

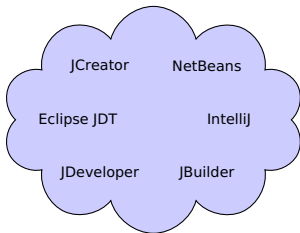
Practical Scope Recovery using Bridge Parsing

Emma Nilsson-Nyman Torbjörn Ekman Görel Hedin

September 29th
SLE'08, Toulouse

Background

IDE:s



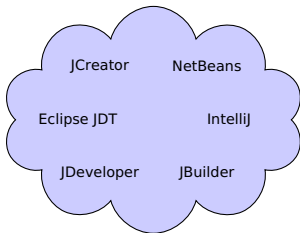
↑
Interactive Editors
Name Completion
Refactorings
Outline, Folding, Hover

Popular Tool

- Very Useful for Developers
- Hard Work to Develop
- Too expensive for DSLs

Background

IDE:s



↑
Interactive Editors
Name Completion
Refactorings
Outline, Folding, Hover

Popular Tool

- Very Useful for Developers
- Hard Work to Develop
- Too expensive for DSLs

Can we make it easier?

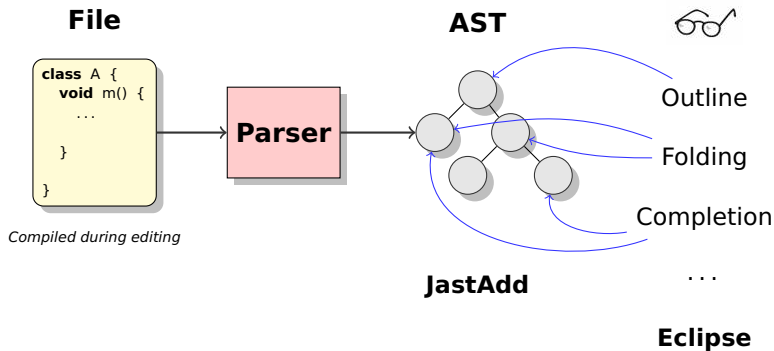
Goal: Generate IDE:s



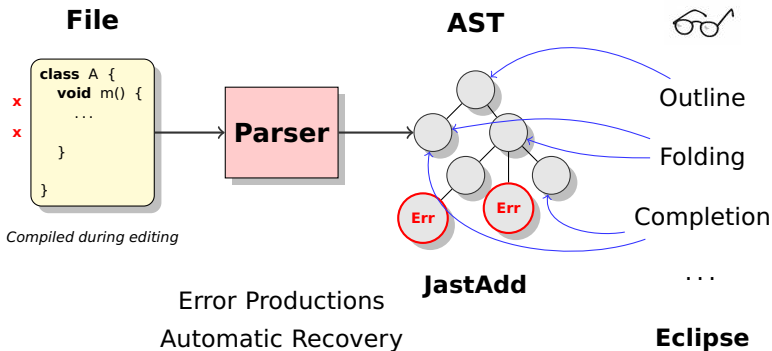
JastAdd: *System used for compiler construction*

