Abstract Syntax Trees: Parsing ABL

GUSTAS MICKUS | g.mickus@balticamadeus.com

PUG CHALLENGE 2026 / AMSTERDAM



baltic amadeus

baltic amadeus

Who are we?

About balticamadeus

We provide IT and strategic consulting services, including Progress OpenEdge, Omnichannel, Self-service, Marketplace, Information security, Enterprise architecture,
Cloud solutions, and more.

96% of clients recommend us

500+ local and international clients

40/60 local vs international clients

2000+ projects, including corporations, SME-level companies, and startups

37+ years of experience

300+ IT talents

baltic amadeus

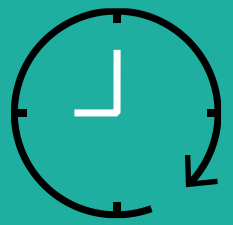
Progress OpenEdge department:

International clients **70%** | Baltic **30%**



Team of OpenEdge
±47 & growing

Av. Age of the team
±29 years



Av. years of experience
±9 years
Retention rate
90%



Helping with:
- Support
- Development
- Modernisation
- Team enhancement

OUR VS CODE EXTENSIONS ^_^



ProBro



ProPeek



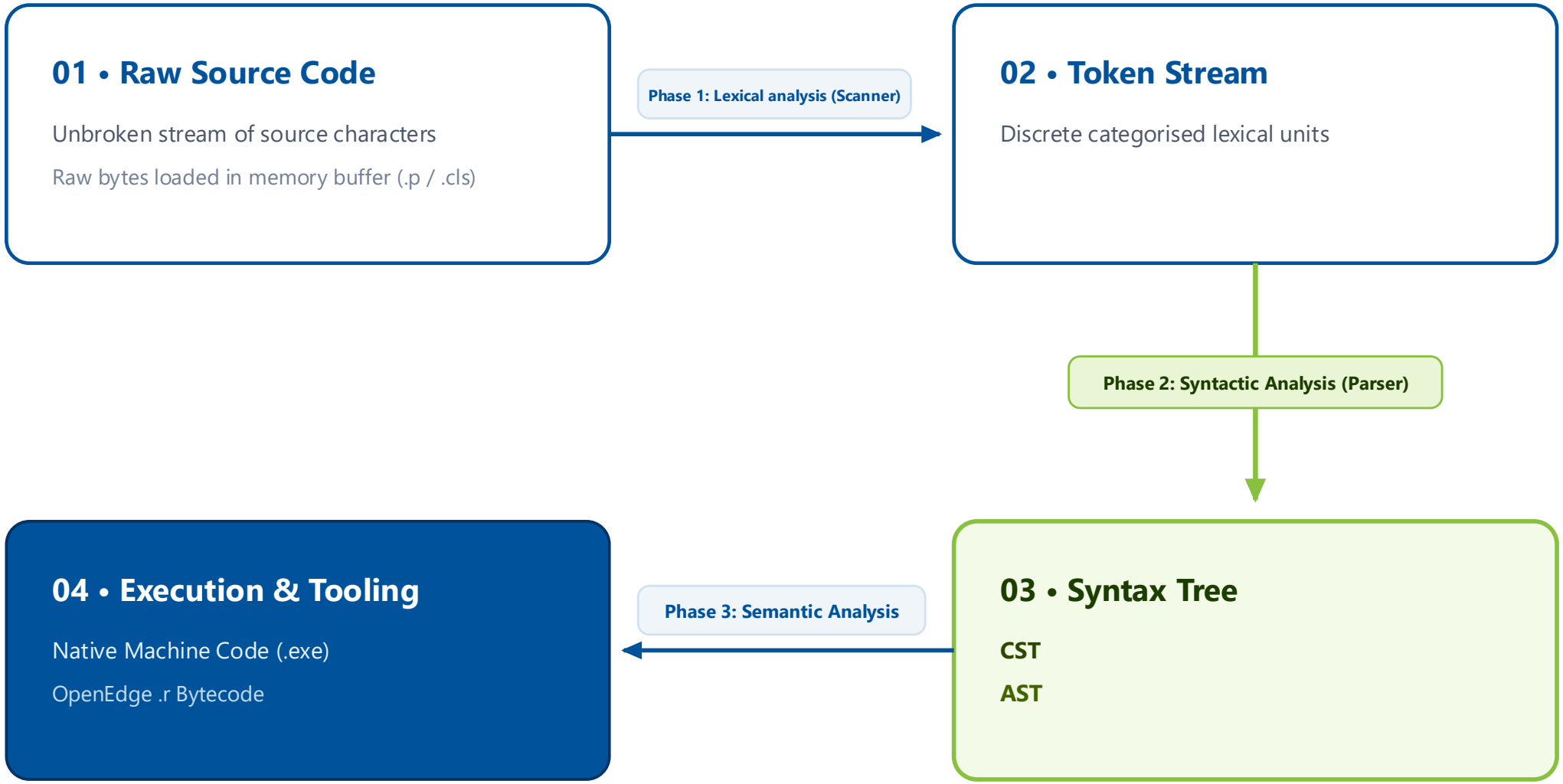
OpenEdge ABL Formatter

baltic
amadeus

How do compilers see code?

baltic amadeus

The compiler pipeline



baltic amadeus

Tokenisation

SOURCE BUFFER (RAW CHARACTERS)

```
01  DEFINE VARIABLE cAwesomeConferenceName AS CHARACTER
02      NO-UNDO INITIAL "PUG Amsterdam".
```

Lexical Scanner (Regex DFA Matching)

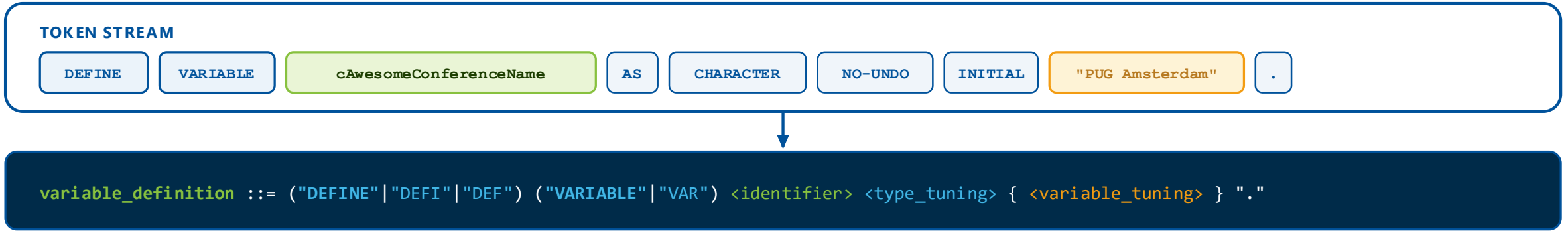
Emitted Token Stream & Metadata Coordinates

