Resource Usage Contracts for .NET

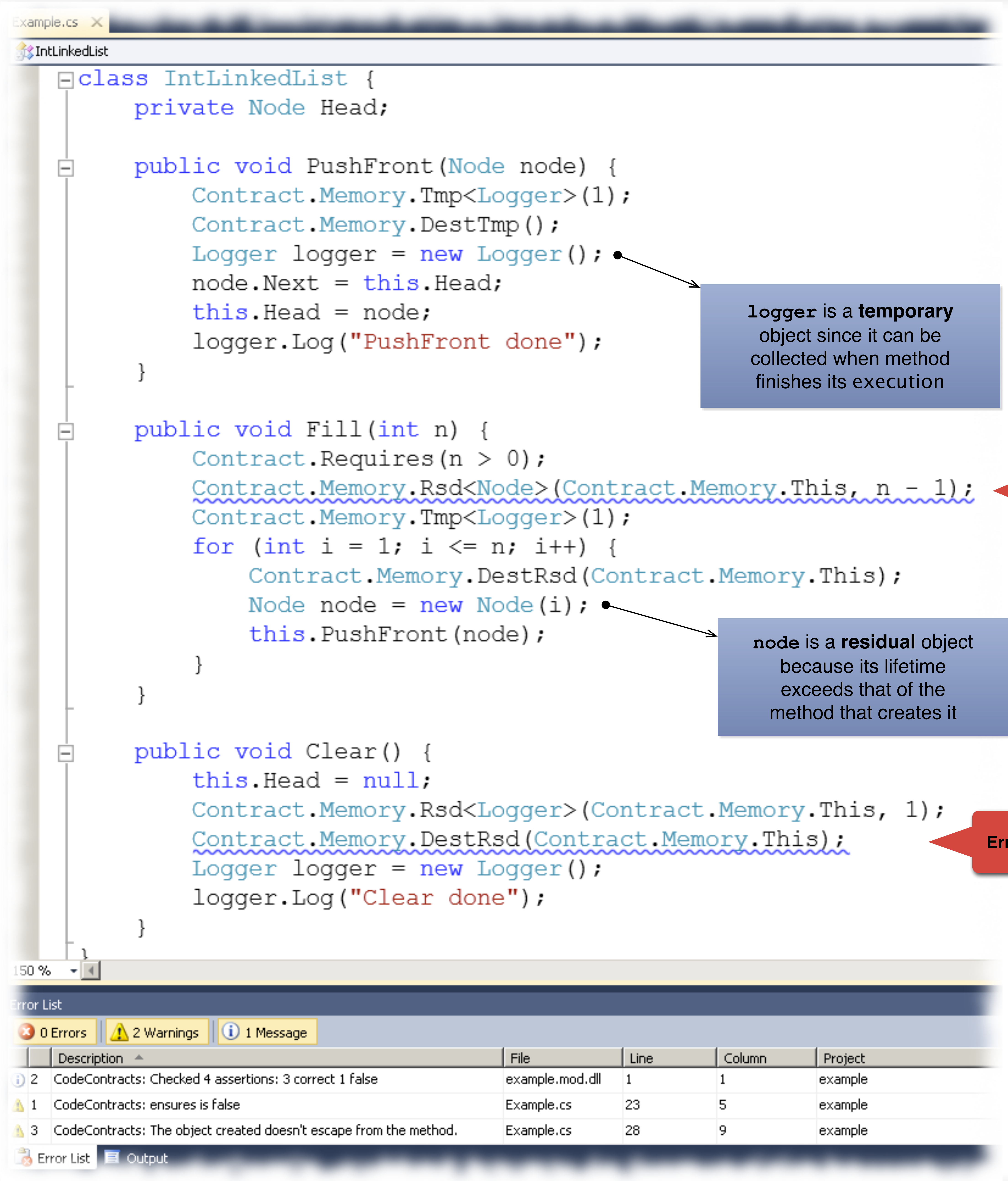
Jonathan Tapicer, Diego Garbervetsky and Martin Rouaux - {jtapicer, diegog, mrouaux}@dc.uba.ar
Departamento de Computación, FCEyN, Universidad de Buenos Aires, Argentina

What?

- An extension of Code Contracts to support resource usage specifications in .NET programs.
- Tailored for specifying dynamic memory consumption, a resource that is not only allocated but it is also reclaimed during program execution.

How?

