \xrightarrow{f()} \text{String} \xrightarrow{g()} \text{String}$$

$$\text{T} \to \text{U} \to \text{T}$$

$$\text{T} \to \text{U} \to \text{V} \to \text{T} \text{ etc.}$$

Also includes constructor calls

Complexity explosion

Beware of implicit conversions

abc␣␣ ⟶ ABC

```
string.Concat(input.Split()).ToUpper()
string.Concat(input.Split()).ToUpperInvariant()
input.ToUpper().Trim()
input.ToUpper().TrimEnd()
input.ToUpperInvariant().Trim()
input.ToUpperInvariant().TrimEnd()
input.Trim().ToUpper()
input.Trim().ToUpperInvariant()
input.TrimEnd().ToUpperInvariant()
input.TrimEnd().ToUpper() // + lots more solutions
```

aaabbb ⟶ aaa

- We need to trim away ‘b’
- ‘b’ is a substring of “aaabbb” (one of many)
- String can be decomposed into characters
  - These can be used on their own; or
  - Combinations lumped together to form strings
- Look for member/static functions which
  - Take a single object of some type that String can be decomposed into (string or char)
  - Take multiple parameters (ordinary or params[])

aaabbb ⟶ aaa

```
input.Trim('b')
input.TrimEnd('b')
```

cater ⟶ cat

```
input.Trim('e','r')
input.Trim('r','e')
input.Trim('a','e','r')
input.Trim('a','r','e')
input.Trim('e','a','r')
input.Trim('e','r','a')
input.Trim('r','a','e')
input.Trim('r','e','a')
input.TrimEnd('e','r')
input.TrimEnd('r','e')
// 30+ more options
```

Commutativity

Sufficiency

aabbcc ⟶ aacc

```
input.Replace("aabb", "aa")
input.Replace("bb", "")
input.Replace("bbcc", "cc")
```

Search all possible *pairs* of arguments

Middle option is least redundant

a␣b␣c ⟶ abc

```
input.Replace(" ", string.Empty)
input.Replace(" b ", "b")
input.Replace("a b ", "ab")
input.Replace(" b c", "bc")
input.Replace("a b c", "abc")
// at greater depth,
string.Concat(input.Split())
```

Only the first option is fundamentally correct

Performance

Reflection

- MethodInfo.Invoke() is *very* expensive
- Delegate.CreateDelegate() only viable for static methods
- Most investigated types are core CLR ones

Complexity explosion (near-infinite in certain cases)

Summary

Exhausive search for single non/static calls

Search for call chains

Argument combinations/permutations

Feature Improvements

Operator support (e.g., op[])
Sequence handling (array/list/IEnumerable)
Regular expression inference (e.g., Regex.Replace)
Other programming languages ☺

Performance Improvements

Static reflection for known types (T4)

Better search (annealing etc.)

Emulated evaluation of complex searches (GPU?)

Distributed/cloud back-end

Summary

CallSharp is open-source

You can help!

http://github.com/nesteruk/callsharp

Skype: dmitri.nesteruk

@dnesteruk