Decompile This: Cross-Platform ILSpy

@PammerSiegfried, @WilleChristoph

We choose to build a decompiler and do the other things, not because they are easy, but because they are hard.

J. Fitzgerald K., OSS Developer

The REAL History

- 4th February 2011 – official start

- Reason: RedGate's (soon regretted) decision to cease the free version of Reflector
  - https://web.archive.org/web/20110205074826/http://www.red-gate.com/products/dotnet-development/reflector/announcement

- Built on top of David Srbecky's master thesis (our SD "debugger guy")

https://github.com/icsharpcode/ILSpy/wiki/Release-History

ILSpy 3.0 Goal: Feature Parity

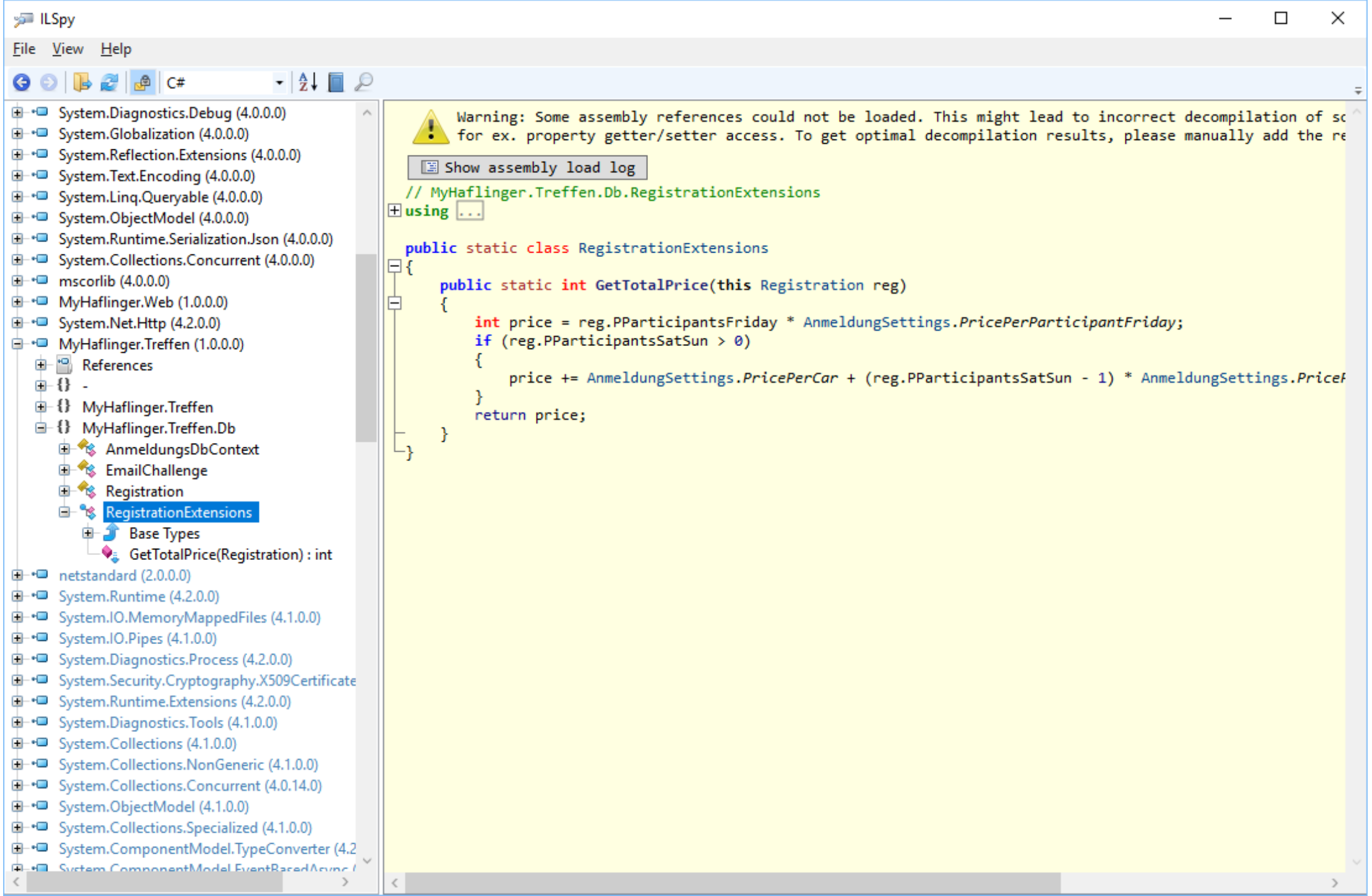
https://github.com/icsharpcode/ILSpy/wiki/newdecompiler (more later)