TOKEN TYPE	RAW LEXEME	COORDINATES	ATTACHED TRIVIA CHANNEL
KEYWORD_DEFINE	"DEFINE"	Line 1, Col 1	Trailing Space: ' '
KEYWORD_VARIABLE	"VARIABLE"	Line 1, Col 8	Trailing Space: ' '
IDENTIFIER	"cAwesomeConferenceName"	Line 1, Col 17	Trailing Space: ' '
TYPE_CHARACTER	"CHARACTER"	Line 1, Col 43	Newline Trivia: '\n' + ' ' (4 spaces)
KEYWORD_NO_UNDO	"NO-UNDO"	Line 2, Col 5	Trailing Space: ' '
STRING_LITERAL	"\"PUG Amsterdam\""	Line 2, Col 21	None
PERIOD	"."	Line 2, Col 36	End of Statement

baltic amadeus

Syntactic Analysis

- We are going to look at all the tokens one by one and compare them against a formal Context-Free Grammar (EBNF) rule.



- Then, the parser collapses the tokens into a hierarchical structure.

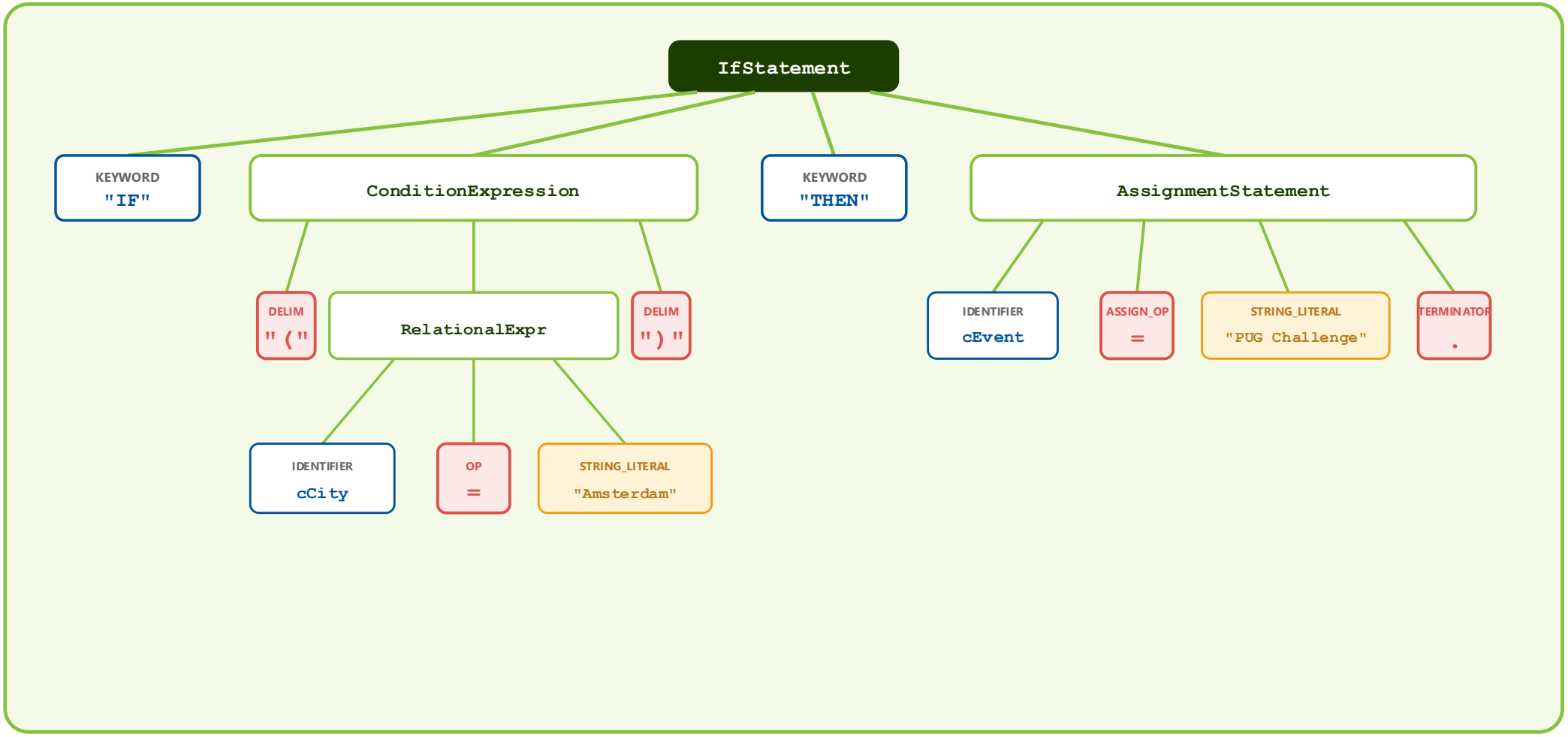
baltic amadeus

Concrete Syntax Tree (CST)

