Implementing Semantic Enrichment in StarLasu

Lorenzo Addazi

Software Language Engineer @ Strumenta

LangDev Meetup

15th of November, 2023

Utrecht, The Netherlands

STRU
MEN
TA

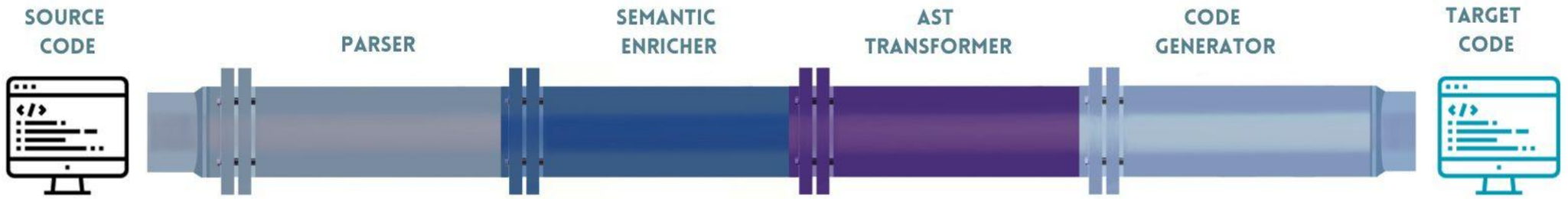
Who am I?