Features

- Decompilation of C# to C#
- Whole-project decompilation (csproj, not sln!)
- Search for types/methods/properties (substring)
- Hyperlink-based type/method/property navigation
- Base/Derived types navigation, history
- BAML to XAML decompiler
- Extensible via plugins (MEF)

829

Answer to the Ultimate Question of Life, The Universe, and Everything


ILSpy
File View Help
C#
System.Diagnostics.Debug (4.0.0.0)
System.Globalization (4.0.0.0)
System.Reflection.Extensions (4.0.0.0)
System.Text.Encoding (4.0.0.0)
System.Linq.Queryable (4.0.0.0)
System.ObjectModel (4.0.0.0)
System.Runtime.Serialization.Json (4.0.0.0)
System.Collections.Concurrent (4.0.0.0)
mscorlib (4.0.0.0)
MyHaflinger.Web (1.0.0.0)
System.Net.Http (4.2.0.0)
MyHaflinger.Treffen (1.0.0.0)
References
-
MyHaflinger.Treffen
MyHaflinger.Treffen.Db
AnmeldungsDbContext
EmailChallenge
Registration
RegistrationExtensions
Base Types
GetTotalPrice(Registration) : int
netstandard (2.0.0.0)
System.Runtime (4.2.0.0)
System.IO.MemoryMappedFiles (4.1.0.0)
System.IO.Pipes (4.1.0.0)
System.Diagnostics.Process (4.2.0.0)
System.Security.Cryptography.X509Certificate
System.Runtime.Extensions (4.2.0.0)
System.Diagnostics.Tools (4.1.0.0)
System.Collections (4.1.0.0)
System.Collections.NonGeneric (4.1.0.0)
System.Collections.Concurrent (4.0.14.0)
System.ObjectModel (4.1.0.0)
System.Collections.Specialized (4.1.0.0)
System.ComponentModel.TypeConverter (4.2
Warning: Some assembly references could not be loaded. This might lead to incorrect decompilation of sc
for ex. property getter/setter access. To get optimal decompilation results, please manually add the re
Show assembly load log
// MyHaflinger.Treffen.Db.RegistrationExtensions
using ...
public static class RegistrationExtensions
{
public static int GetTotalPrice(this Registration reg)
{
int price = reg.PParticipantsFriday * AnmeldungSettings.PricePerParticipantFriday;
if (reg.PParticipantsSatSun > 0)
{
price += AnmeldungSettings.PricePerCar + (reg.PParticipantsSatSun - 1) * AnmeldungSettings.PriceF
}
return price;
}
}

Decompiler Architecture

- Metadata
  - Mono.Cecil
  - Decompiler.TypeSystem

- Decompilation Pipeline
  1. ILReader
  2. ILAst Transforms
  3. StatementBuilder/ExpressionBuilder/CallBuilder
  4. C# Transforms

Metadata

- Mono.Cecil
  - Low-level representation of metadata
  - Holds all information about types and members encoded in the PE File
  - Wraps access to IL instructions

- Decompiler.TypeSystem
  - High-level representation of metadata
  - Language-specific (C#)
    - Type Conversions
    - Overload Resolution

Pipeline Step #1: ILReader

- Transform stack-based IL assembly into statement-based ILAst
- Infer types of “stack-slots”
- Detect basic blocks by analyzing branch instructions
- Make implicit conversions explicit where possible

Pipeline Step #2: ILAst / ILAst Transforms

- Annotated and structured representation of IL instructions
  - Tree structure
  - Blocks, Statements, Expressions
- Every instruction has an explicit result type
- ILAst Transforms modify the structure, but not the semantics of the code, e.g.
  - Transform conditional branches into IfInstructions
  - Inline blocks

Pipeline Step #3: Build C# code

- StatementBuilder
  - Generate C# statements
- ExpressionBuilder
  - Translate to C# expressions
  - Convert Stack type to C# type
  - Eliminate conversions where possible
- CallBuilder
  - Translate calls to property access, method calls
  - Ensure correct method is called → overload resolution

Pipeline Step #4: C# Transforms

- “Beautify” the generated code
- Replace method calls with operators
- Introduce using declarations
- Introduce extension method syntax
- Build LINQ queries

Demo

From slides to reality

Architecure Redesign (newdecompiler)