SOURCE ABL:

```
IF (cCity = "Amsterdam") THEN cEvent = "PUG Challenge".
```

COMPLETE CST



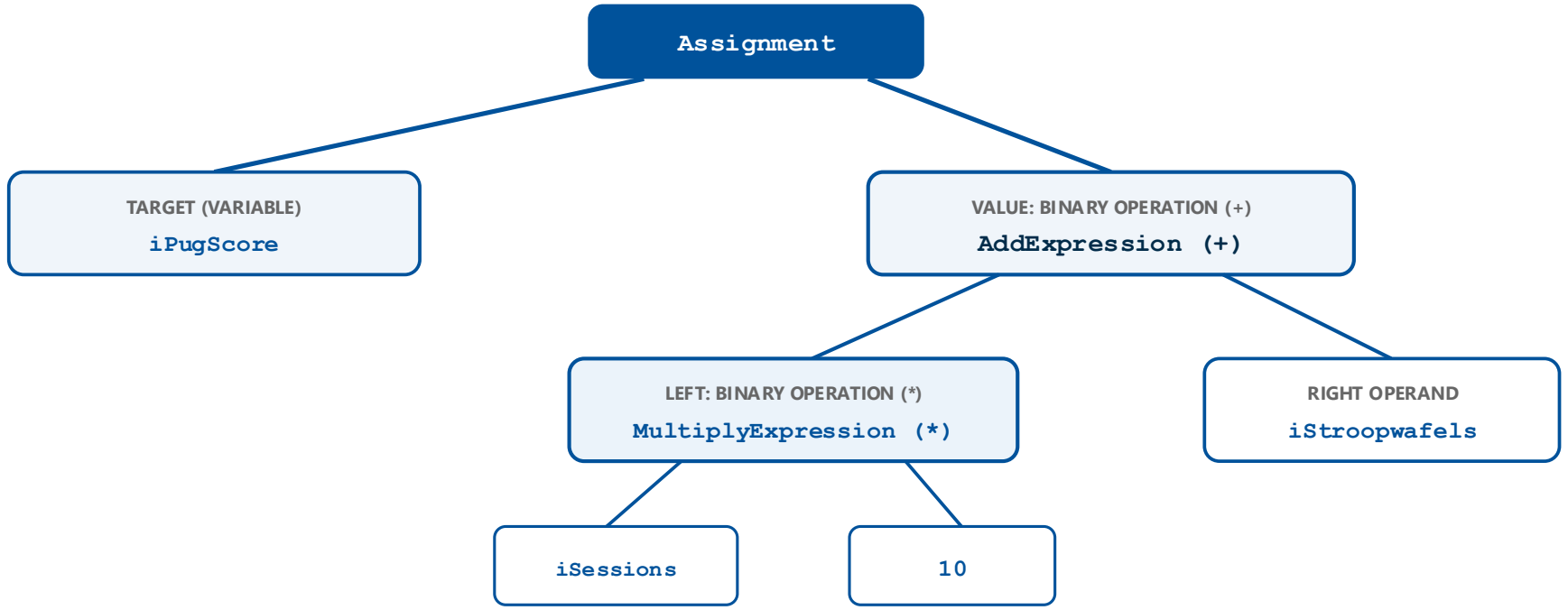
baltic amadeus

Abstract Syntax Tree (AST)

SOURCE ABL:

```
iPugScore = (iSessions * 10) + iStroopwafels.
```

ABSTRACT (AST)



baltic amadeus

Why developer tooling needs CSTs?

- *You cannot format, refactor, or highlight what your parser throws away.*

```
01  /* If a parser throws away delimiters, how do you format this? */
02  ASSIGN    cEventName     =     "PUG Amsterdam"    /* keep this comment! */
03            iAttendees     =     42.
```

baltic amadeus

Why do compilers & linters prefer ASTs?

- *Eliminating syntactic noise accelerates semantic validation and code generation.*

```
01  /* A linter or compiler backend only cares about the red flags: */

02  FIND FIRST Customer WHERE Customer.City = "Amsterdam".

03  /* Warning: Missing NO-LOCK! The AST makes this check trivial. */
```

baltic amadeus

CST vs. AST

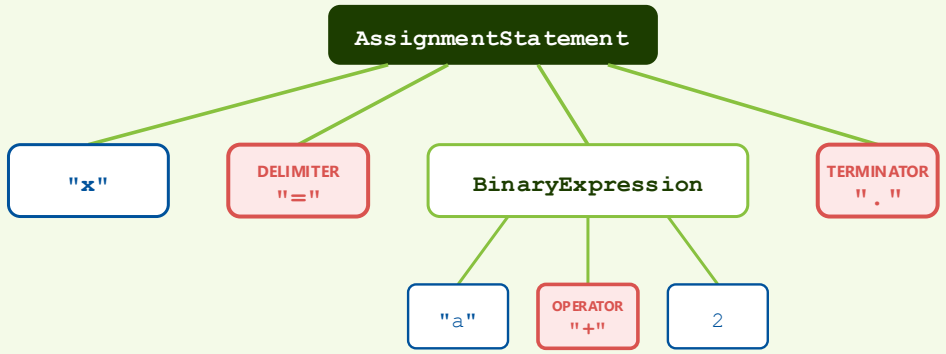
TARGET EXPRESSION: `x = a + 2.`

4 TOKENS + TRIVIA

Concrete Syntax Tree (CST)

THE FORMATTER'S TREE

1:1 surface fidelity ; Retains EVERYTHING



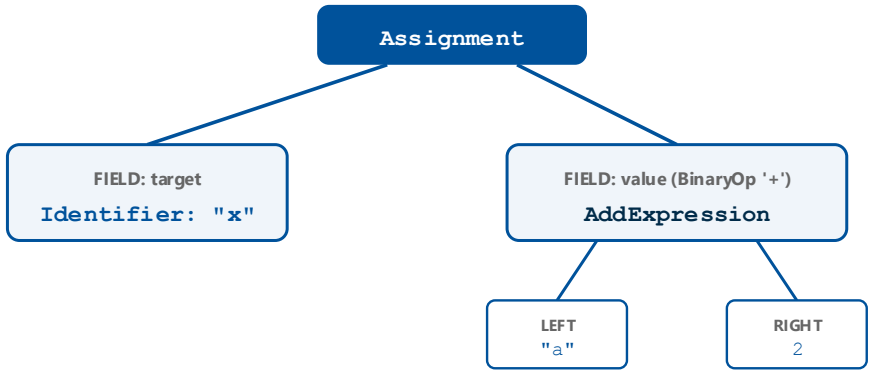
CAN RECONSTRUCT 100% OF ORIGINAL TEXT

Contains exact delimiter positions to calculate column offsets, pad whitespace around '=', and format indentation safely.

Abstract Syntax Tree (AST)

THE COMPILER'S TREE

Condensed semantic intent



CANNOT RECONSTRUCT SOURCE LAYOUT

The '=' operator, full stop '.', and grouping parentheses are completely stripped.