Lorenzo Addazi
Software Language Engineer at Strumenta

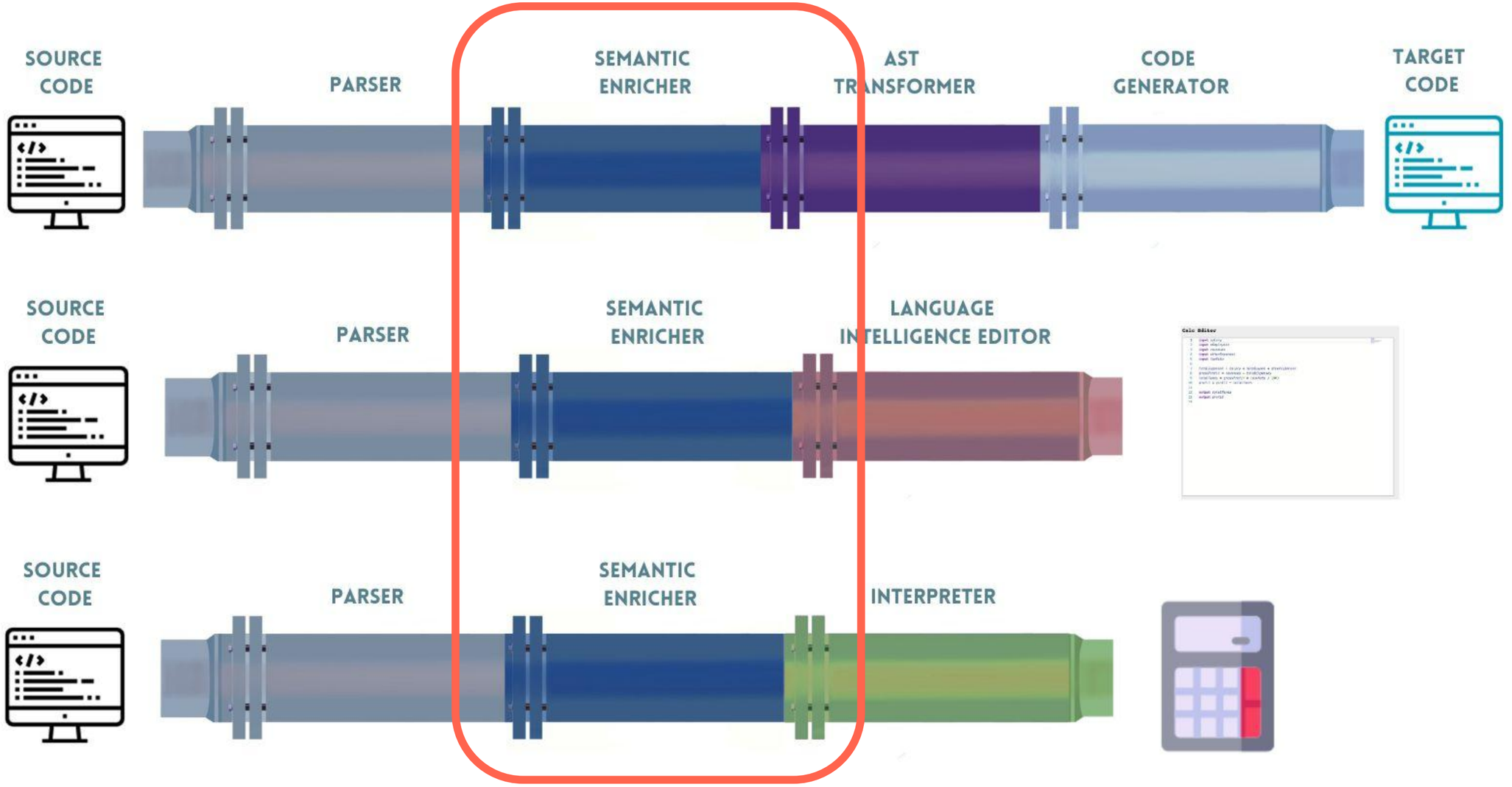
Agenda

- Context
- Semantic Enrichment
  - Definition
  - Implementation
- Demo

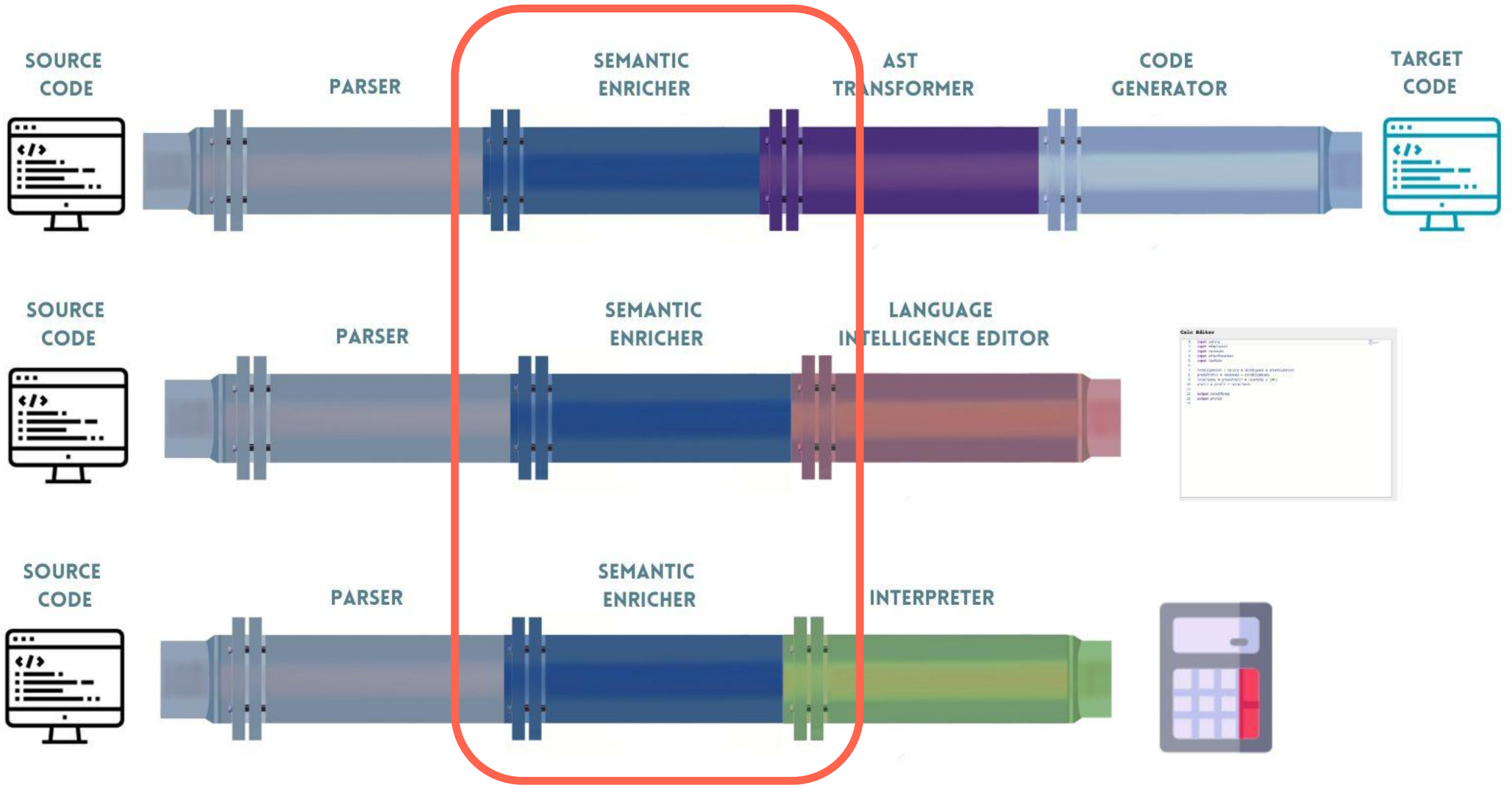
Context - The StarLasu Approach



