Default Disambiguation in Online Parsers

SLE 2019, Athens

Lukas
Diekmann

Laurence
Tratt

KING'S College LONDON

Software Development Team

2019-10-21

Language composition

Useful for domain specific languages and code migration.


AMBIGUITY

Delimiter-based approaches

```
for (String e:  «SELECT name FROM table») { ...  }
```

Pro

- Easy to implement

Con

- Visually intrusive
- Prevents use of delimiters in embedded language

Parses all CFGs (incl. ambiguous ones)

Pro

- No need for delimiters

Con

- Requires disambiguation operators
- Ambiguity generally undecidable

Edit ASTs directly; no parsing required.

Pro

- No ambiguity
- No delimiters

Con

- Difficult to use
- Steep learning curve

Embed languages using “language boxes”

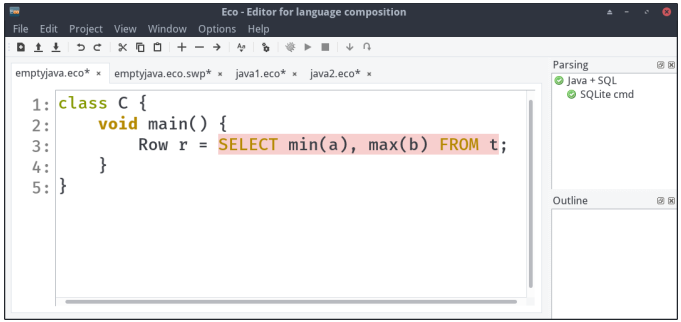
Pro

- Invisible delimiters
- Doesn’t restrict embedded language

Con

- Must be inserted manually

Demo


Eco - Editor for language composition
File Edit Project View Window Options Help
emptyjava.eco* ×
emptyjava.eco.swp* ×
java1.eco* ×
java2.eco* ×
1: class C {
2: void main() {
3: Row r = SELECT min(a), max(b) FROM t;
4: }
5: }
Parsing
Java + SQL
SQLite cmd
Outline

Can we add them automatically?

We can’t always get it right.

How do we know when to insert a box?

Using parse errors

```
public Row get_students() {
    return SELECT name FROM students;
}
```

1) Trigger on parsing errors

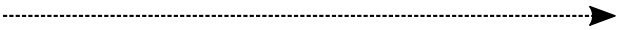
Using parse errors

```
public Row get_students() {
    return SELECT name FROM students;
}
```

2) Find start location(s)

Using parse errors

```
public Row get_students() {
    return SELECT name FROM students;
}
```



3) Find end location(s)

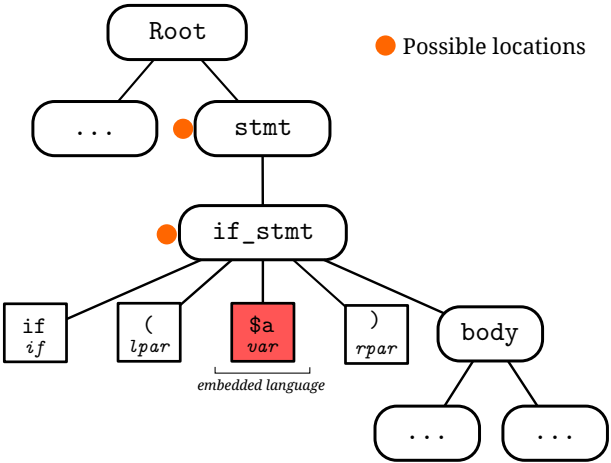
```
public Row get_students() {
    return SELECT name FROM students;
}
```

4) Wrap inside language box

Too many locations degrade performance.

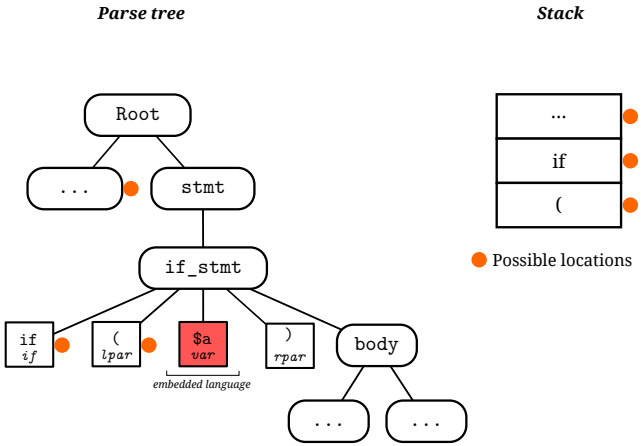
Parse tree heuristic

- Try parents of error node up to root
- Attempt to insert box *before* each node