PRIMARY CONSUMERS

CST: Formatters, IDE highlighters, refactoring engines

AST: Compilers (IR/SSA codegen), type checkers, linters

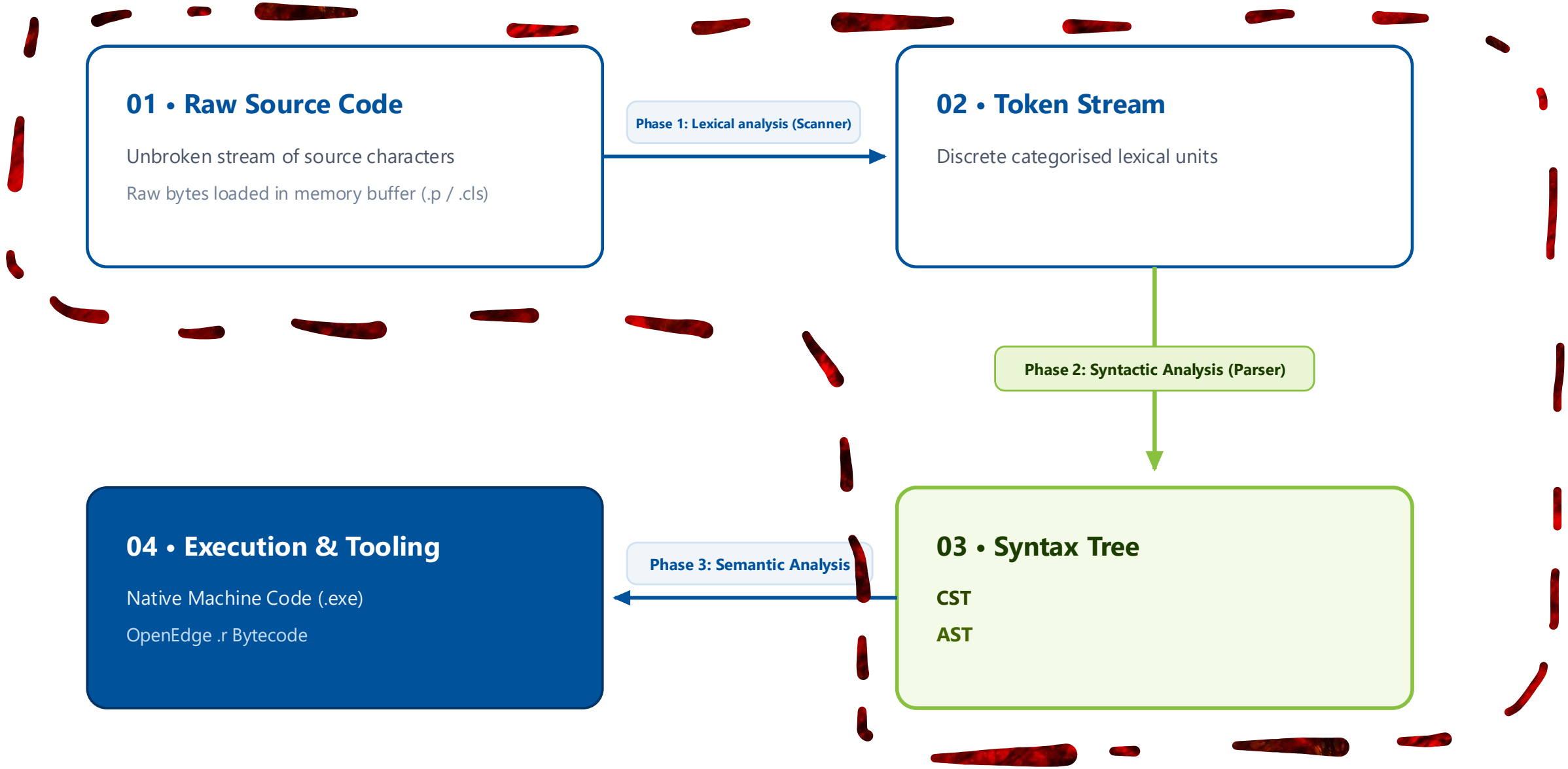
ARCHITECTURAL TAKEAWAY

Compilers discard structural syntax sugar to optimise execution speed.

Formatters **require concrete syntax trees** to preserve author intent.

baltic amadeus

The compiler pipeline (Front-end)