Context - The StarLasu Approach



Semantic Enrichment - Definition

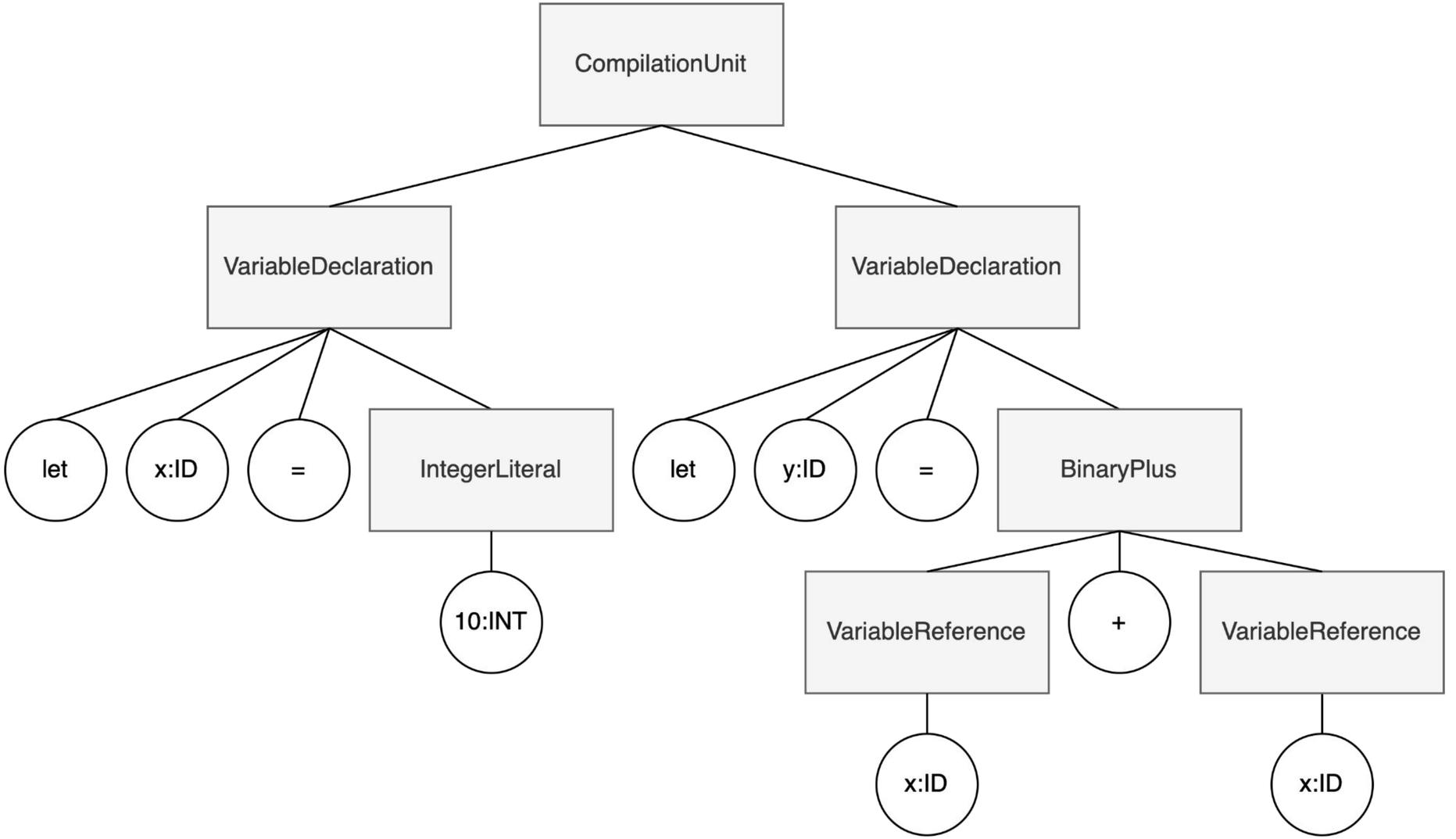


Semantic Enrichment - Definition

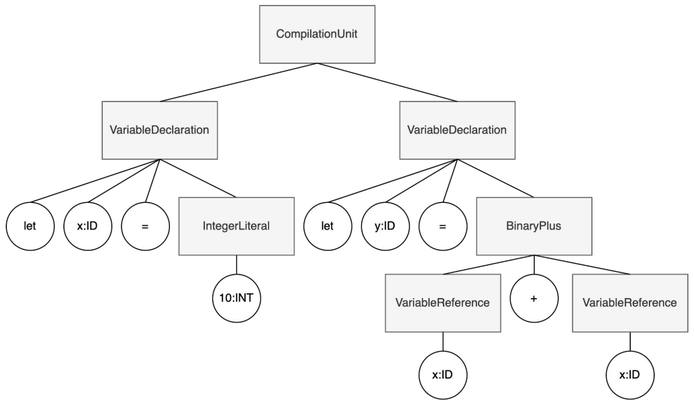
```
compilation_unit:
    variable_declarations+=variable_declaration*
;

variable_declaration:
    'let' name=ID '=' value=expression ';'
;

expression
    : left=expression '+' right=expression  #binary_plus
    | variable=ID                           #variable_reference
    | value=INT                             #integer_literal
    ;
```

