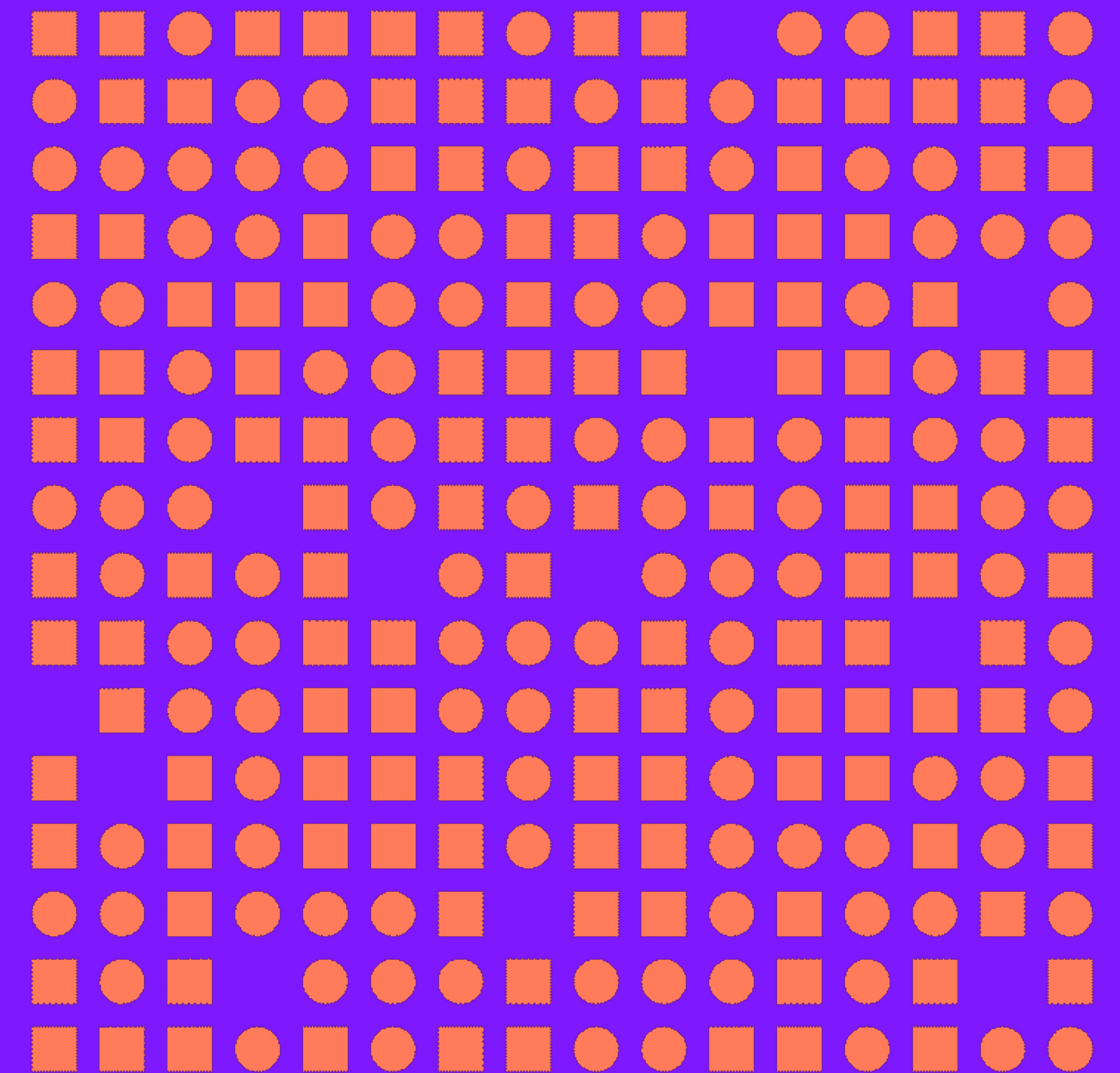


TypeFox

LangDev
CON2026

A generic Serializer for Langium



Johannes Meier



Langium

 TypeScript

Quick-start:

- langium.org/playground
- `yo langium`

Language workbench
for textual DSLs

- LL(*) parsing
- Typed AST
- Cross-references
- Validation

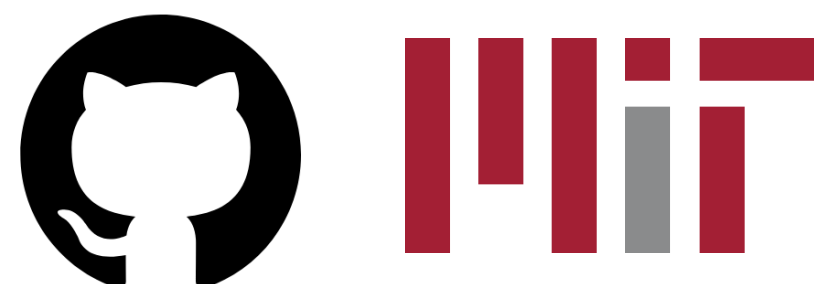
built-in LSP support



Statistics (July 2026)

- >4M weekly downloads
- >370 GitHub repos
- >25 commercial projects

Open source



Governance

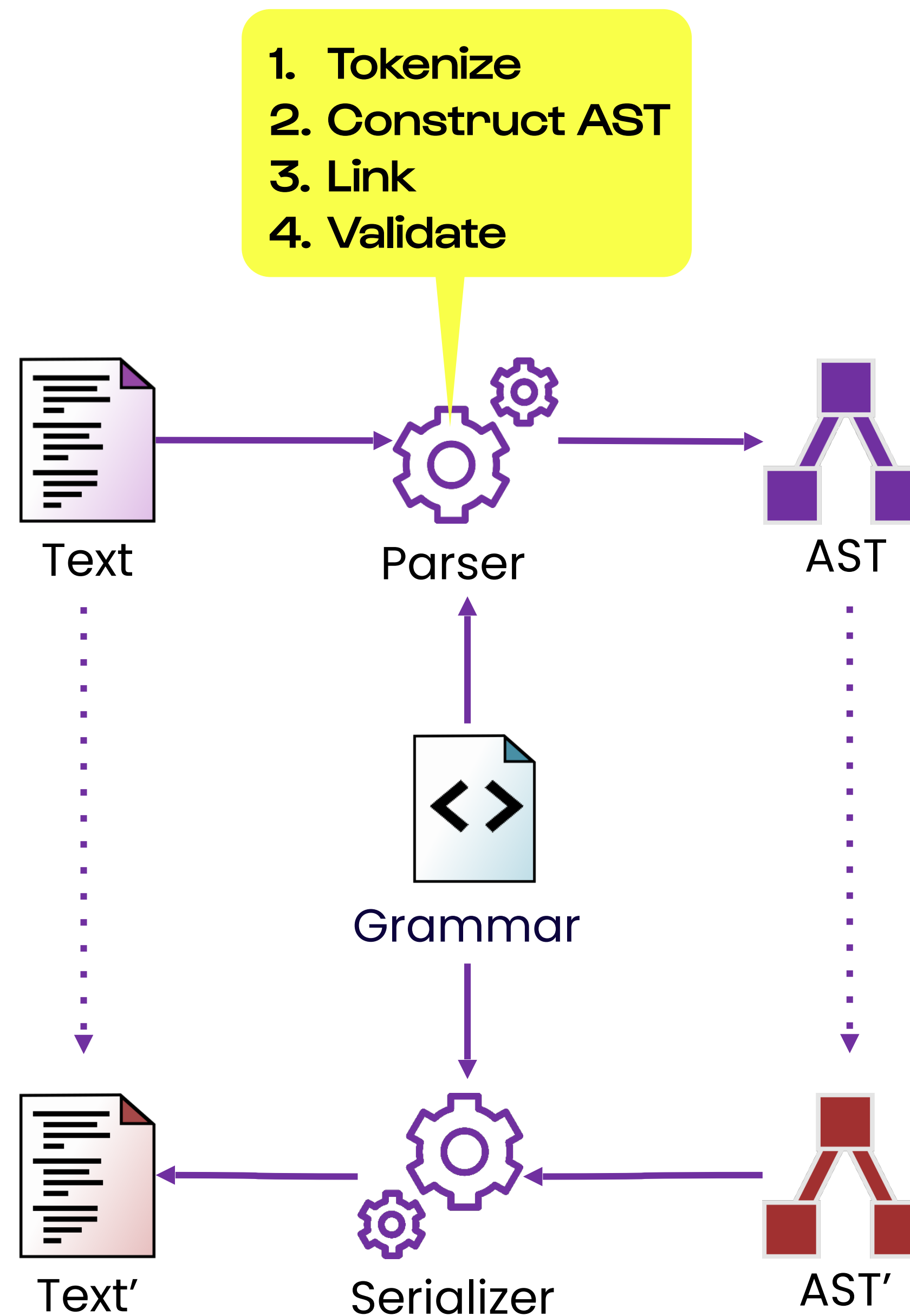
- since 2021 developed by **TypeFox**
- since 2023 hosted by **ECLIPSE**
FOUNDATION