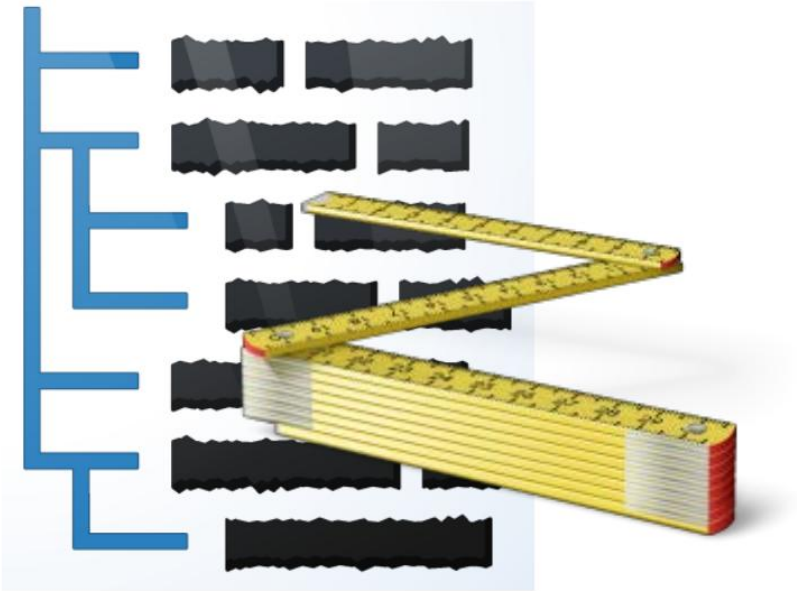


baltic amadeus

The OpenEdge Landscape – Two Parsing Titans

ProParse

JAVA / .NET



VS

Tree-sitter

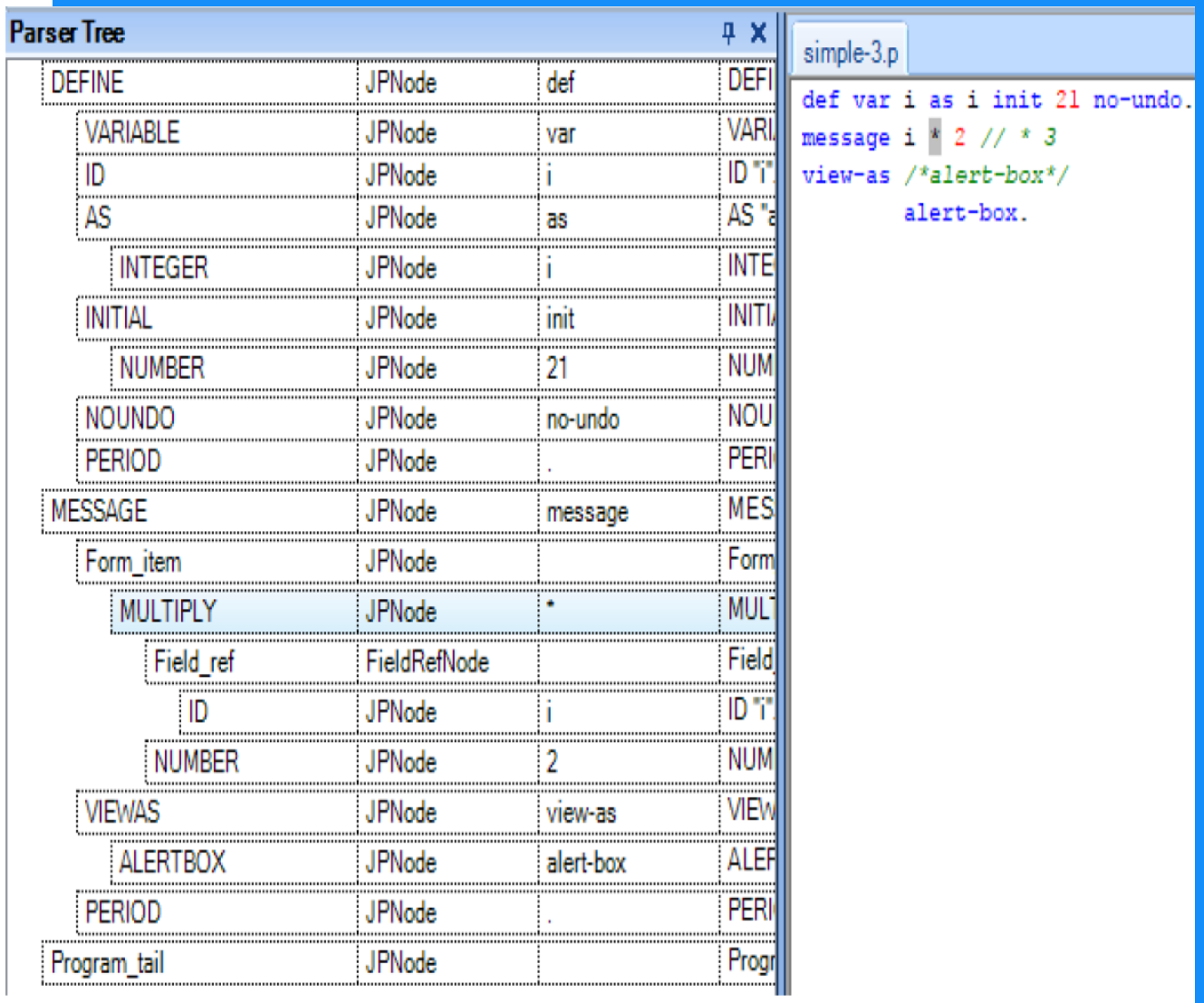
C11 / WASM



baltic amadeus

ProParse

- 👍 deep AST static analysis (Prolint)
- 👍 automated migration pipelines
- 👍 architectural refactoring
- 😥 batch-oriented runtime
- 😪 zero incremental support
- ☹️ brittle by design



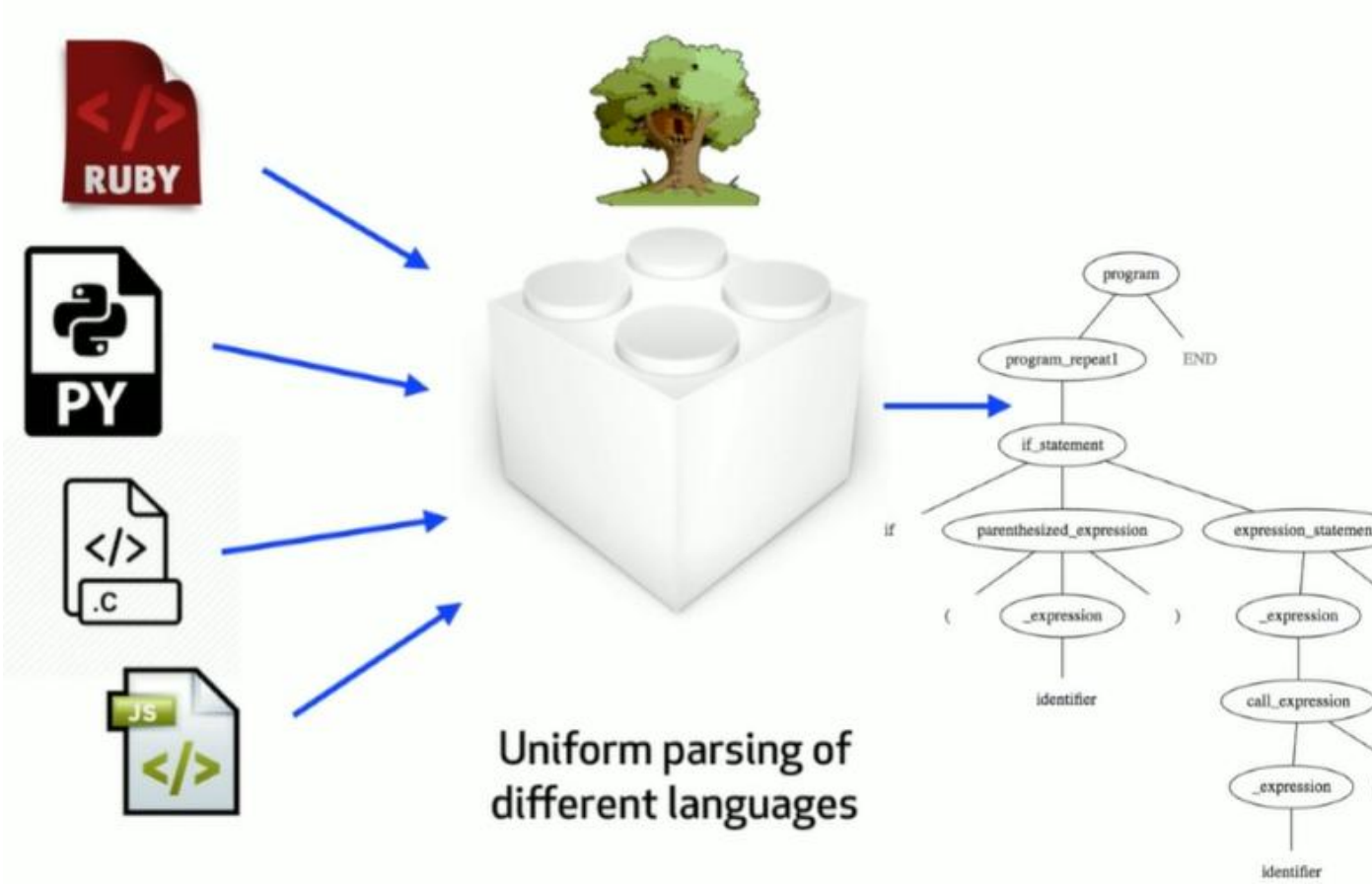
baltic amadeus

tree-sitter

Created by Max Brunsfeld.