- ILAst instructions now carry an IL stack type => bit width known
- ILAst instructions no longer use C# types, introduced only ILAst->C#
- ILAst instruction set is reduced, now has clear semantics for each operation
- ILAst uses (extended) basic blocks for control flow
- Many transforms now are run per-block, which solves a number of problems related to transform pass ordering.
- In debug builds, transform steps can be tracked via the Stepper API to visualize the decompilation process
- Instead of Cecil.TypeReference, decompiler now uses the NR TypeSystem
- The required portions of NRefactory are included into ICS.Decompiler

Going x-plat

“.NET Standard is a specification”

Standards are great, there are so many to choose from

Give us this day our daily MSBuild problem

“It works on my machine”

AppVeyor, msbuild and devenv.exe

ILSpy Team
@ilspy

Following

How hard can it possibly get? @dotnet standard (tooling) takes the cake github.com /icsharpcode/IL ... #ILSpy

Retarget ICS.Decompiler to .NET Standard 2.0 · Issue #831 ·…

Please see https://docs.microsoft.com/en-us/dotnet/standard/net-standard#net-implementation-support Minimum Framework compat will be 4.6.1 and Core 2.0. Dev dependency: VS2017 ...

github.com

7:47 AM - 31 Aug 2017

The Journey in Pull Requests

- Retargetting post mortem for PR Experimental dotnet #833
  - More Details in PR 832
- Update to the new project system #836 Sam Harwell
  - The solution (do not port WPF to the new project system...)
- Our main issues
  - A multi-targeting source code dependency (Cecil)
  - A desire to target 4.6.2 (today we target net46 & netstandard 2.0 – find the mistake!)

Conditional Project References

```
<ItemGroup Condition="'$(Configuration)' == 'Debug'">
  <ProjectReference Include="..\cecil\Mono.Cecil.csproj" AdditionalProperties="NuGetRestoreTargets=;ResolveNuGetPackages=false" />
  <ProjectReference Include="..\ICSharpCode.Decompiler\ICSharpCode.Decompiler.csproj" />
</ItemGroup>

<ItemGroup Condition="'$(Configuration)' == 'Release'">
  <PackageReference Include="ICSharpCode.Decompiler" Version="3.0.0-beta1" />
</ItemGroup>
```

https://github.com/icsharpcode/ILSpy/pull/836/files#diff-37a82c6ae3b55ce5d833adbd171c918eR42

.NET Core DLL Hell

AutoGenerateBindingRedirects (PR 836, again)

- https://stackify.com/net-core-dll-hell/
- https://stackoverflow.com/questions/39252136/how-can-i-add-an-assembly-binding-redirect-to-a-net-core-unit-test-project
- https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/how-to-enable-and-disable-automatic-binding-redirection

error NU1605: Detected package downgrade

- The Problem
  - “System.IO.FileSystem.Primitives from 4.3.0 to 4.0.1. Reference the package directly from the project to select a different version.” + 3 more

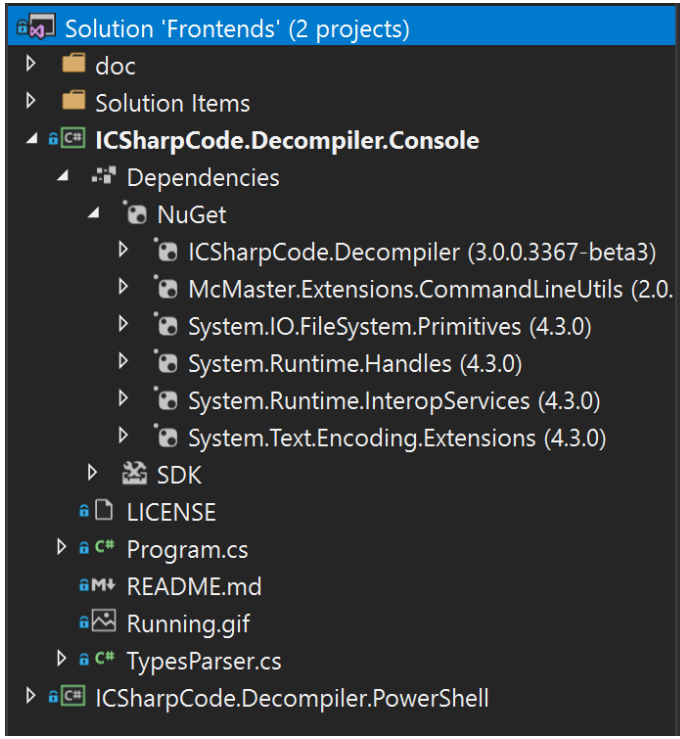
```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp2.0</TargetFramework>
    <RuntimeIdentifiers>win10-x64;osx-x64;linux-x64</RuntimeIdentifiers>
    <AssemblyName>ilspycmd</AssemblyName>
  </PropertyGroup>

  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|AnyCPU'">
    <TreatWarningsAsErrors>false</TreatWarningsAsErrors>
    <WarningsAsErrors>NU1605</WarningsAsErrors>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="McMaster.Extensions.CommandLineUtils" Version="2.0.1" />
    <PackageReference Include="ICSharpCode.Decompiler" Version="3.0.0.3403-beta4" />

    <PackageReference Include="System.IO.FileSystem.Primitives" Version="4.3.0" />
    <PackageReference Include="System.Runtime.Handles" Version="4.3.0" />
```