Serialization

Changed AST (use case 2)

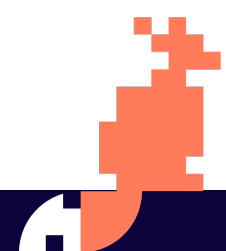
- Refactorings
- Code actions
- Enrich with additional information



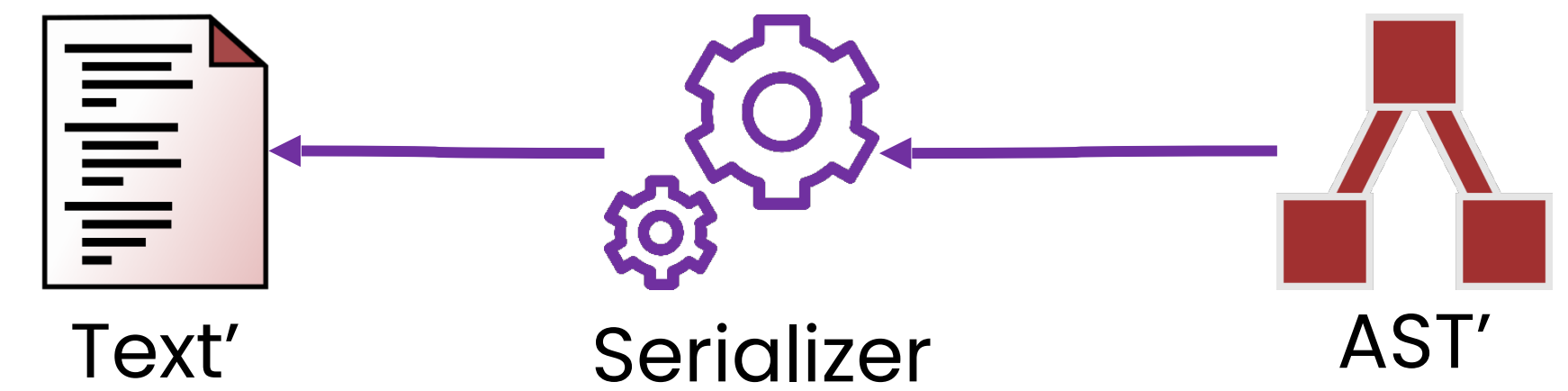
Focus
today

New AST (use case 1)

- Non-textual or hybrid editors
- Derived from databases, AI, ...
- Transformations between languages