Powers syntax highlighting and tooling across GitHub.

Implemented in pure, zero-dependency C11, easily compiled to WebAssembly (WASM).

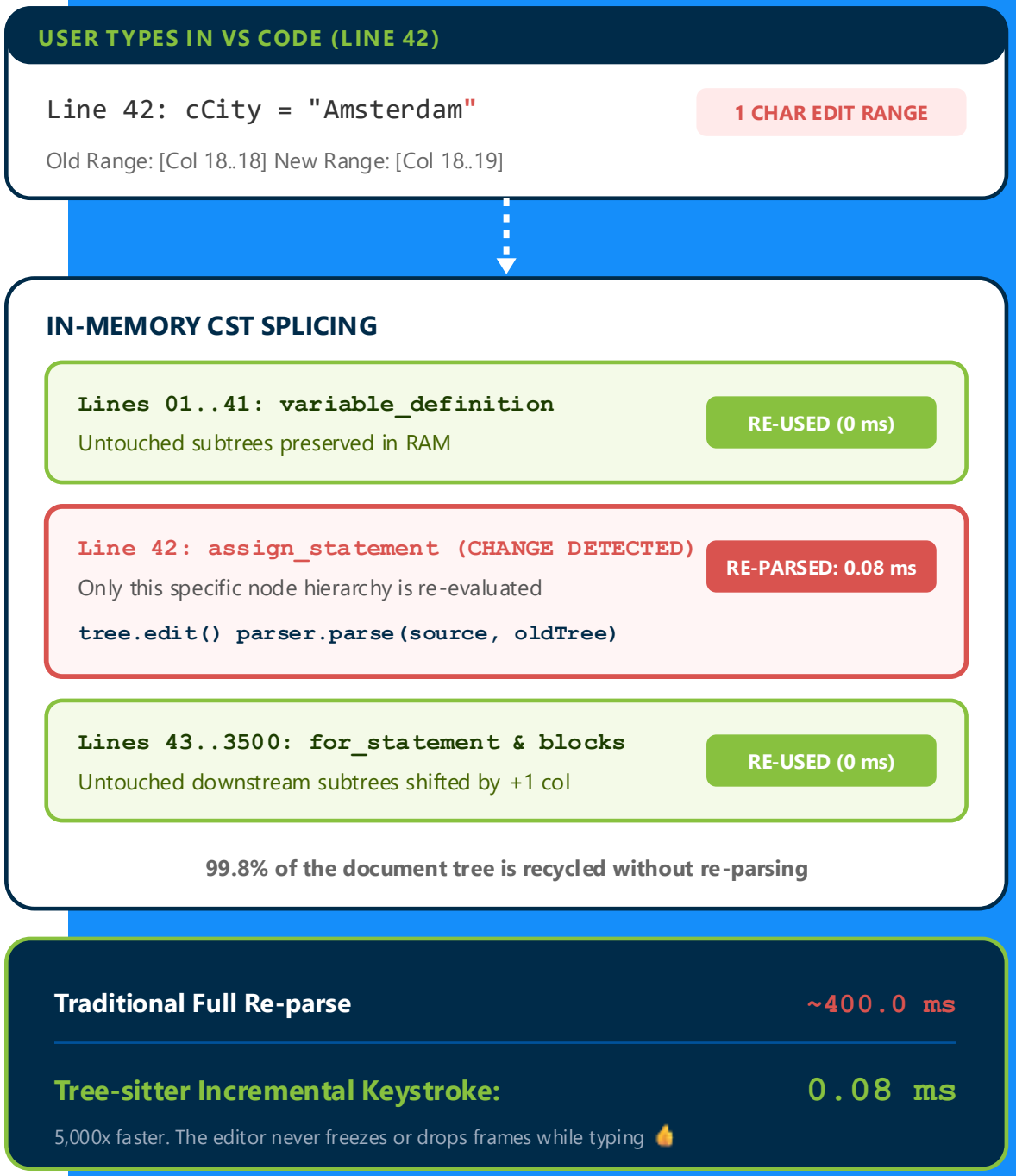


baltic amadeus

Incremental Re-parsing

ProParse re-parses the entire document upon every change.

`tree-sitter` compares edit ranges and re-parses **only the modified syntax nodes.**



baltic amadeus

Fault Tolerance

```
DEFINE VARIABLE cConferenceName
DEFINE VARIABLE iBikesRented
DEFINE VARIABLE lEnjoyingLife
IF lEnjoyingLife = TRUE THEN DO        /* ← THE FAULT: missing ":" after DO */
    MESSAGE "HIGHLIGHT ME!" VIEW-AS ALERT-BOX.
END.
FOR EACH ttCuteAttendee NO-LOCK:
    ASSIGN ttCuteAttendee.cCanalTourStatus = 'Booked'.
END.
```

ID	TYPE	START POS	END POS	INDEX	TEXT
2603400	ERROR	33:29	35:3	313:415	DO /* ← THE FAULT: missing ":" after DO */ MESSAGE "HIGHLIGHT ME!" VIEW-AS ALERT-BOX. END
2603496	if_statement	33:0	38:4	284:512	IF lEnjoyingLife = TRUE THEN DO /* ← THE FAULT: missing ":" after DO */ MESSAGE "HIGHLIGHT ME!" VIEW-AS ALERT-BOX. END. FOR EACH ttCuteAttendee NO-LOCK: ASSIGN ttCuteAttendee.cCanalTour

Syntax errors do not halt parsing.

baltic amadeus

`tree-sitter` for ABL

Formal ABL Context-Free Grammar defined programmatically in JavaScript.