Eclipse: *Plugin-architecture used for IDEs*

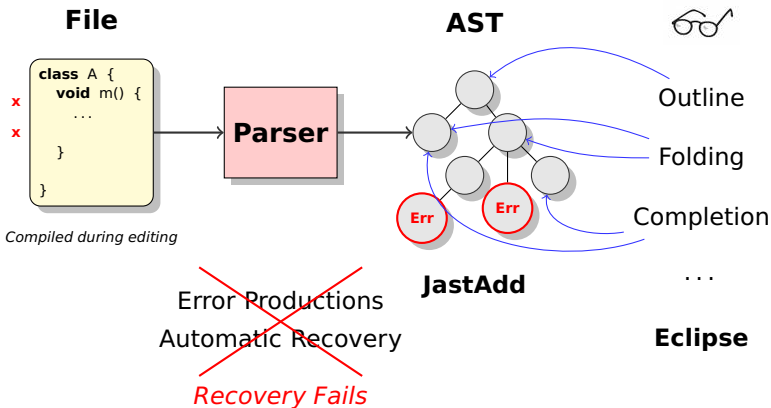
Our Approach



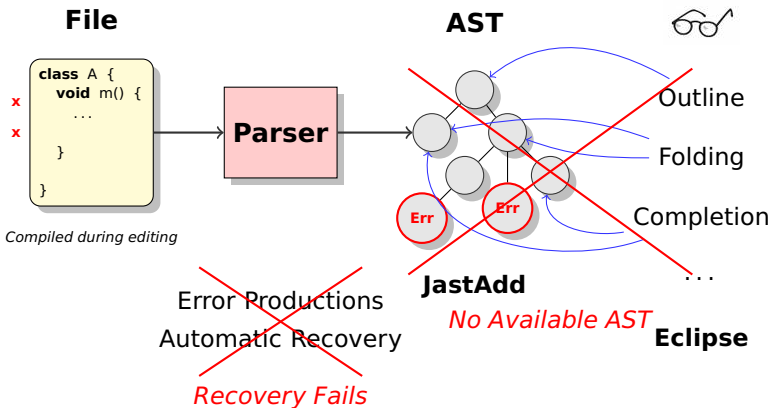
Our Approach



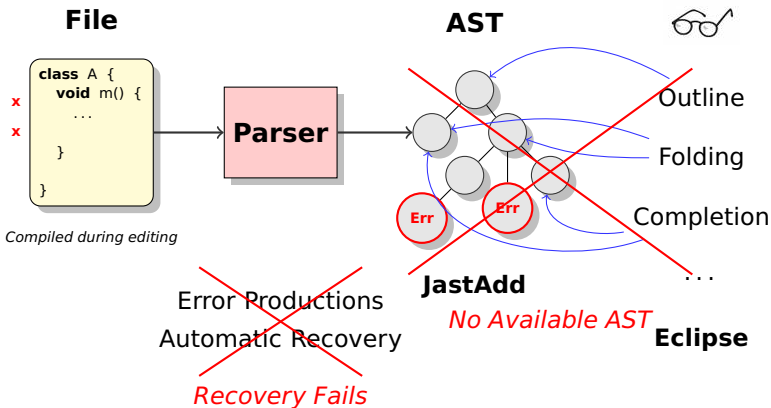
Our Approach



Our Approach



Our Approach



Can we do something to help recovery?

Problem & Approach

```
class C {  
    void m() {  
        int y;  
        int x
```

```
class C {  
    void m() {  
        int y;  
        int x
```

Problem

*Missing synchronization tokens
lead to failed recovery*

Problem & Approach

```
class C {  
    void m() {  
        int y;  
        int x
```

```
class C {  
    void m() {  
        int y;  
        int x
```

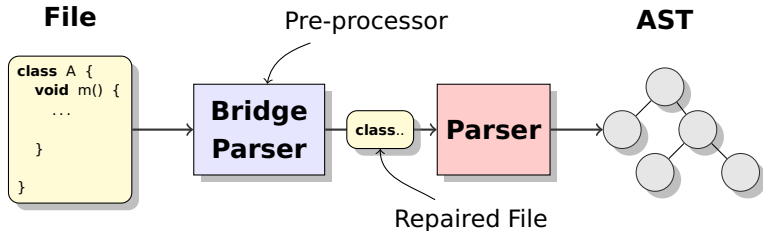
Problem

*Missing synchronization tokens
lead to failed recovery*

Approach

*Pre-process using layout
information*

Problem & Approach



Bridge Parsing

```
class C {  
    void m() {  
        // ...  
    }  
    void n() {  
        // ...  
    }  
}
```

I Island Parsing

- Uninteresting: Water
- Interesting: Islands
- Extension: Reefs

II Bridge Building

- Bridge – *Scope*

Bridge Parsing

```
class C {  
    void m() {  
        // ...  
    }  
    void n() {  
        // ...  
    }  
}
```

I Island Parsing

- Uninteresting: Water
- Interesting: Islands
- Extension: Reefs

II Bridge Building

- Bridge – *Scope*

Bridge Parsing

```
class C {
```

```
    void m() {
```

```
        // ...
```

```
    }
```

```
    void n() {
```

```
        // ...
```

```
    }
```

```
}
```

I Island Parsing

- Uninteresting: Water
- Interesting: Islands
- Extension: Reefs

II Bridge Building

- Bridge – *Scope*

Bridge Parsing

```
class C {
```

```
    void m() {
```

```
        // ...
```

```
    }
```

```
    void n() {
```

```
        // ...
```

```
    }
```

```
}
```