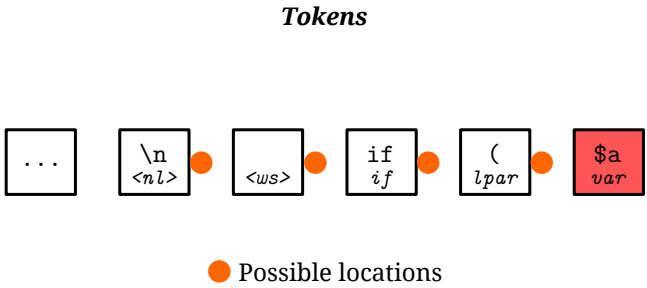
Stack heuristic

- Try each location on the stack
- Attempt to insert box *after* node referenced by location



Line heuristic

- Search each token preceeding the error
- Attempt to insert box *after* each token
- Stop at beginning of line



Combine all heuristics and rank their results.

Rank by how far each candidate allows us to continue parsing.

Demo

Run randomised tests with compositions of Java, Lua, PHP, and SQL.

Methodology

1) Find Java expression

```
public class DCMPG {
  public void accept(Visitor v) {
    v.visit(this);
  }
}
```

2) Delete it

```
public class DCMPG {
  public void accept(Visitor v) {
    v.visit();
  }
}
```

3) Type PHP expression
(one character at a time)

```
public class DCMPG {
  public void accept(Visitor v) {
    v.visit(is_array($bo);
  }
}
```

4) Check if box was inserted

```
public class DCMPG {
  public void accept(Visitor v) {
    v.visit(is_Array($body));
  }
}
```

Total percentage of acceptable outcomes for each benchmark and heuristic.

	JavaLua	JavaPHP	JavaSQL	LuaJava	LuaPHP	LuaSQL	PHPJava	PHPLua	PHPSQL	SQLJava	SQLLua	SQLPHP	Overall
# Tests	888	864	573	275	255	130	562	529	355	233	212	224	5,100
All	98.8%	93.4%	99.3%	98.9%	94.9%	95.4%	99.3%	97.7%	97.5%	97.9%	97.6%	85.3%	96.8%
Parse tree	76.4%	91.1%	100.0%	98.5%	93.3%	93.8%	99.5%	68.6%	98.3%	97.9%	96.2%	83.9%	89.4%
Stack	96.6%	93.8%	99.3%	97.5%	92.9%	67.7%	99.5%	97.9%	97.5%	97.0%	97.6%	85.3%	95.6%
Line	96.4%	79.9%	99.0%	98.9%	94.9%	95.4%	99.3%	95.5%	97.5%	97.9%	97.6%	85.3%	93.8%

The total percentage of outcomes by category.

	Complete insertion (No errors)	Partial insertion (No errors)	Partial insertion (Errors)	No insertion (Valid)	No insertion (Errors)	No insertion (Multi)
All	65.8%	2.8%	2.3%	25.3%	0.8%	2.9%
Parse tree	58.5%	3.1%	1.0%	25.3%	9.6%	2.5%
Stack	64.9%	2.9%	2.0%	25.3%	2.4%	2.5%
Line	63.3%	2.8%	2.3%	25.3%	3.8%	2.5%

The total percentage of outcomes by category.

	Complete insertion ✓ (No errors)	Partial insertion (No errors)	Partial insertion (Errors)	No insertion (Valid)	No insertion (Errors)	No insertion (Multi)
All	65.8%	2.8%	2.3%	25.3%	0.8%	2.9%
Parse tree	58.5%	3.1%	1.0%	25.3%	9.6%	2.5%
Stack	64.9%	2.9%	2.0%	25.3%	2.4%	2.5%
Line	63.3%	2.8%	2.3%	25.3%	3.8%	2.5%

The total percentage of outcomes by category.

	Complete insertion ✓ (No errors)	Partial insertion ✓ (No errors)	Partial insertion (Errors)	No insertion (Valid)	No insertion (Errors)	No insertion (Multi)
All	65.8%	2.8%	2.3%	25.3%	0.8%	2.9%
Parse tree	58.5%	3.1%	1.0%	25.3%	9.6%	2.5%
Stack	64.9%	2.9%	2.0%	25.3%	2.4%	2.5%
Line	63.3%	2.8%	2.3%	25.3%	3.8%	2.5%

The total percentage of outcomes by category.