Operates with zero reliance on _progres, prowin, or OpenEdge licenses.

```
// From tree-sitter-abl/grammar.js
module.exports = grammar({
  name: "abl",

  // Operator precedence hierarchy
  const PREC = {
    UNARY: 8,
    EXP: 7,
    MULTI: 6,
    ADD: 5,
    COMPARE: 4,
    LOGICAL: 3,
    ASSIGN: 2
  };

  // External C scanner for contextual tokens
  externals: ($) => [
    $._namedot,
    $._namecolon,
    $._escaped_string,
    $._for_keyword,
    $._return_keyword
    // ... 17 contextual symbols
  ],

  // Conflict resolution sets
  conflicts: ($) => [
    [$.sort_clause],
    [$.if_statement],
    [$.radio_set_phrase, $._expression],
    // ... 24 ambiguity declarations
  ]
});
```

baltic amadeus

The Secret Weapon:

scanner.c

ABL has a bunch of quirks.

```
DEFINE VARIABLE input AS CHARACTER NO-UNDO.
```

```
input = INPUT FRAME f1 speaker-name.
```

The `scanner.c` solution. Lookahead: keyword only before expressions.

ABL Quirk	CFG Problem	C Scanner Solution
myTable.field	Dot operator vs statement terminator	Zero whitespace check after .
"~n~t""embedded""	Tilde escapes + doubled quotes	Stateful byte-level parsing
Unclosed preprocessors	&IF ... &ENDIF spans files	External state tracking

```c
// From tree-sitter-abl/src/scanner.c
bool tree_sitter_abl_external_scanner_scan(
  void *payload,
  TSLexer *lexer,
  const bool *valid_symbols
) {
  // Case 1: Contextual punctuation (. vs :: vs :)
  if (valid_symbols[NAMEDOT] && lexer->lookahead == '.') {
    lexer->advance(lexer, false);
    if (!iswspace(lexer->lookahead)) {
      lexer->result_symbol = NAMEDOT;  // myObject.property
      return true;
    }
  }

  // Case 2: Contextual keywords (RETURN can be identifier)
  if (valid_symbols[RETURN_KEYWORD]) {
    if (match_keyword(lexer, "RETURN", RETURN_KEYWORD)) {
      // Only matches at word boundaries
      return true;
    }
  }

  // Case 3: String escapes with tilde (~) and embedded quotes
  if (valid_symbols[ESCAPED_STRING]) {
    while (!lexer->eof(lexer)) {
      if (lexer->lookahead == '~') {
        lexer->advance(lexer, false);  // Escape next char
        lexer->advance(lexer, false);
      } else if (lexer->lookahead == '"') {
        if (next_char == '"')  // Embedded ""
          continue;
        return true;  // End of string
      }
    }
  }
}
```

baltic amadeus

The Formatter pipeline

STEP 01: DECLARATIVE GRAMMAR — JS DSL

`grammar.js`

~3,000 lines of declarative JavaScript

8 operator precedences & 24 conflict sets

Formally defines 540+ ABL AST node types

Human-readable, version-controlled source

Phase 1: tree-sitter generate

STEP 02: GENERATED C11 AUTOMATON — RAW C11

`parser.c` (16.7 MB / 560k lines)

100,000+ GLR parse state transitions

`scanner.c` (6.0 KB hand-written)

17 contextual tokens (dots, tilde escapes)

Phase 2: Emscripten (emcc / Clang)

STEP 03: SANDBOXED WASM BINARY — WASM

`tree-sitter-abl.wasm`

~200 KB compressed (gzip)

Phase 3: Sandbox Loading

STEP 04: PRODUCTION EXTENSION RUNTIME — CLIENT

VS Code Extension Host

Baltic Amadeus OpenEdge ABL Formatter

Zero native dependencies (single binary)

Cross-platform: Windows / macOS / Linux

Zero OpenEdge runtime licences or DB needed

baltic amadeus

THE BENCHMARKS

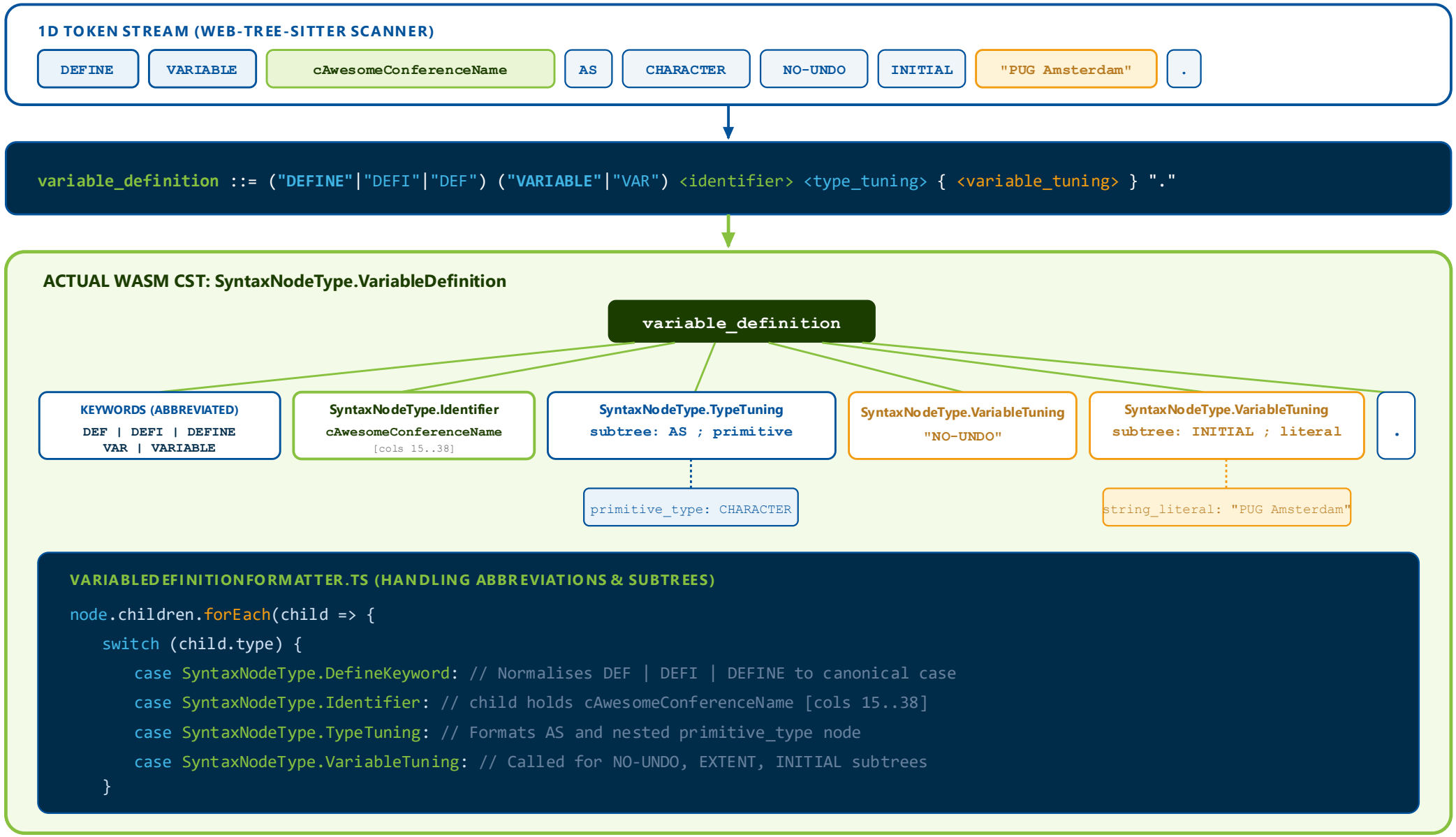
Results from 5 benchmark runs on production OpenEdge ABL files:

File	Lines	Size	Tree-sitter	ProParse	Speedup
small.p	60	2.4 KB	0.51 ms	3.38 ms	**6.6×**
medium.cls	536	20.9 KB	4.01 ms	15.02 ms	**3.7×**
large.p	6,049	292.4 KB	7.02 ms	90.54 ms	**12.9×**

Source: https://github.com/GytRag/abl-tree-sitterparser-benchmarks

baltic amadeus

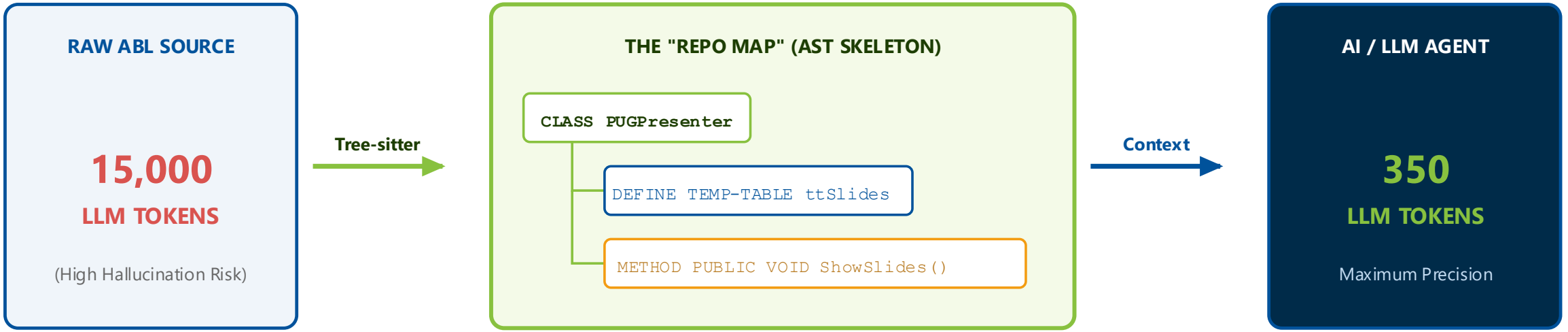
From theory to practice!



baltic amadeus

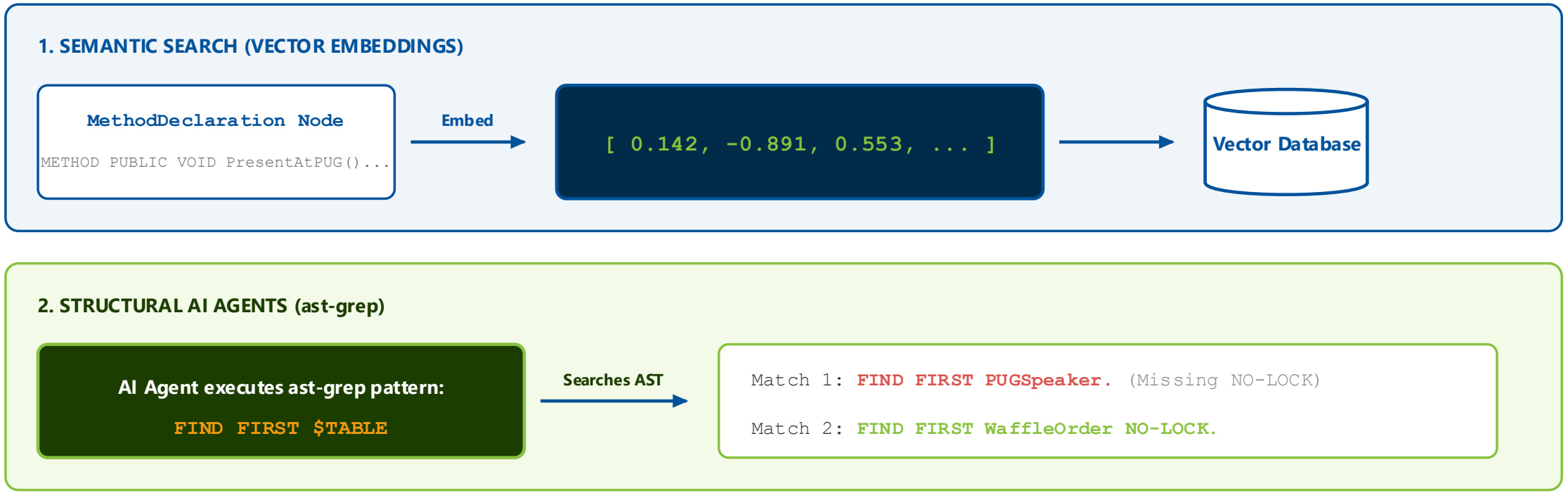
THE FUTURE: AST-Grounded AI Coding Agents

- Cursor or Windsurf use `tree-sitter` to index codebases into trees.
- For instance, CocoIndex use `tree-sitter` to build a map of class definitions, signatures, temp-tables. This saves tokens and prevents hallucinations.



baltic amadeus

- Cursor embeds logical AST chunks (methods/blocks) into vector databases, allowing the AI to search code semantically rather than line-by-line.
- Agents now equipped with `ast-grep` to search code structures.



baltic amadeus

Conclusions

- **Tooling Requires Trees**: Compilers rely on ASTs to optimise execution. IDEs and formatters rely on CSTs to preserve author intent.
- ProParse remains very powerful for deep semantic migrations and batch analysis.
- `tree-sitter` brings keystroke-speed, fault-tolerant, zero-dependency parsing to modern IDEs.

baltic amadeus

Let's talk?

GUSTAS MICKUS |
g.mickus@balticamadeus.com

baltic amadeus

baltic amadeus

Subscribe to our newsletter!

(✿´◡`✿)