Embed Source / x-plat PDBs

- The Problem
  - Windows PDBs do not contain source code
- The Solution
  - https://github.com/icsharpcode/ILSpy/issues/943#issuecomment-339914287
  - Started working in 15.5 (of course we did it with a Preview [2])
  - csc can do it, msbuild can’t -> another workaround
- The Surprise
  - Our NuGet got smaller!

x-plat Debugging

- The Problem
  - Debugging issues in *nix (#975)
- The Solution
  - https://github.com/Microsoft/MIEngine/wiki/Offroad-Debugging-of-.NET-Core-on-Linux---OSX-from-Visual-Studio
  - But Visual Studio Code turned out to be more effective...
- The Surprise
  - VS Debugger != dotnet run != dotnet publish

EnC – Edit and Continue

- Multi-TFM projects cannot use EnC
- Scheduled for 15.6 – at least at the moment

https://github.com/dotnet/roslyn/issues/21170

Copying NuGets to Local

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netstandard2.0</TargetFramework>
    <CopyLocalLockFileAssemblies>true</CopyLocalLockFileAssemblies>
    <RootNamespace>ICSharpCode.Decompiler.PowerShell</RootNamespace>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="PowerShellStandard.Library" Version="3.0.0-preview-01" />
    <PackageReference Include="ICSharpCode.Decompiler" Version="3.0.0.3403-beta4" />
  </ItemGroup>

</Project>

```

<asmname>.deps.json

“If you’re not making mistakes, then you’re not doing anything”

CSharpLanguage.cs

aka “Poisoning the well”, or “API Design”

The API Journey

- Dead End: “RFC: Simple API NuGet”

- The Solution
  - Build a couple clients yourself
    - Xamarin Workbook (https://github.com/Microsoft/workbooks)
    - Console App (added bonus: getting rid of command line switches in ILSpy)
    - PowerShell cmdlets

  - Ask for feedback: VSCode Extension had used CSharpLanguage.cs

PowerShell Usage

```powershell
$basePath = $PSScriptRoot
if ([string]::IsNullOrEmpty($basePath))
{
    $basePath = Split-Path -parent $psISE.CurrentFile.Fullpath
}

$modulePath = $basePath + '\bin\Debug\netstandard2.0\ICSharpCode.Decompiler.Powershell.dll'

Import-Module $modulePath
$decompiler = Get-Decompiler $modulePath

$classes = Get-DecompiledTypes $decompiler -Types class
$classes.Count

foreach ($c in $classes)
{
    Write-Output $c.FullName
}


Get-DecompiledSource $decompiler -TypeName ICSharpCode.Decompiler.PowerShell.GetDecompilerCmdlet

Get-DecompiledProject $decompiler -OutputPath .\decomptest
```

https://github.com/christophwille/ilspy-pscore/issues/1 Mac "Bug"

Finding .NET Assemblies on *nix

- UniversalAssemblyResolver
  - Detect target framework (attribute) of assembly
  - Supports .NET Framework, Mono, .NET Core, .NET Standard

- DotNetCorePathFinder
  - Parsing *.deps.json if available
  - Finding dotnet(.exe) in $PATH
  - Finding /dotnet/shared/ directory

AppVeyor NuGet feed

- Turns out having the wrong fallback (version) can be bad
- Do not push PRs...