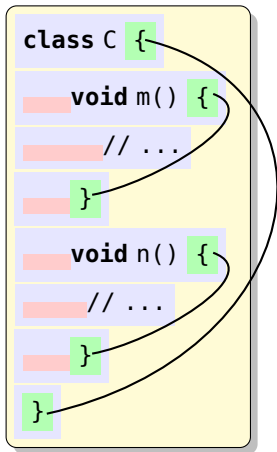
I Island Parsing

- Uninteresting: Water
- Interesting: Islands
- Extension: Reefs

II Bridge Building

- Bridge – *Scope*

Bridge Parsing



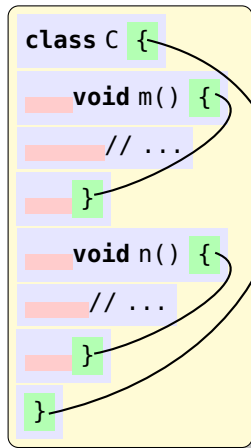
I Island Parsing

- Uninteresting: Water
- Interesting: Islands
- Extension: Reefs

II Bridge Building

- Bridge – *Scope*

Bridge Parsing



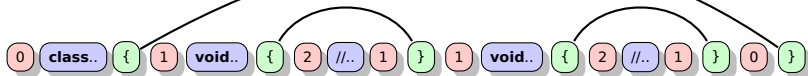
I Island Parsing

- Uninteresting: Water
- Interesting: Islands
- Extension: Reefs

II Bridge Building

- Bridge – Scope

different view

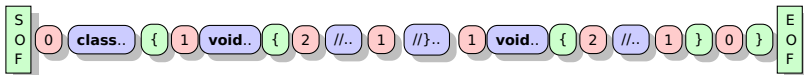


Bridge Parsing

```
class C {  
    void m() {  
        // ...  
    }  
    void n() {  
        // ...  
    }  
}
```

Algorithm

- I Tokenizer
- II Bridge Builder
- III Bridge Repairer

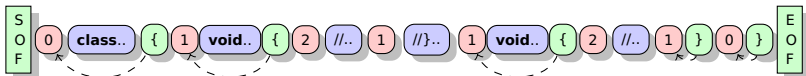


Bridge Parsing

```
class C {  
    void m() {  
        // ...  
    }  
    void n() {  
        // ...  
    }  
}
```

Algorithm

- I Tokenizer
- II Bridge Builder
- III Bridge Repairer

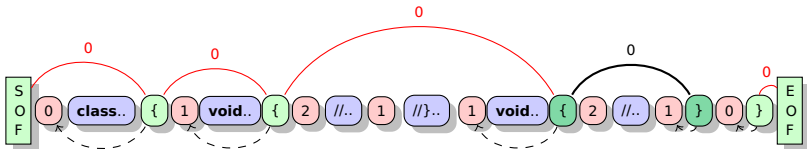


Bridge Parsing

```
class C {
    void m() {
        // ...
    }
    void n() {
        // ...
    }
}
```

Algorithm

- I Tokenizer
- II Bridge Builder
- III Bridge Repairer

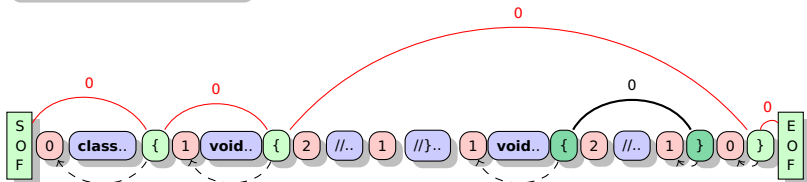


Bridge Parsing

```
class C {
    void m() {
        // ...
    }
    void n() {
        // ...
    }
}
```

Algorithm

- I Tokenizer
- II Bridge Builder
- III Bridge Repairer

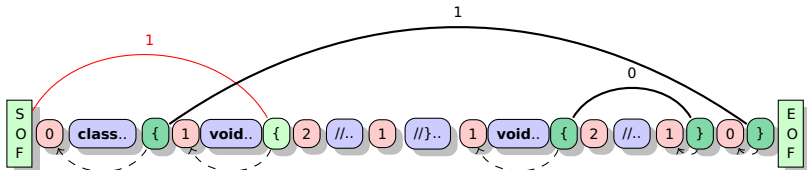


Bridge Parsing

```
class C {  
    void m() {  
        // ...  
    }  
    void n() {  
        // ...  
    }  
}
```

Algorithm

- I Tokenizer
- II Bridge Builder
- III Bridge Repairer

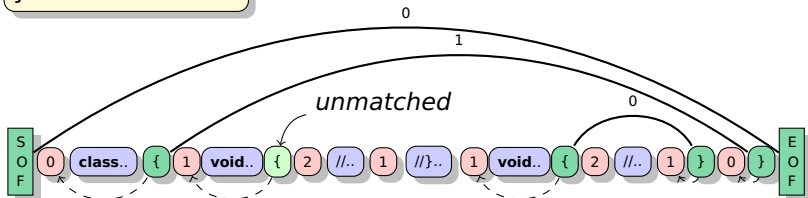


Bridge Parsing

```
class C {  
    void m() {  
        // ...  
    }  
    void n() {  
        // ...  
    }  
}
```