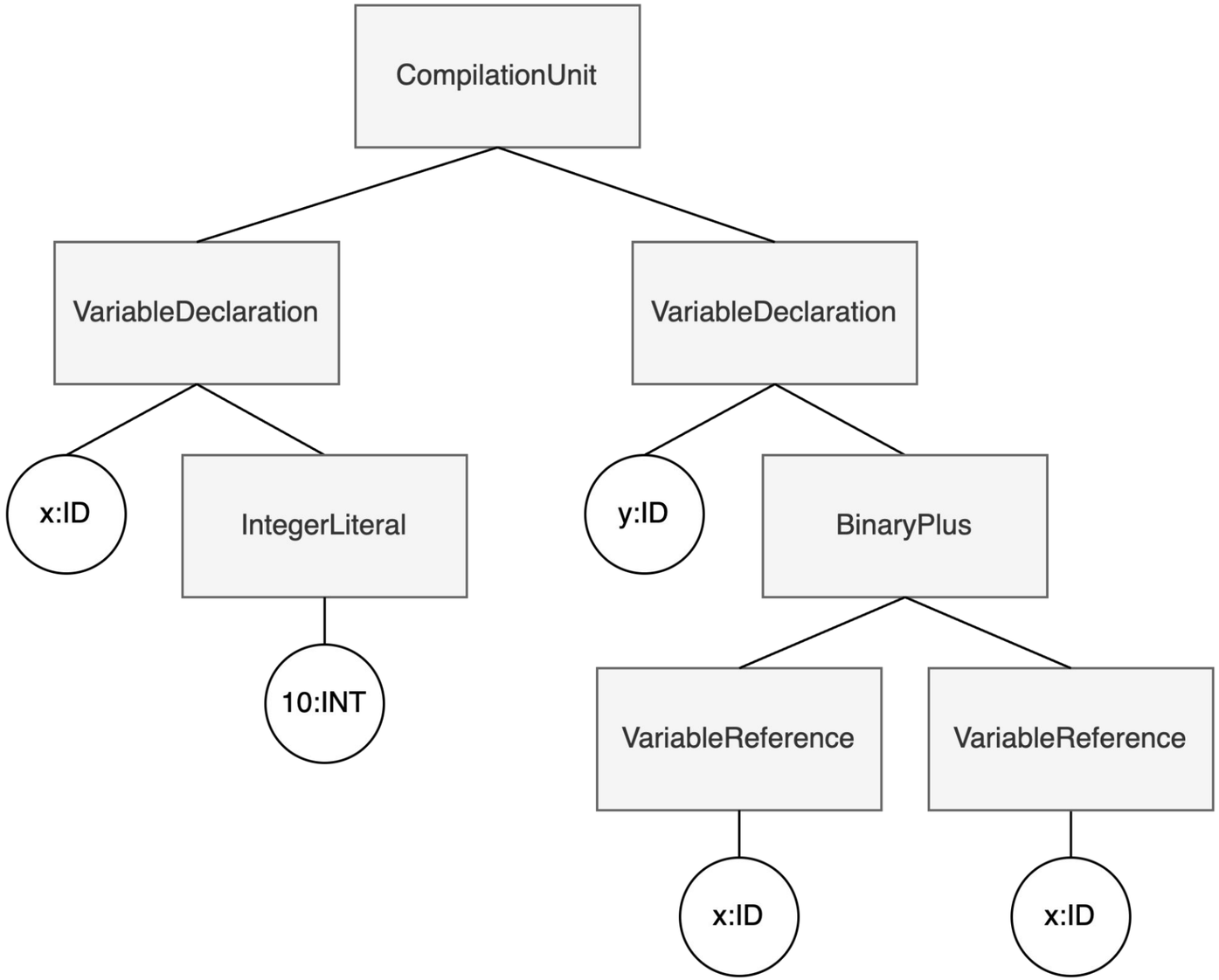
Semantic Enrichment - Definition

```
let x = 10;
let y = x + x;
```

