

Eco: A language composition editor



Edd Barrett



Carl
Friedrich
Bolz



Lukas
Diekmann



Laurence
Tratt



Naveneetha
Krishnan
Vasudevan

KING'S
College
LONDON

Software Development Team
2014-05-19

Our problem

We want **better** programming languages

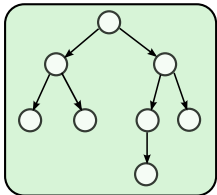
We want **better** programming languages

But better always seems to end up **bigger**

Language composition



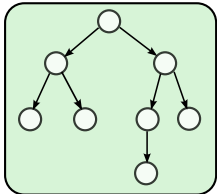
Parsing



Running

```
SUB    AX,AX
MOV    ES,AX
SUB    BH,BH
MOV    BL,INT_NUMBER
SHL    BX,1
SHL    BX,1
MOV    DI,ES:[BX]
MOV    ES,ES:[BX+2]
ADD    DI,4
LEA    SI,TAG
MOV    CX,TAG_LEN
```

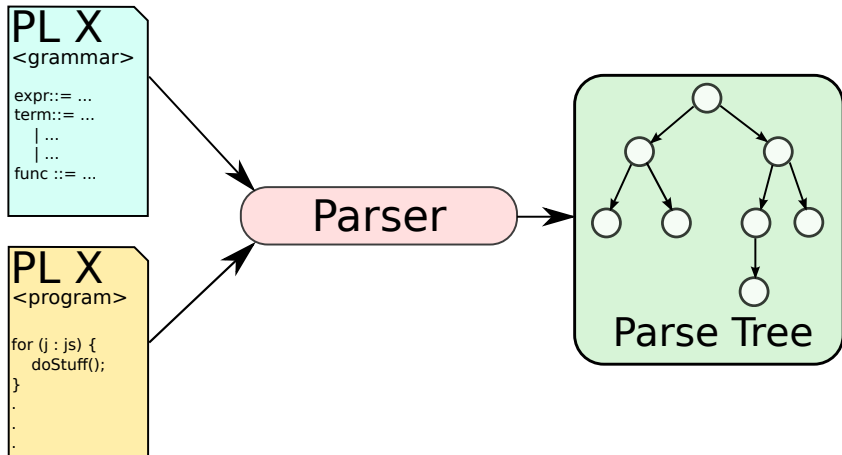
Parsing



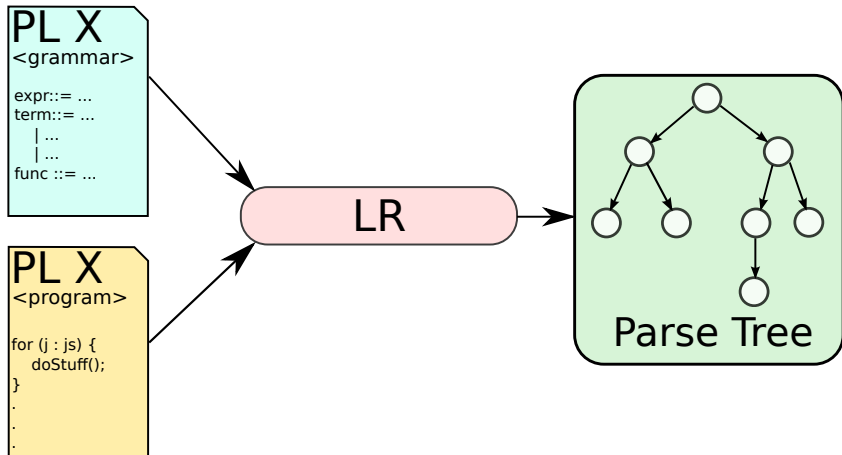
Running

```
SUB    AX, AX
MOV    ES, AX
SUB    BH, BH
MOV    BL, INT_NUMBER
SHL    BX, 1
SHL    BX, 1
MOV    DI, ES:[BX]
MOV    ES, ES:[BX+2]
ADD    DI, 4
LEA   SI, TAG
MOV    CX, TAG_LEN
```

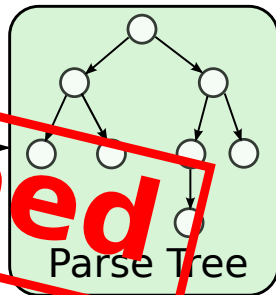
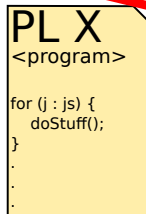
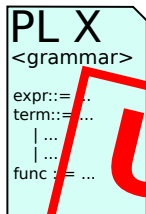
Parsing composition



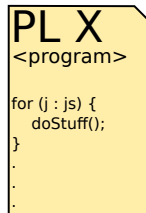
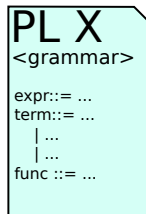
Parsing composition



