

PHP Generators in .NET

Compilation of a dynamic language Generators into MSIL



Petr Houška | Mgr. Jakub Míšek | Department of Software engineering | github.com/peachpiecompiler/peachpie | github.com/petrroll/bachelor-thesis

Goals

- Generators support within Peachpie
- Keeping reference PHP semantics
- Reusing already existing facilities
- Both design and implementation

Peachpie

- Bridge between PHP and .NET
- PHP to MSIL (CIL) compiler
- Reimplementation of PHP class library
- Written in C#, .NET foundation member

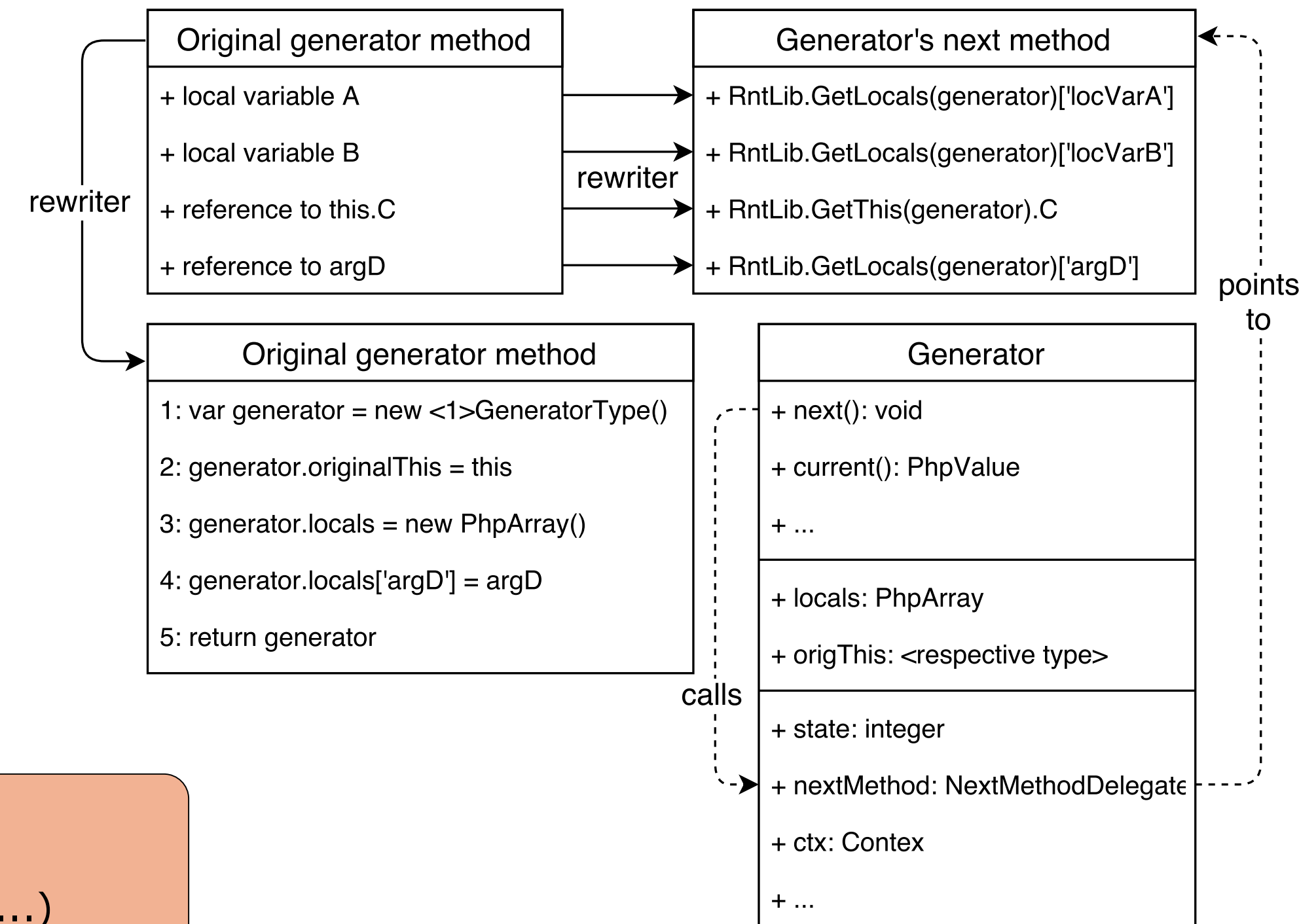
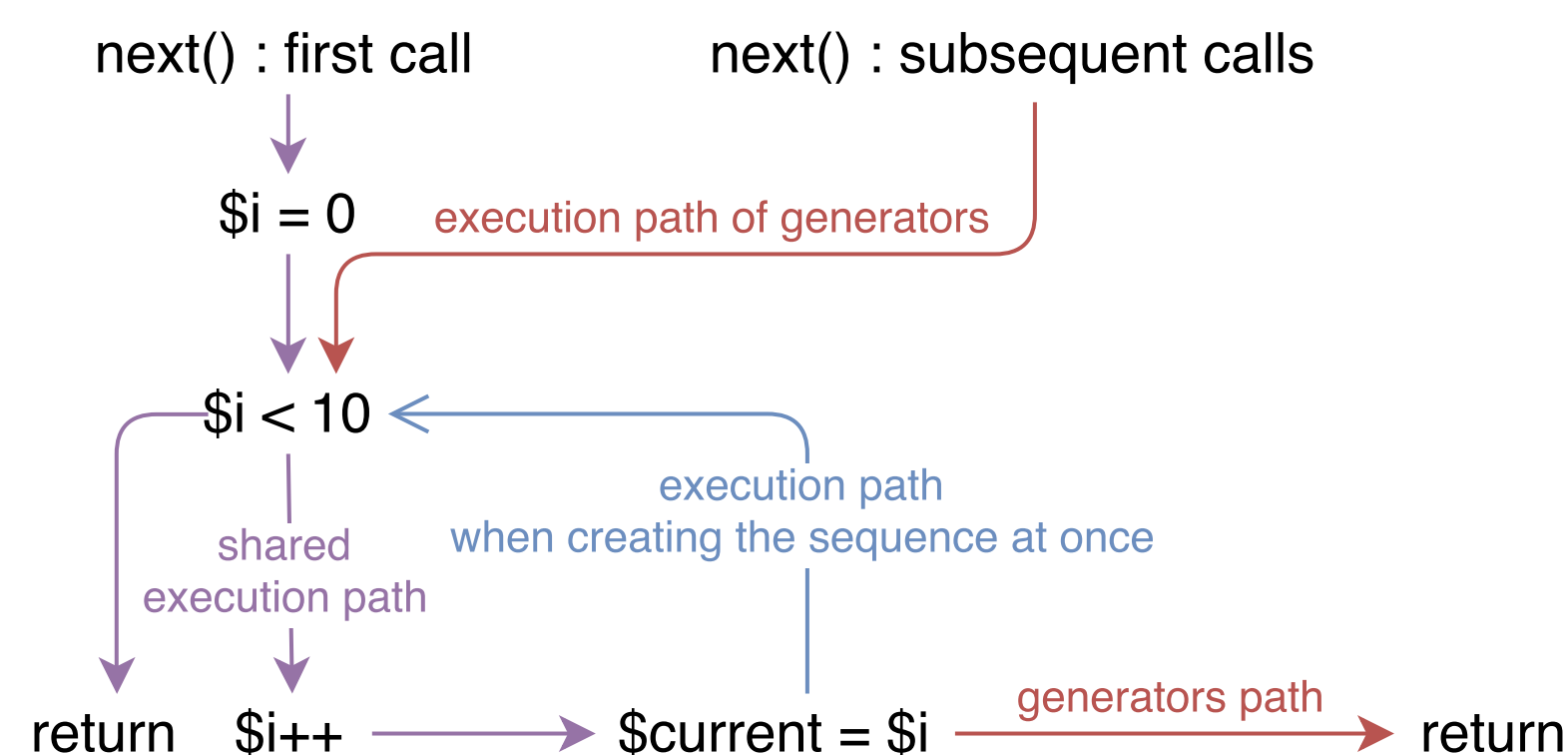
MSIL

