NonNullStack Static Checking Example

Abstract

This example shows object invariants checking involving ForAll over the elements of the array, and caching of the analysis results to avoid re-analysis. The example is a simple stack containing only non-null elements.

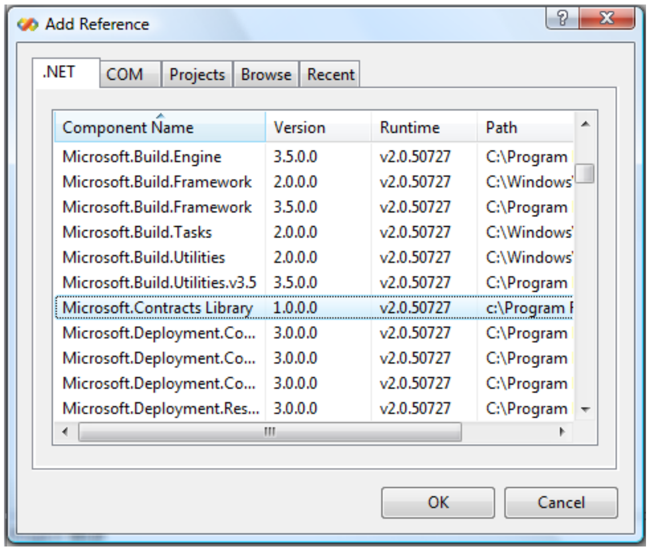
1 Adding the Contract Library Reference

If you are using Visual Studio 2008, or if you for some reason want to target a pre-v4 .NET runtime, then you need to:

- Change the target framework of the project.
- Manually add a reference to Microsoft.Contracts.dll

Otherwise, you may skip this section and go directly the next section!

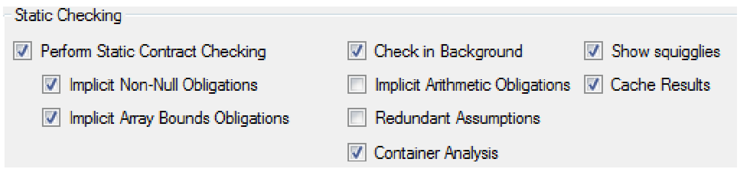
To add the reference, open the NonNullStack solution and right-click on References in the NonNullStack project and select Add Reference. Find the Microsoft.Contracts library in the .NET tab as shown below and click OK.



Component Name	Version	Runtime	Path
Microsoft.Build.Engine	3.5.0.0	v2.0.50727	C:\Program
Microsoft.Build.Framework	2.0.0.0	v2.0.50727	C:\Windows
Microsoft.Build.Framework	3.5.0.0	v2.0.50727	C:\Program
Microsoft.Build.Tasks	2.0.0.0	v2.0.50727	C:\Windows
Microsoft.Build.Utilities	2.0.0.0	v2.0.50727	C:\Windows
Microsoft.Build.Utilities.v3.5	3.5.0.0	v2.0.50727	C:\Program
Microsoft.Contracts Library	1.0.0.0	v2.0.50727	c:\Program F
Microsoft.Deployment.Co...	3.0.0.0	v2.0.50727	C:\Program
Microsoft.Deployment.Co...	3.0.0.0	v2.0.50727	C:\Program
Microsoft.Deployment.Co...	3.0.0.0	v2.0.50727	C:\Program
Microsoft.Deployment.Res...	3.0.0.0	v2.0.50727	C:\Program

2 Sample Walkthrough

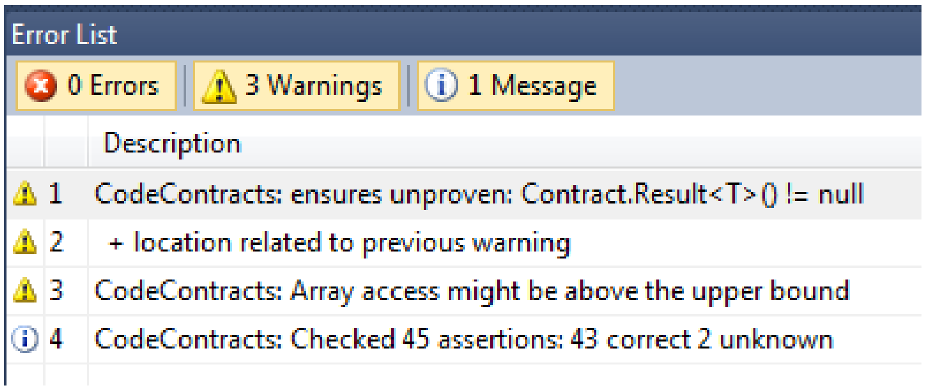
After adding the proper reference, go to the Properties of project NonNullStack, select the Code Contracts pane (at the bottom), and enable static checking by clicking on the static checking box. Also enable implicit non-null, array checks, container analysis and caching as shown in this screenshot:



Static Checking

[✓] Perform Static Contract Checking	[✓] Check in Background	[✓] Show squigglies
[✓] Implicit Non-Null Obligations	[ ] Implicit Arithmetic Obligations	[✓] Cache Results
[✓] Implicit Array Bounds Obligations	[ ] Redundant Assumptions	
	[✓] Container Analysis	

Then build the example. The build should succeed. After a moment[1], the static checker should warn about the following problems:



Error List

0 Errors | 3 Warnings | 1 Message

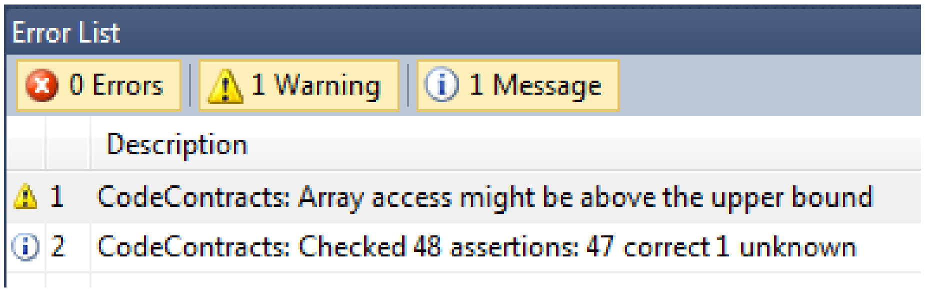
		Description
⚠	1	CodeContracts: ensures unproven: Contract.Result<T>() != null
⚠	2	+ location related to previous warning
⚠	3	CodeContracts: Array access might be above the upper bound
ⓘ	4	CodeContracts: Checked 45 assertions: 43 correct 2 unknown

The first warning is orginated by the fact that the static checker is missing the information that all the array elements of indexes 0 . . . index are not null. We can make this information explicit by adding the following object invariant (in the example, just uncomment it):

```
Contract.Invariant(Contract.ForAll(0, nextFree, i => arr[i] != null));
```

If we build again, we notice that the number of checked assertions as increased. This is due to the fact that now also the condition on the elements of arr is checked at method exits.

[1]The static checker runs in the background after the regular build.



Error List

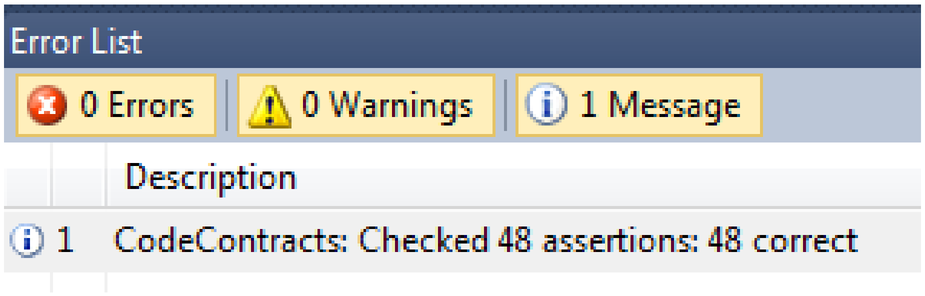
0 Errors | 1 Warning | 1 Message

		Description
⚠	1	CodeContracts: Array access might be above the upper bound
ⓘ	2	CodeContracts: Checked 48 assertions: 47 correct 1 unknown

We are left with the last message from the static checker, warning us that the array access may be not be in bounds. This happens when arr.Length is zero. Changing the array creation expression as

```
var newArr = new T[arr.Length * 2 + 1];
```

solves the problem:



Error List

0 Errors | 0 Warnings | 1 Message

		Description
ⓘ	1	CodeContracts: Checked 48 assertions: 48 correct

Push was the only modified method, so the checker used the cache to avoid re-analyzing the unmodified methods (Properties are not cached):

```
CodeContracts: NonNullStack: Total methods analyzed 6
CodeContracts: NonNullStack: Total method analysis read from the cache 3
```