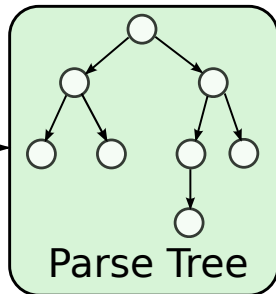
Parsing composition



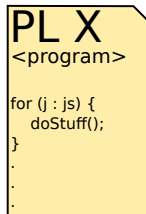
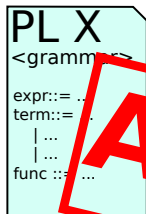
Parsing composition



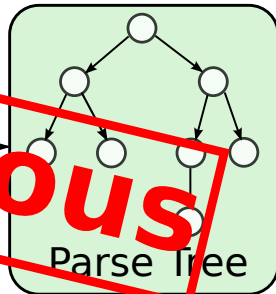
Generalised



Parsing composition

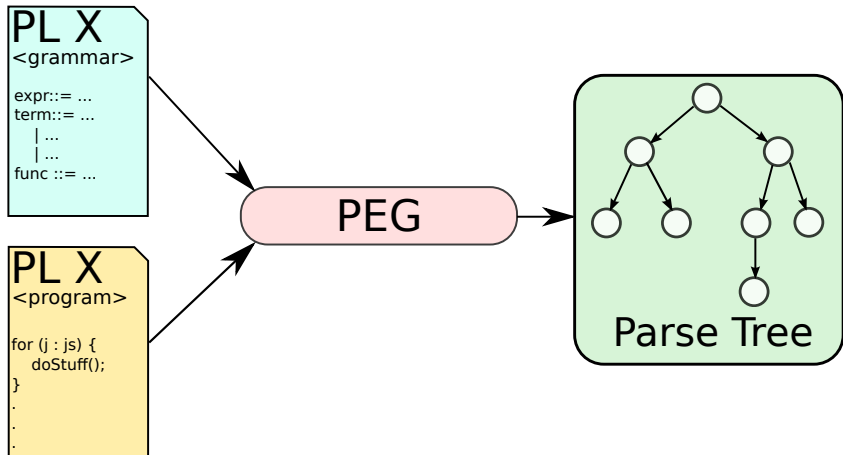


Generaliser

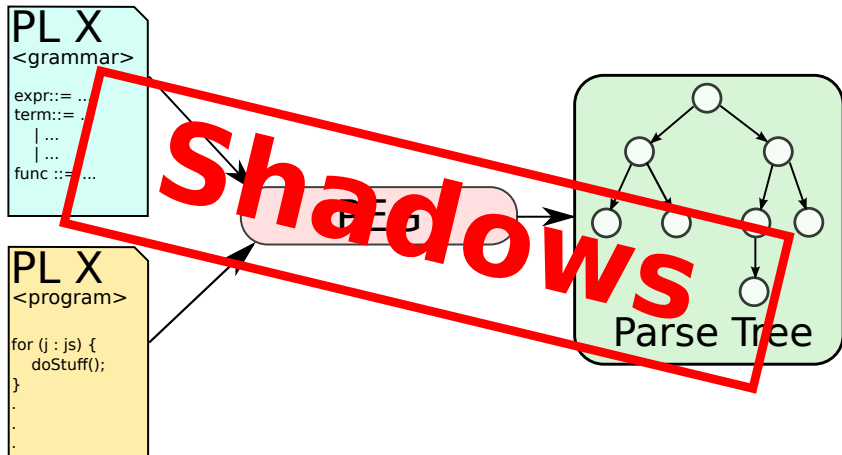


Ambiguous

Parsing composition



Parsing composition



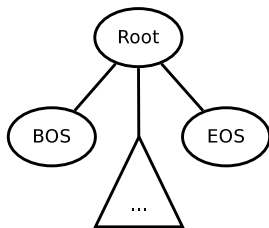
Syntax-directed editing (SDE)

Challenge:
SDE's power +
a text editor feel?

Demo

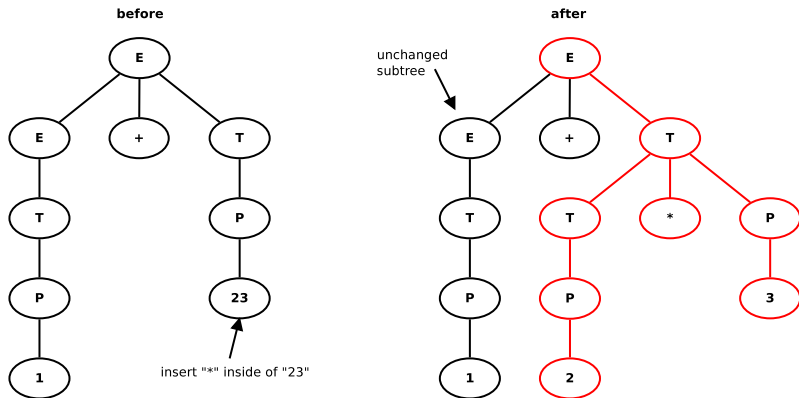
Incremental parser

The previous parse tree acts as input for the incremental parser



Incremental parser

The incremental parser reuses unchanged subtrees from the previous tree to speed up parsing