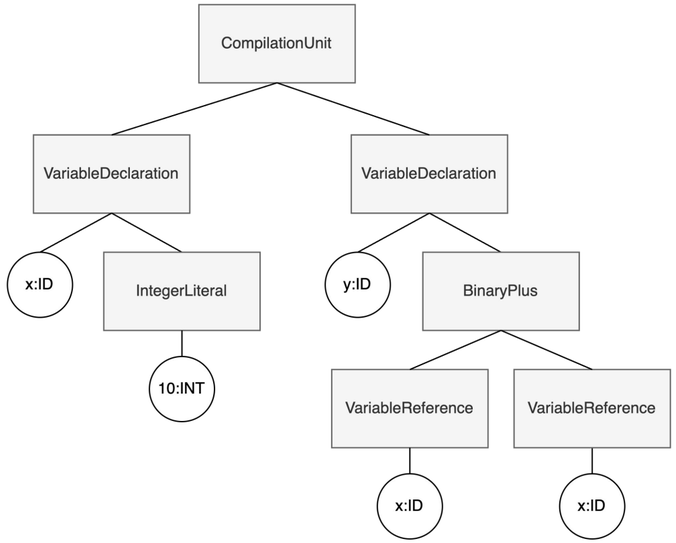


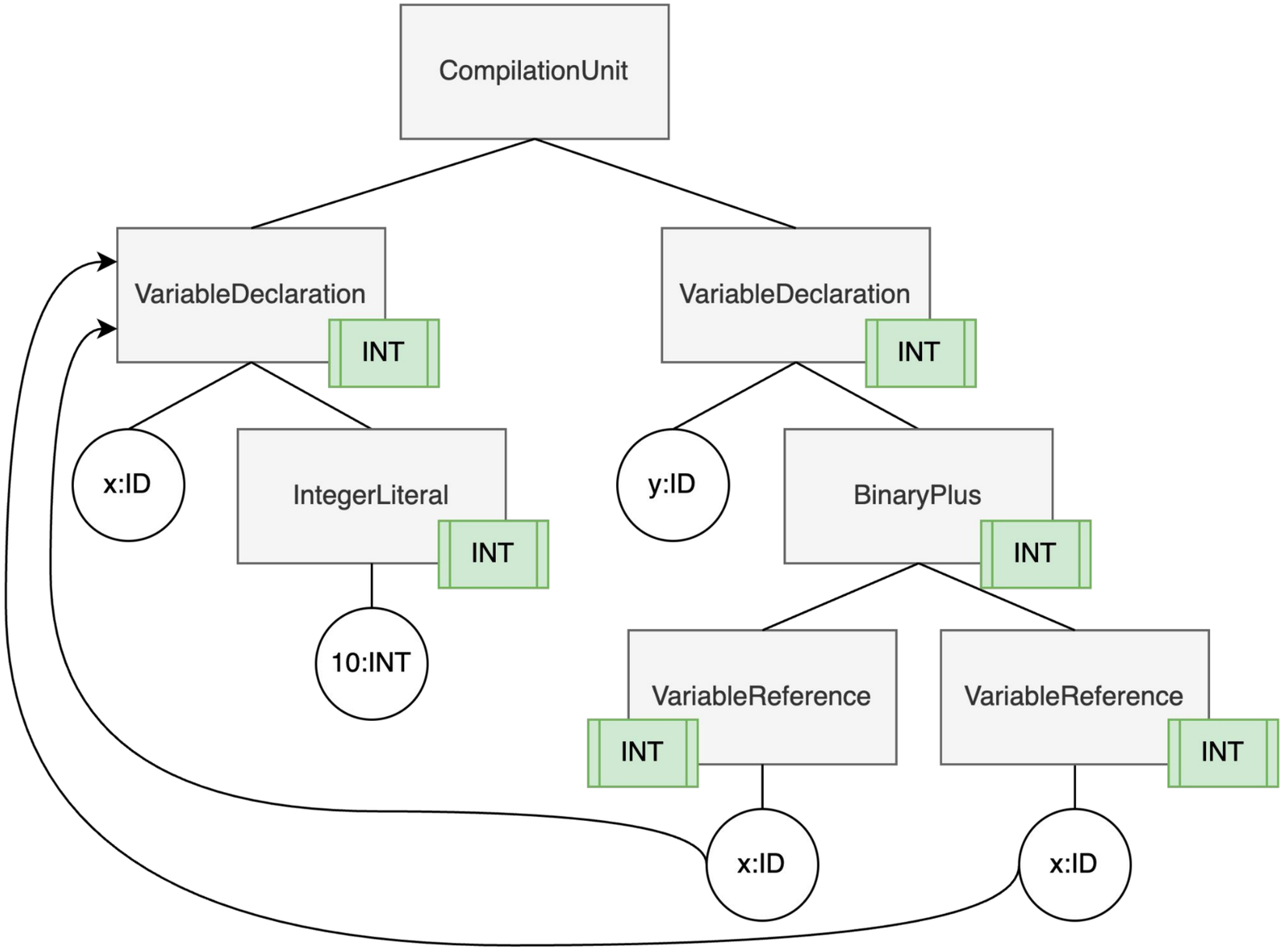
Semantic Enrichment - Definition





Semantic Enrichment - Definition





Semantic Enrichment - Implementation

Declarative specification of symbol resolution and type computation rules.

- Avoid redundant rule evaluations
- Clear separation of concerns
- Transparent support for mixed scenarios