	Complete insertion ✓ (No errors)	Partial insertion ✓ (No errors)	Partial insertion ✗ (Errors)	No insertion (Valid)	No insertion (Errors)	No insertion (Multi)
All	65.8%	2.8%	2.3%	25.3%	0.8%	2.9%
Parse tree	58.5%	3.1%	1.0%	25.3%	9.6%	2.5%
Stack	64.9%	2.9%	2.0%	25.3%	2.4%	2.5%
Line	63.3%	2.8%	2.3%	25.3%	3.8%	2.5%

The total percentage of outcomes by category.

	Complete insertion ✓ (No errors)	Partial insertion ✓ (No errors)	Partial insertion ✗ (Errors)	No insertion (Valid) ✓	No insertion (Errors)	No insertion (Multi)
All	65.8%	2.8%	2.3%	25.3%	0.8%	2.9%
Parse tree	58.5%	3.1%	1.0%	25.3%	9.6%	2.5%
Stack	64.9%	2.9%	2.0%	25.3%	2.4%	2.5%
Line	63.3%	2.8%	2.3%	25.3%	3.8%	2.5%

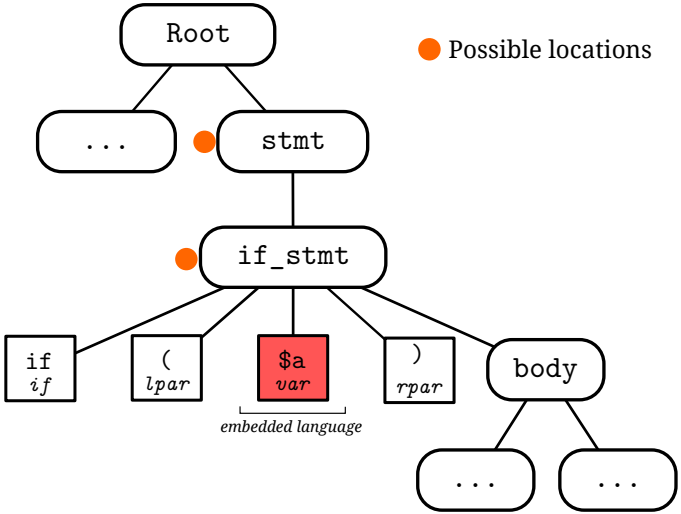
The total percentage of outcomes by category.

	Complete insertion ✓ (No errors)	Partial insertion ✓ (No errors)	Partial insertion ✗ (Errors)	No insertion (Valid) ✓	No insertion (Errors) ✗	No insertion (Multi)
All	65.8%	2.8%	2.3%	25.3%	0.8%	2.9%
Parse tree	58.5%	3.1%	1.0%	25.3%	9.6%	2.5%
Stack	64.9%	2.9%	2.0%	25.3%	2.4%	2.5%
Line	63.3%	2.8%	2.3%	25.3%	3.8%	2.5%

The total percentage of outcomes by category.

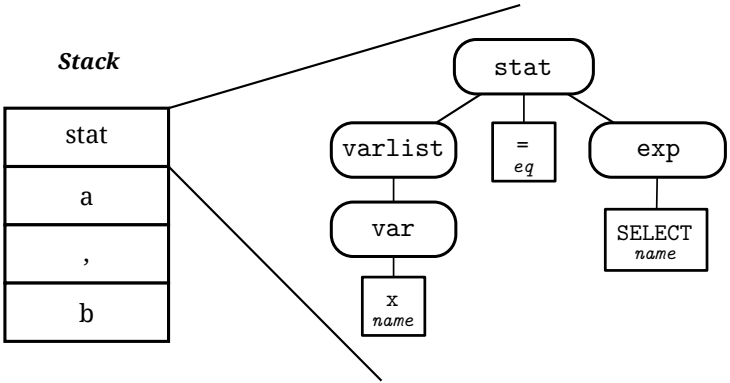
	Complete insertion ✓ (No errors)	Partial insertion ✓ (No errors)	Partial insertion ✗ (Errors)	No insertion (Valid) ✓	No insertion (Errors) ✗	No insertion (Multi) ✓
All	65.8%	2.8%	2.3%	25.3%	0.8%	2.9%
Parse tree	58.5%	3.1%	1.0%	25.3%	9.6%	2.5%
Stack	64.9%	2.9%	2.0%	25.3%	2.4%	2.5%
Line	63.3%	2.8%	2.3%	25.3%	3.8%	2.5%

Parse tree weakness



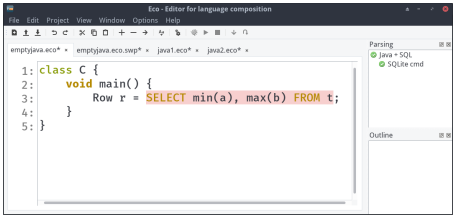
Input

```
x = SELECT a, b FROM t
```



```
class JavaClass {
    function php_func() {
        $a = 1;
    }
}
```

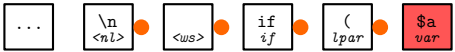
Thank you!




```
public Row get_students() {
    return SELECT name FROM students;
}
```

2) Find start location(s)

Tokens



Possible locations

Inbetween HTML tags any text is valid, thus embedding another language won’t lead to errors.