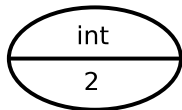
Demo

Language boxes

Property of context-free grammars:

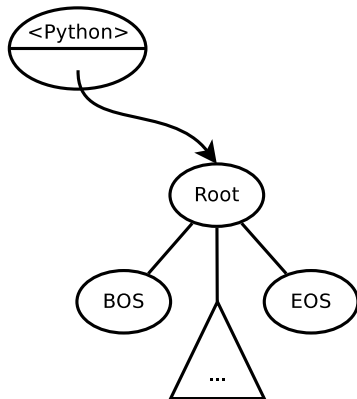
- Tokens consist of a type and a value.
- The parser only cares about the type.

Normal token



Language boxes

We can choose any type/value combination we want.



Demo

The traditional way to parse indentation based languages is too slow.

Indentation

```
def calc_indent1(l):
    if prev(l) == None:
        l.indent1 = 0
    elif prev(l).wsl == l.wsl:
        l.indent1 = prev(l).indent1
    elif prev(l).wsl < l.wsl:
        l.indent1 = prev(l).indent1 + 1
    else:
        assert prev(l).wsl > l.wsl
        prevl = prev(prev(l))
        while prevl != None:
            if prevl.wsl == l.wsl:
                l.indent1 = prevl.indent1
                return
            elif prevl.wsl < l.wsl:
                break
            prevl = prev(prevl)
        mark_unbalanced(l)
```

Indentation

```
def calc_indent1(l):
    if prev(l) == None:
        l.indent1 = 0
    elif prev(l).wsl == l.wsl:
        l.indent1 = prev(l).indent1
    elif prev(l).wsl < l.wsl:
        l.indent1 = prev(l).indent1 + 1
    else:
        assert prev(l).wsl > l.wsl
        prevl = prev(prev(l))
        while prevl != None:
            if prevl.wsl == l.wsl:
                l.indent1 = prevl.indent1
                return
            elif prevl.wsl < l.wsl:
                break
            prevl = prev(prevl)
        mark_unbalanced(l)
```

- Each line knows its own indentation level

Indentation

```
def calc_indent1(l):
    if prev(l) == None:
        l.indent1 = 0
    elif prev(l).wsl == l.wsl:
        l.indent1 = prev(l).indent1
    elif prev(l).wsl < l.wsl:
        l.indent1 = prev(l).indent1 + 1
    else:
        assert prev(l).wsl > l.wsl
        prevl = prev(prev(l))
        while prevl != None:
            if prevl.wsl == l.wsl:
                l.indent1 = prevl.indent1
                return
            elif prevl.wsl < l.wsl:
                break
            prevl = prev(prevl)
        mark_unbalanced(l)
```

- Each line knows its own indentation level
- Editing whitespace of a line recalculates its indentation level by looking at previous lines

Indentation

```
def calc_indent1(l):
    if prev(l) == None:
        l.indent1 = 0
    elif prev(l).wsl == l.wsl:
        l.indent1 = prev(l).indent1
    elif prev(l).wsl < l.wsl:
        l.indent1 = prev(l).indent1 + 1
    else:
        assert prev(l).wsl > l.wsl
        prevl = prev(prev(l))
        while prevl != None:
            if prevl.wsl == l.wsl:
                l.indent1 = prevl.indent1
                return
            elif prevl.wsl < l.wsl:
                break
            prevl = prev(prevl)
        mark_unbalanced(l)
```

- Each line knows its own indentation level
- Editing whitespace of a line recalculates its indentation level by looking at previous lines
- Afterwards, affected succeeding lines need to be updated as well

Incremental ASTs

Namebinding

Non-textual languages

What is it good for?

Unipycation

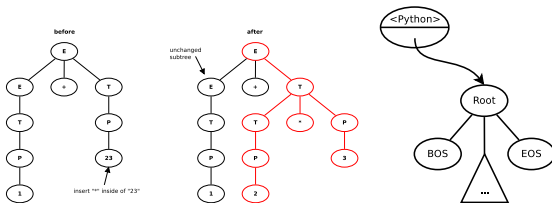
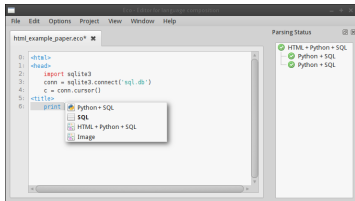
What is it good for?

Gradual migration



Incremental semantic analysis on ASTs
and code generation

Thanks for listening



<http://soft-dev.org/>