Algorithm

- I Tokenizer
- II Bridge Builder
- III Bridge Repairer

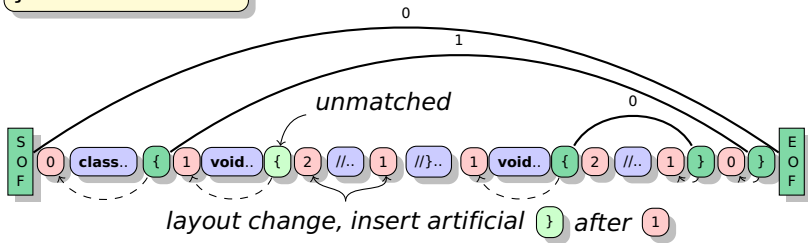


Bridge Parsing

```
class C {  
    void m() {  
        // ...  
    }  
    void n() {  
        // ...  
    }  
}
```

Algorithm

- I Tokenizer
- II Bridge Builder
- III Bridge Repairer



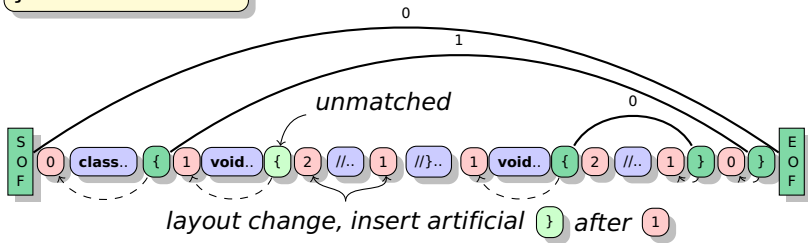
Bridge Parsing

```
class C {  
    void m() {  
        // ...  
    }  
    void n() {  
        // ...  
    }  
}
```

Inserted }

Algorithm

- I Tokenizer
- II Bridge Builder
- III Bridge Repairer



Evaluation

Correct

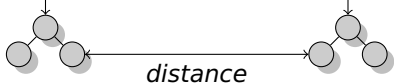
```
class {  
  ..  
}
```

Broken

```
class {  
  ..  
  //}
```

Bridge Parser

Parser



Test Suite

- 10 Correct
- 41 Broken

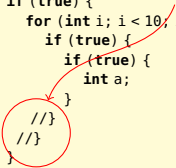
Parser Suite

- Antlr, AntlrBT
- Beaver, BeaverEP
- LPG

Example of Test Cases

```
class C {  
    void m() {  
        if (true) {  
            for (int i; i < 10; i++) {  
                if (true) {  
                    if (true) {  
                        int a;  
                    }  
                }  
            }  
        }  
    }  
}
```


Incomplete



Example of Test Cases


```
class C {  
    void m() {  
        if (true) {  
            for (int i; i < 10; i++) {  
                if (true) {  
                    if (true) {  
                        int a;  
                    }  
                }  
            }  
        }  
    }  
}
```

Incomplete



```
class C {  
    void m() {  
        if (true) {  
            for (int i; i < 10; i++) {  
                if (true) {  
                    if (true) {  
                        int a; }  
                    }  
                }  
            }  
        }  
    }  
}
```

Missing Start



Example of Test Cases

```
class C {  
    void m() {  
        if (true) {  
            for (int i; i < 10; i++) {  
                if (true) {  
                    if (true) {  
                        int a;  
                    }  
                }  
            }  
        }  
    }  
}
```

Incomplete

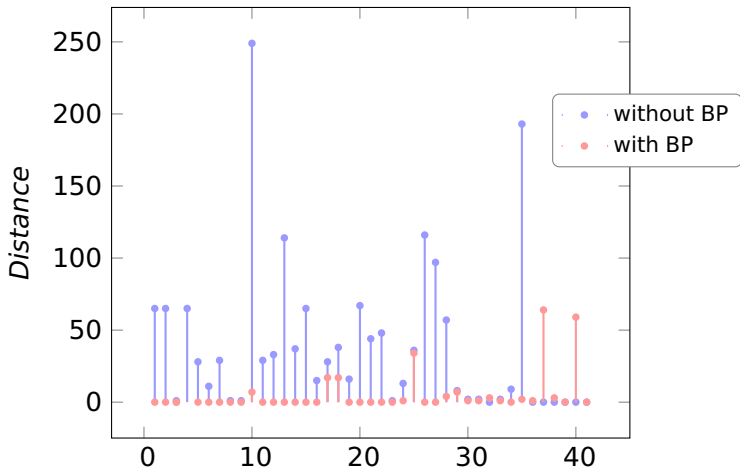
```
class C {  
    void m() {  
        int a;  
    }  
    void n() {  
        int b;  
    }  
}
```