```
9   nuget:
10    account_feed: false
11    project_feed: true
12    disable_publish_on_pr: true
```

Versioning ICS.Decompiler

- update-assemblyinfo.ps1
  - See its history to learn from our x-plat woes

- Why not https://github.com/GitTools/GitVersion or https://github.com/AArnott/Nerdbank.GitVersioning
  - Doesn’t cut it for us, but just might for you

Things you never wanted to know (or cared)

- When an int is not an int
  - https://github.com/dotnet/coreclr/issues/14492
- .NET Core *is* different
  - https://github.com/dotnet/coreclr/issues/14784

A Bridge We Won’t Cross

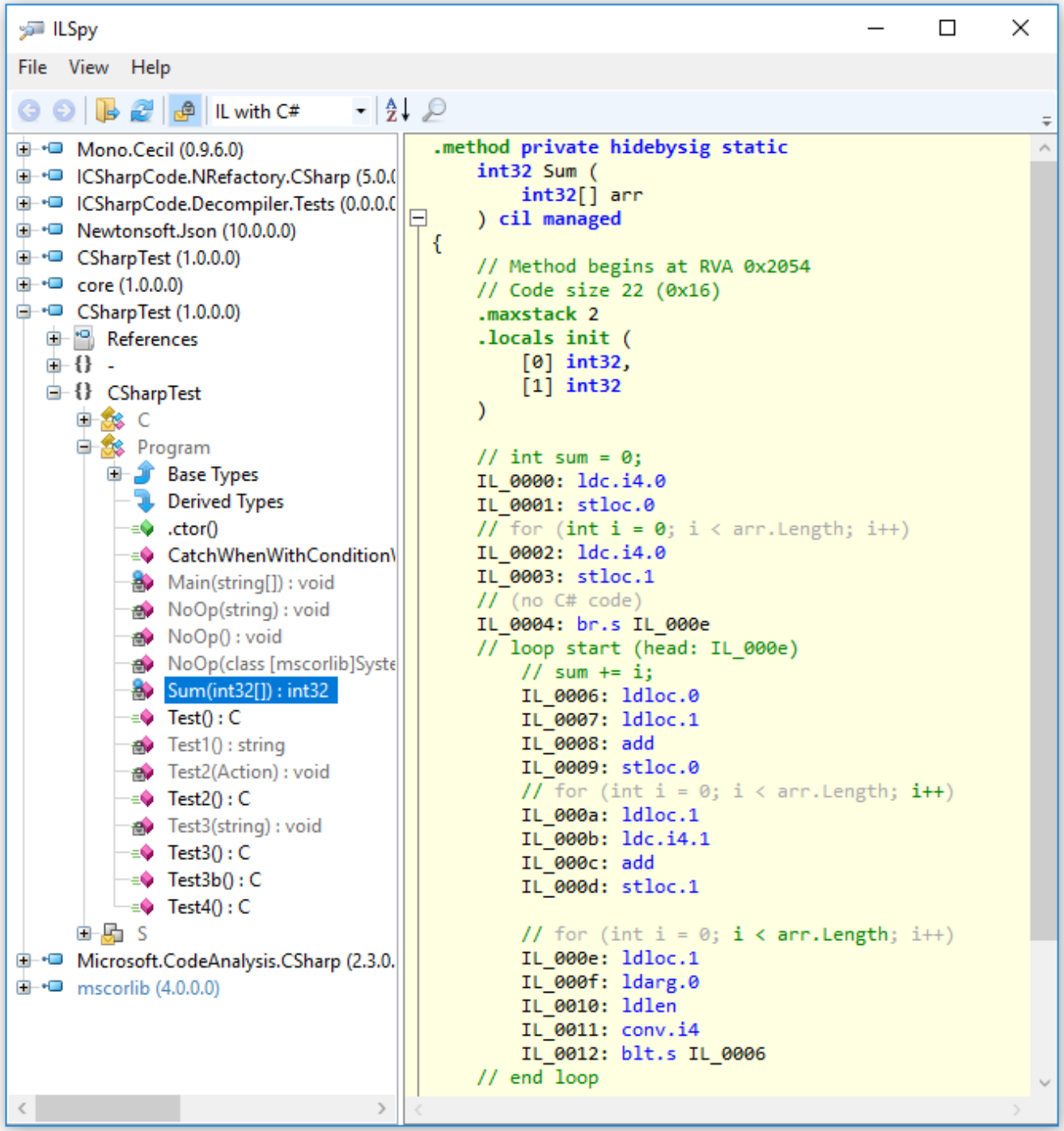
- Xamarin Forms, Avalonia, ... your choice of x-plat UI fx

- XAML Standard is possibly shaking up the landscape
- Besides: WPF works (even on *nix)

The Future

IL with C# View

- A precursor of things to come
- Map IL instructions to C# statements
- Sequence points introduced



Plans (can and will change)

- PDB generation

- Language Features
  - ref returns
  - await in catch/finally
  - Elvis
  - String Interpolation
  - async Main
  - out variables

- https://github.com/dotnet/roslyn/pull/23430

Roslyn

The elephant in the room 896

Conclusion

- What a (fun) ride – depending on your definition of fun

- Takeaways for “Joe Average”
  - Read, read again, bookmark https://github.com/dotnet/standard/issues/481
  - Move to .NET 4.7.1 when using .NET Standard NuGets