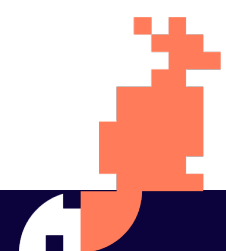
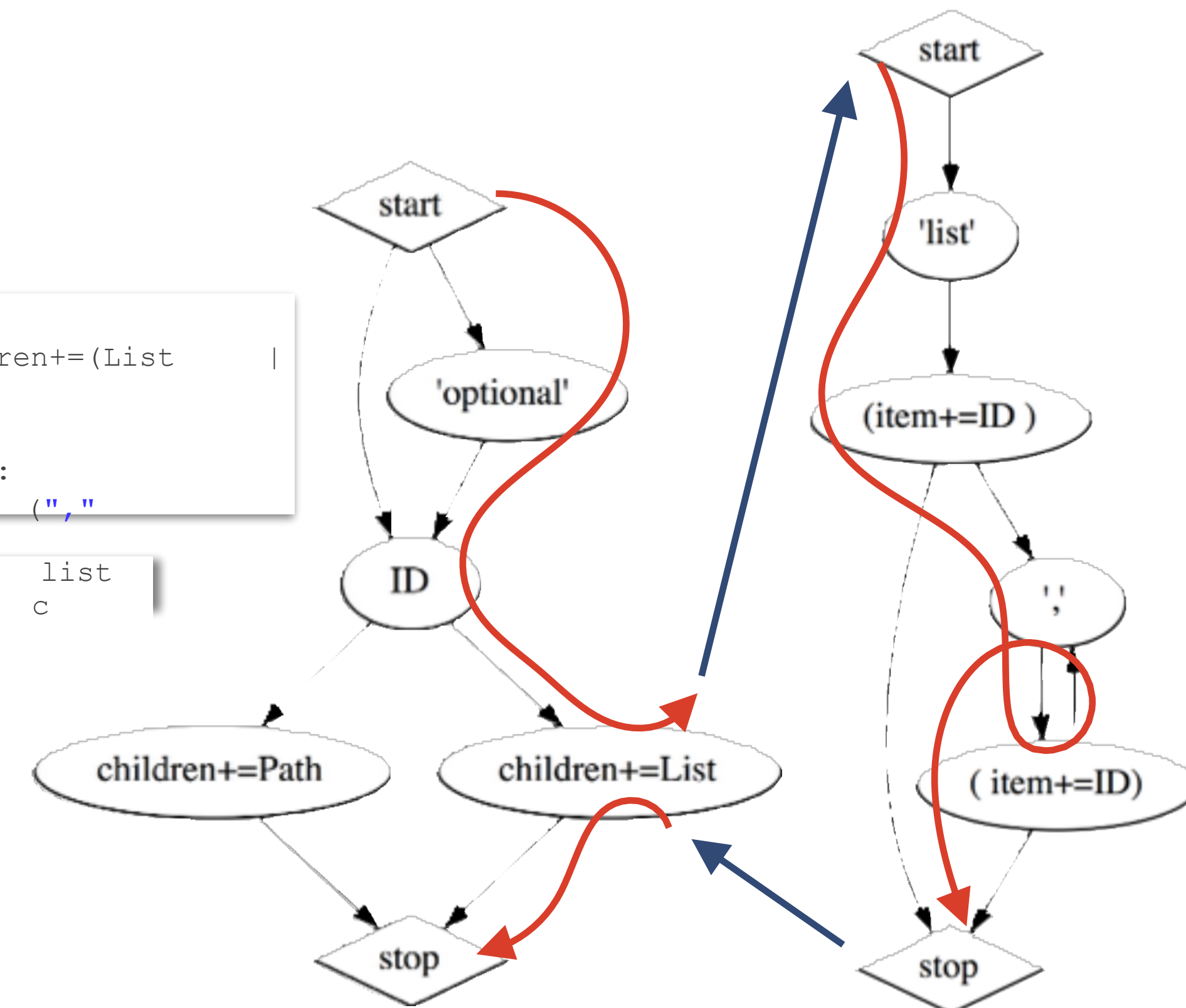
Design objectives



- “Generic”: Provide default serialization for a given grammar out-of-the-box
- “Project-specific”: Customize the default serialization
- Lessons learned from the Xtext serializer:
 - Performance
 - Understanding

```
Root:
  "optional"? ID      children+=(List
  Path);

List returns Child:
  "list" item+=ID      (","
  item+=ID)*;
Output: optional      Foo      list
       a,             b,       c
```

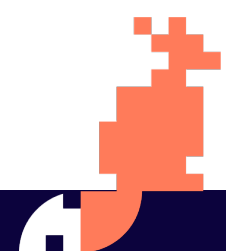
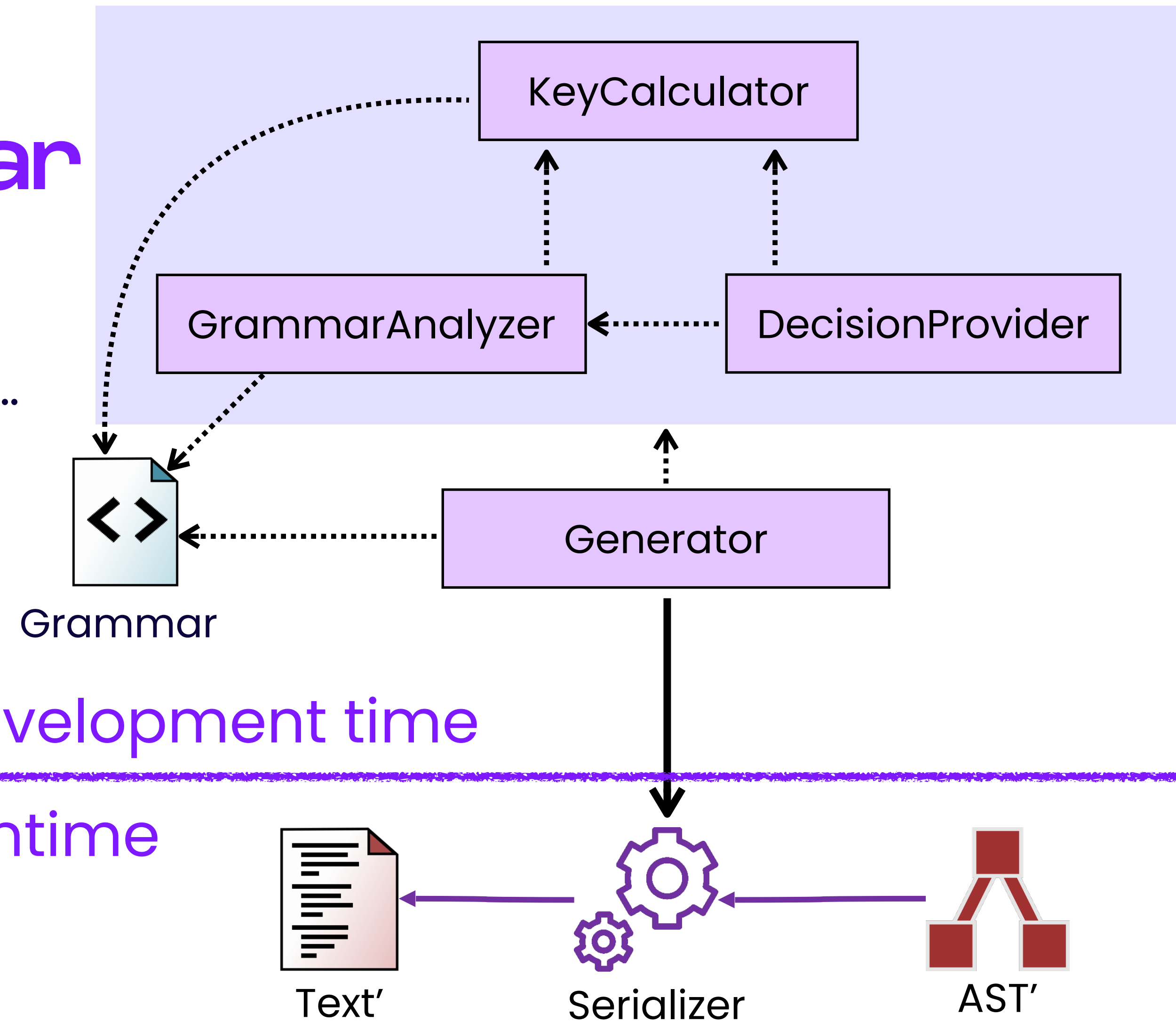