Lua’s division operator is a comment in Java.

```
int x = 3 // 4 + 1;
```

Evaluation

The all candidates heuristic.

	Complete insertion (No errors)	Partial insertion (No errors)	Partial insertion (Errors)	No insertion (Valid)	No insertion (Errors)	No insertion (Multi)
JavaLua	56.4%	2.4%	1.1%	34.6%	0.1%	5.4%
JavaPHP	86.3%	4.7%	4.4%	1.5%	2.2%	0.8%
JavaSQL	94.4%	0.2%	0.7%	4.7%	0.0%	0.0%
LuaJava	51.3%	4.4%	0.4%	41.8%	0.7%	1.5%
LuaPHP	84.3%	6.7%	1.2%	0.8%	3.9%	3.1%
LuaSQL	86.2%	0.8%	1.5%	8.5%	3.1%	0.0%
PHPJava	42.0%	4.8%	0.5%	49.6%	0.2%	2.8%
PHPLua	47.8%	3.2%	1.7%	42.0%	0.4%	4.7%
PHPSQL	93.5%	0.0%	2.3%	3.9%	0.3%	0.0%
SQLJava	32.2%	0.9%	1.3%	60.9%	0.9%	3.9%
SQLLua	32.1%	1.9%	2.4%	50.0%	0.0%	13.7%
SQLPHP	60.3%	0.4%	14.7%	23.2%	0.0%	1.3%

Evaluation

The parse tree candidates heuristic.

	Complete insertion (No errors)	Partial insertion (No errors)	Partial insertion (Errors)	No insertion (Valid)	No insertion (Errors)	No insertion (Multi)
JavaLua	36.8%	0.0%	0.2%	34.6%	23.3%	5.0%
JavaPHP	83.2%	6.4%	0.5%	1.5%	8.4%	0.0%
JavaSQL	95.3%	0.0%	0.0%	4.7%	0.0%	0.0%
LuaJava	52.7%	3.6%	0.4%	41.8%	1.1%	0.4%
LuaPHP	82.7%	6.7%	0.8%	0.8%	5.9%	3.1%
LuaSQL	84.6%	0.8%	1.5%	8.5%	4.6%	0.0%
PHPJava	37.4%	10.7%	0.2%	49.8%	0.4%	1.6%
PHPLua	19.7%	2.3%	0.2%	42.2%	31.2%	4.5%
PHPSQL	94.4%	0.0%	1.4%	3.9%	0.3%	0.0%
SQLJava	32.2%	0.4%	0.4%	60.9%	1.7%	4.3%
SQLLua	32.5%	0.5%	1.9%	50.0%	1.9%	13.2%
SQLPHP	58.5%	0.0%	12.5%	23.2%	3.6%	2.2%

Evaluation

The stack candidates heuristic.

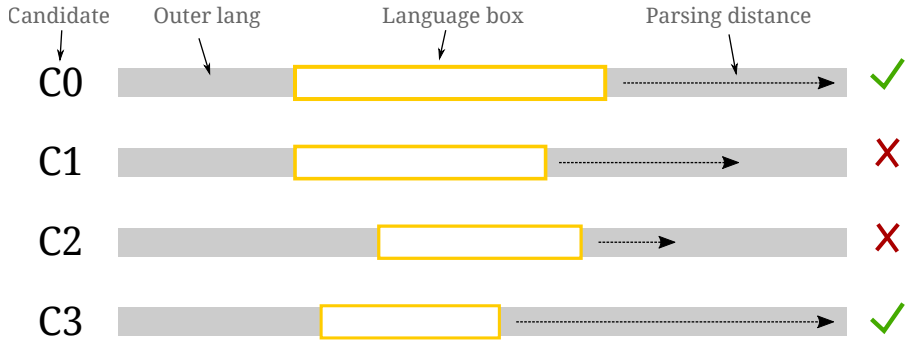
	Complete insertion (No errors)	Partial insertion (No errors)	Partial insertion (Errors)	No insertion (Valid)	No insertion (Errors)	No insertion (Multi)
JavaLua	54.5%	2.4%	0.5%	34.6%	2.8%	5.2%
JavaPHP	87.3%	4.9%	4.1%	1.5%	2.2%	0.1%
JavaSQL	94.4%	0.2%	0.7%	4.7%	0.0%	0.0%
LuaJava	49.8%	5.1%	0.4%	41.8%	2.2%	0.7%
LuaPHP	82.4%	6.7%	1.2%	0.8%	5.9%	3.1%
LuaSQL	58.5%	0.8%	0.8%	8.5%	31.5%	0.0%
PHPJava	43.4%	5.2%	0.4%	49.8%	0.2%	1.1%
PHPLua	48.0%	3.4%	0.2%	42.2%	1.7%	4.3%
PHPSQL	93.5%	0.0%	2.3%	3.9%	0.3%	0.0%
SQLJava	31.3%	0.9%	1.7%	60.9%	1.3%	3.9%
SQLLua	32.5%	1.9%	2.4%	50.0%	0.0%	13.2%
SQLPHP	59.8%	0.4%	14.7%	23.2%	0.0%	1.8%