- Intermediate language for .NET platform (C#, F#, ...)
- High level, stack based, object oriented assembler
- Native understanding of exception handling
- No concept of generators or execution state saving

Generators

- Contain a sequential algorithm, return an Iterator
- Natively supported by the reference PHP runtime (Zend)
- Lowered by compilers in .NET languages (e.g. Roslyn)
- Execution pauses at *yield* keyword, continues on *next()* call
- Statement in most .NET languages (C#, F#)
- Value carrying expression in PHP

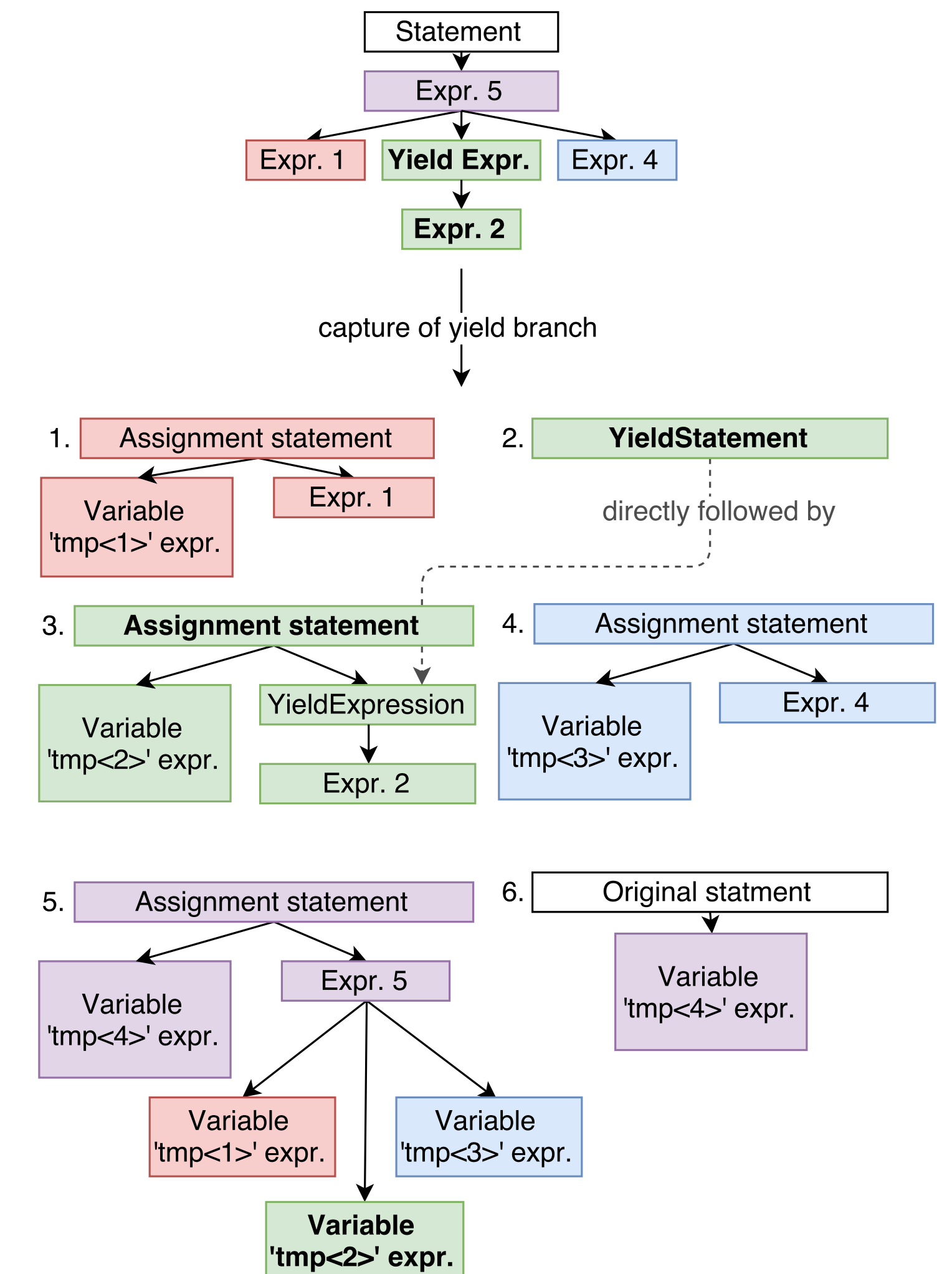
Generator method



Solution

- New Generator type representing the returned Iterator
- Changes to the original generator method's body:
 - Transformed to Generator's state machine as a new method
 - Replaced with code that initiates and returns the Generator
- State machine transformations:
 - Local variables lifted to Generator instance
 - Explicit state saving on each yield
 - State backed jump table in the beginning
- Semantic tree transformation to lower yields:
 - Moves yields under expression trees' roots
 - Handles conditioned branches with yields
 - Maintains the original order of execution
 - Splitting yield into a statement and an expression

Semantic tree transformation



Generator method

```
function by_one_generator(){
    for($i = 0; $i < 10; $i++){
        yield $i;
    }
}
```

Yield splitting