- We introduce new set of annotations enabling specification of both memory consumption and lifetime properties in a modular fashion.
- These annotations allow us to compute an upper bound of the real memory allocated using a compositional analysis.



```
class IntLinkedList {
    private Node Head;

    public void PushFront(Node node) {
        Contract.Memory.Tmp<Logger>(1);
        Contract.Memory.DestTmp();
        Logger logger = new Logger();
        node.Next = this.Head;
        this.Head = node;
        logger.Log("PushFront done");
    }

    public void Fill(int n) {
        Contract.Requires(n > 0);
        Contract.Memory.Rsd<Node>(Contract.Memory.This, n - 1);
        Contract.Memory.Tmp<Logger>(1);
        for (int i = 1; i <= n; i++) {
            Contract.Memory.DestRsd(Contract.Memory.This);
            Node node = new Node(i);
            this.PushFront(node);
        }
    }

    public void Clear() {
        this.Head = null;
        Contract.Memory.Rsd<Logger>(Contract.Memory.This, 1);
        Contract.Memory.DestRsd(Contract.Memory.This);
        Logger logger = new Logger();
        logger.Log("Clear done");
    }
}
```

```
150 %
Error List
0 Errors | 2 Warnings | 1 Message
```

	Description	File	Line	Column	Project
2	CodeContracts: Checked 4 assertions: 3 correct 1 false	example.mod.dll	1	1	example
1	CodeContracts: ensures is false	Example.cs	23	5	example
3	CodeContracts: The object created doesn't escape from the method.	Example.cs	28	9	example

```
Error List | Output
```

`logger` is a **temporary** object since it can be collected when method finishes its execution

`node` is a **residual** object because its lifetime exceeds that of the method that creates it

Error: the method requires `n` residual `Node` objects

Error: the object `logger` is temporary but marked as residual

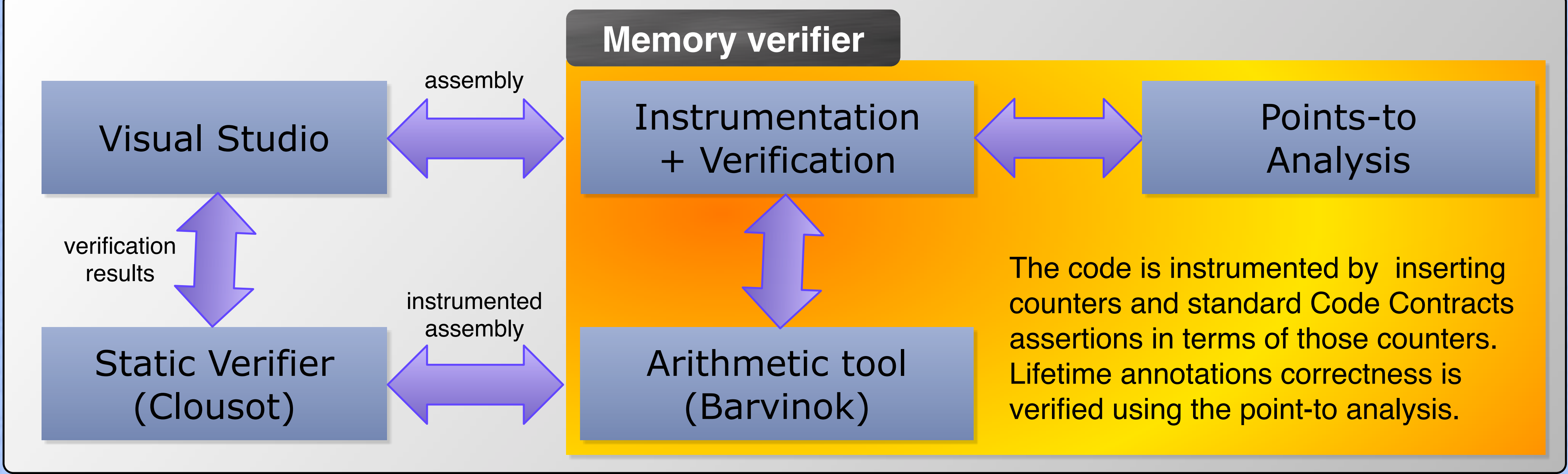
`Contract.Memory.DestTmp();` indicates that the next object created is **temporary**

`Contract.Memory.Tmp<T>(n);` defines a **temporary** memory of type `T` of at most `n`

`Contract.Memory.DestRsd(t);` indicates that the next object created is **residual** and tagged as `t`

`Contract.Memory.Rsd<T>(t, n);` defines a **residual** memory of type `T` tagged as `t` of at most `n`

How are the annotations checked?



The code is instrumented by inserting counters and standard Code Contracts assertions in terms of those counters. Lifetime annotations correctness is verified using the point-to analysis.

Future work

- Automatic inference of quantitative and lifetime annotations in order to mitigate annotation burden.
- Upgrade the language in order to enable finer grained lifetime specs. while maintaining information hiding.
- Use SMT solvers (e.g Z3) and integrate them with tools capable of dealing with non-linear expressions.

http://lafhis.dc.uba.ar/resourcecontracts