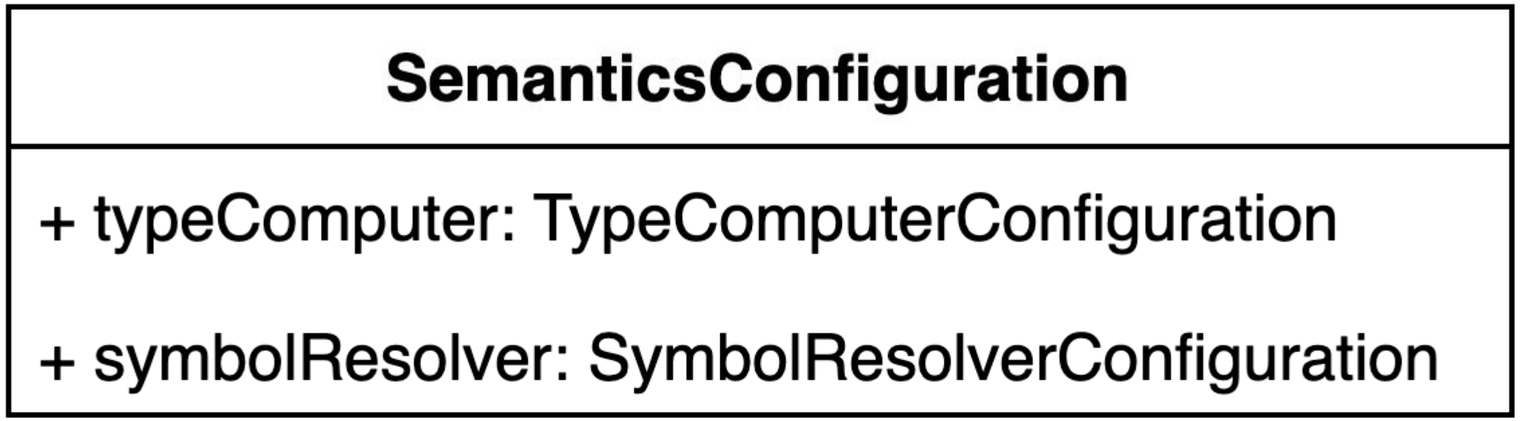
Semantic Enrichment - Implementation

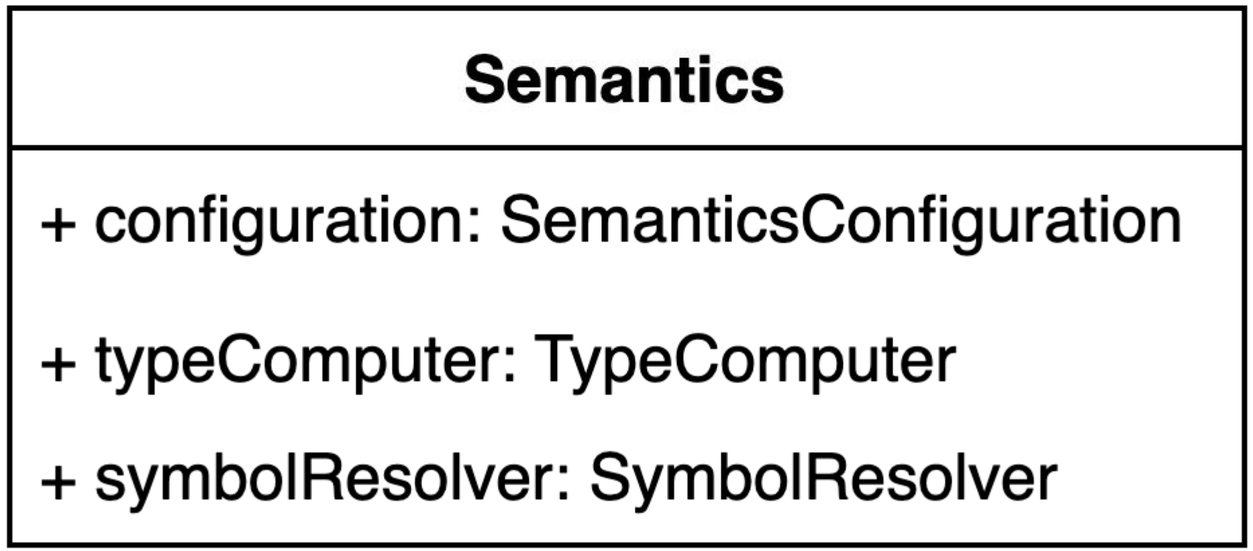
How do represent references between nodes?

```
data class VariableDefinition(
  var name: String
) : Node()

data class VariableReference(
    var variable: ReferenceByName<VariableDefinition>
): Node()
```

```
val x = VariableDefinition(name = "x")

val xRef = VariableReference(variable = ReferenceByName(name = "a"))

xRef.variable.referred = x
```