Indentation

```
class C {  
    void m() {  
        if (true) {  
            for (int i; i < 10; i++) {  
                if (true) {  
                    if (true) {  
                        int a;  
                    }  
                }  
            }  
        }  
    }  
}
```

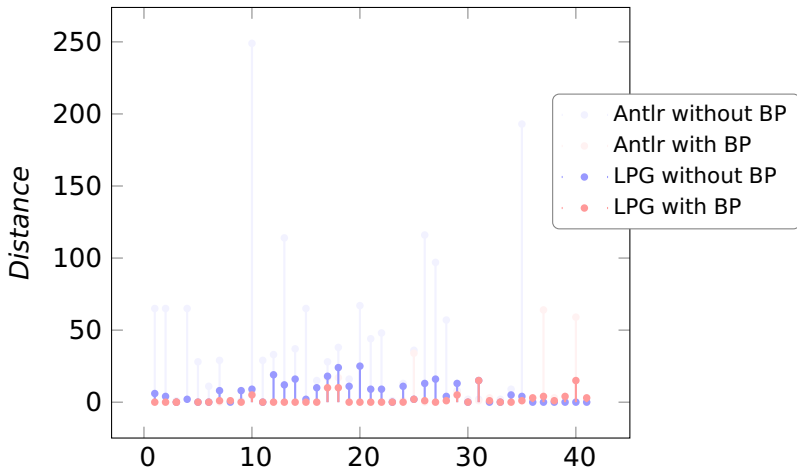
Missing Start

Results using Best Antlr Variant



Incomplete (1-33), Missing Start (34-36), Indentation (37-41)

Results using LPG



Incomplete (1-33), Missing Start (34-36), Indentation (37-41)

Handling Layout Information

Correct Program → No Bridge Parsing

Controlling Layout

- *Copy&Paste*
- *Newline*
- *End Scopes*

```
void m(int a) {  
    int x = 5;  
    int y = 0;  
    if (a > 0) {  
        //...  
    }  
    x++;  
    //...  
}
```

Copy&Paste

Handling Layout Information

Correct Program → ***No Bridge Parsing***

Controlling Layout

- *Copy&Paste*
- *Newline*
- *End Scopes*

```
void m(int a) {  
    int x = 5;  
    int y = 0;  
    if (a > 0) {  
        //...  
    }  
    x++;  
    //...  
}
```

Copy&Paste

Handling Layout Information

Correct Program → ***No Bridge Parsing***

Controlling Layout

- *Copy&Paste*
- *Newline*
- *End Scopes*

```
void m(int a) {  
    int x = 5;  
    int y = 0;  
    if (a > 0) {  
        //...  
    }  
    x++;  
    //...  
}
```

Newline

Handling Layout Information

Correct Program → ***No Bridge Parsing***

Controlling Layout

```
void m(int a) {  
    int x = 5;  
    int y = 0;  
    if (a > 0) {  
        -  
        //...  
    }  
    x++;  
    //...  
}
```

Newline


- *Copy&Paste*
- *Newline*
- *End Scopes*

Handling Layout Information

Correct Program → No Bridge Parsing

Controlling Layout

```
void m(int a) {  
    int x = 5;  
    int y = 0;  
    if (a > 0) {  
        if (a < x) {  
            //...  
        }  
        x++;  
        //...  
    }  
}
```



End Scope

- *Copy&Paste*
- *Newline*
- *End Scopes*

Handling Layout Information

Correct Program → No Bridge Parsing

Controlling Layout

```
void m(int a) {  
    int x = 5;  
    int y = 0;  
    if (a > 0) {  
        if (a < x) {  
            -  
        }  
        //..  
    }  
    x++;  
    //...  
}
```

End Scope

- *Copy&Paste*
- *Newline*
- *End Scopes*

Handling Layout Information

Correct Program → ***No Bridge Parsing***

Controlling Layout

- *Copy&Paste*
- *Newline*
- *End Scopes*

Future Work

- *History*
- *Editing Patterns*

```
void m(int a) {  
    int x = 5;  
    int y = 0;  
    if (a > 0) {  
        if (a < x) {  
            -  
        }  
        //..  
    }  
    x++;  
    //...  
}
```

End Scope

Conclusions

Recovery Technique *for IDE:s*

- ***Independent***
of Parser Generator
- ***Improved Recovery***
Excellent in Many Cases
- **Used in Practice**
Java IDE, JastAdd IDE, JModelica IDE
(Ask for demo during the break)