Evaluation

The line candidates heuristic.

	Complete insertion (No errors)	Partial insertion (No errors)	Partial insertion (Errors)	No insertion (Valid)	No insertion (Errors)	No insertion (Multi)
JavaLua	54.6%	2.4%	1.1%	34.6%	2.5%	4.8%
JavaPHP	74.0%	4.2%	4.3%	1.5%	15.9%	0.2%
JavaSQL	94.1%	0.2%	0.7%	4.7%	0.3%	0.0%
LuaJava	51.3%	4.4%	0.4%	41.8%	0.7%	1.5%
LuaPHP	84.3%	6.7%	1.2%	0.8%	3.9%	3.1%
LuaSQL	86.2%	0.8%	1.5%	8.5%	3.1%	0.0%
PHPJava	43.4%	5.2%	0.5%	49.6%	0.2%	1.1%
PHPLua	45.6%	3.2%	1.7%	42.0%	2.6%	4.7%
PHPSQL	93.5%	0.0%	2.3%	3.9%	0.3%	0.0%
SQLJava	32.2%	0.9%	1.3%	60.9%	0.9%	3.9%
SQLLua	32.1%	1.9%	2.4%	50.0%	0.0%	13.7%
SQLPHP	60.3%	0.4%	14.7%	23.2%	0.0%	1.3%

Ranking


Candidate
Outer lang
Language box
Parsing distance
C0
C1
C2
C3

Code migration gone wrong

Bank legacy systems will remain until CIO life expectancy increases
04 April 2013

Banks still handicapped by IT legacy
11 May 2012

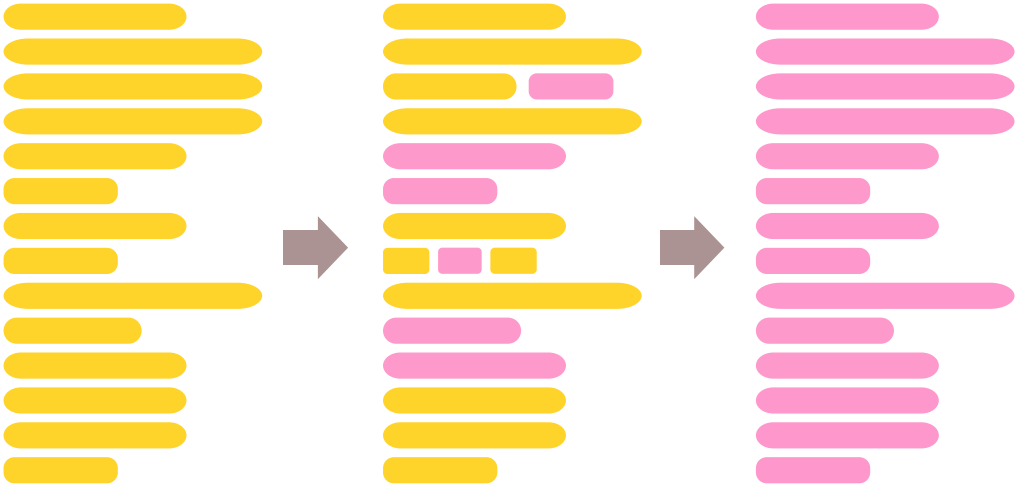
Botched IT migration cost Vodafone customers thousands
26 October 2016

UK bank IT systems "a long way from being robust"
09 January 2014

The biggest surprise about TSB's IT disaster is that people are still surprised when banks' IT fails
25 April 2018

EDF blames system upgrade for customer complaint failures
06 November 2012

Gradual migration



Syntax directed editing