Semantic Enrichment - Implementation





Semantic Enrichment - Implementation

SymbolResolverConfiguration
+ scopeProvider: ScopeProviderConfiguration
+ scopeFor(referenceByName: KReferenceByName<out Node>, scopeResolutionRule: Semantics.(Node) -> Scope) + scopeFrom(nodeType: KClass<out Node>, scopeConstructionRule: Semantics.(Node) -> Scope)

SymbolResolver
+ loadFrom(configuration: SymbolResolverConfiguration, semantics: Semantics) + resolve(node: Node) + resolve(property: KReferenceByName<out Node>, node: Node) + scopeFor(property: KReferenceByName<out Node>, node: Node?): Scope + scopeFrom(node: Node): Scope

Semantic Enrichment - Implementation

ScopeProviderConfiguration
+ scopeResolutionRules: MutableMap<String, MutableMap<KClass<out Node, Semantics.(Node) -> Scope>>> + scopeConstructionRules: MutableMap<KClass<out Node>, Semantics.(Node) -> Scope>
+ scopeFor(referenceByName: KReferenceByName<out Node>, scopeResolutionRule: Semantics.(Node) -> Scope) + scopeFrom(nodeType: KClass<out Node>, scopeConstructionRule: Semantics.(Node) -> Scope)

ScopeProvider
+ loadFrom(configuration: ScopeProviderConfiguration, semantics: Semantics) + scopeFor(referenceByName: KReferenceByName<out Node>, node: Node): Scope

Semantic Enrichment - Implementation

Scope
+ parent: Scope? + symbolTable: SymbolTable + ignoreCase: Boolean
+ define(symbol: PossiblyNamed) + resolve(name: String?, type: KClass<out PossiblyNamed>): PossiblyNamed?

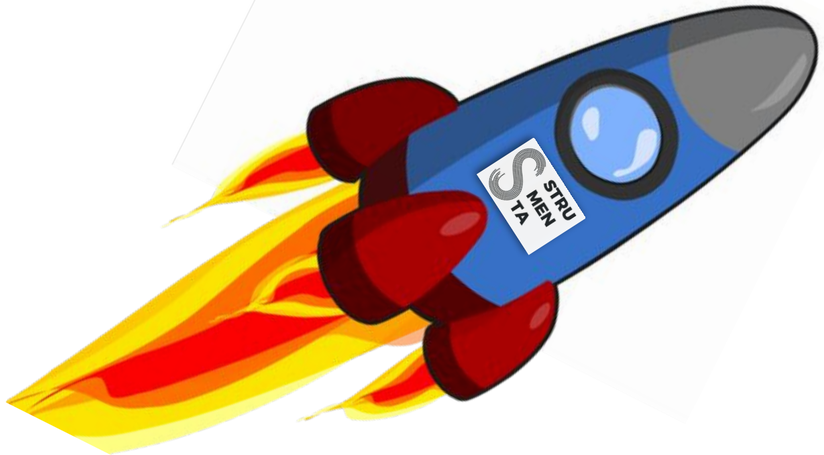
Semantic Enrichment - Implementation

TypeComputerConfiguration
+ typingRules: MutableMap<KClass<out Node>, Semantics.(Node) -> Node?>
+ typeFor(nodeType: KClass<out Node>, typeRule: Semantics.(Node) -> Node?>

TypeComputer
+ loadFrom(configuration: TypeComputerConfiguration, semantics: Semantics)
+ typeFor(node: Node): Node?

DEMO

https://github.com/Strumenta/kolasu-entitv-parser



Questions?

Thank you!

Contact Us

https://twitter.com/strumenta

https://www.linkedin.com/company/strumenta

STRUMENTA COMMUNITY

https://strumenta.community/

https://strumenta.com/

Implementing Symbol Resolution in StarLasu

Lorenzo Addazi

Software Language Engineer @ Strumenta

LangDev'23

15th of November, 2023

Utrecht, The Netherlands

STRU
MEN
TA


STRU
MEN
TA