Live demo

Traversing the grammar

- For fast serialization of a given AST ...
 - don't plan the paths in advance, but ...
 - decide directly at the decision!
 - Loop (*, +)
 - Optional (?)
 - Alternatives (|)
- Use pre-calculated decisions
- Analyse the grammar once before, not each AST during serialization



Decision example for loop

entry Model:
persons+=Person*
greetings+=Greeting*
;

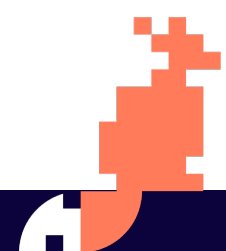


Grammar

```
serialize_Model_02_Assignment_persons(context: Context<Model>): Result<Model> {  
  const result: Generated[] = [];  
  let current: Result<Model> = { generated: undefined, context };  
  const visitLooped = serialize_Model_02_Assignment_persons_Loop();  
  while (visitLooped(current.context)) {  
    current = serialize_Model_02_Assignment_persons_Body(current.context);  
    result.push(current.generated);  
  }  
  return { generated: this.joinGenerated(result.reverse()), context: current.context };  
}
```

```
serialize_Model_02_Assignment_persons_Loop(): VisitLoopDecision<Model, Model> {  
  return (context) => context is Context<Model> => {  
    const done = context.serializedValuesForFeatures.get('persons') ?? 0;  
    return done < context.node.persons.length;  
  }  
}
```

"counter" feature



Decision example for alternatives

different cardinalities of the “values” property

Model:

```
values+=ID
```

```
(
```

```
'a' values+=ID
```

```
|
```

```
'b' values+=ID values+=ID
```

```
)
```

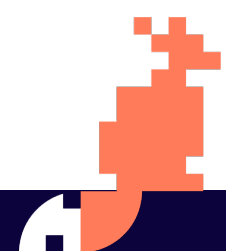
```
values+=ID
```

```
;
```



Grammar

```
serialize_Model_04_Alternatives(context: Context<Model>): Result<Model> {  
  if (context.node.values.length === 3) {  
    return serialize_Model_05_Group(context);  
  }  
  if (context.node.values.length === 4) {  
    return serialize_Model_09_Group(context);  
  }  
  throw new Error('None of the decisions matches');  
}
```



Serialize backwards

value of the "op" property

```
serialize_Add_03_Group_Loop(): VisitLoopDecision<Expr | Num, Expr> {  
  return (context): context is Context<Expr> => {  
    return context.node.$type === 'Expr' && context.node.op === '+';  
  }  
}
```

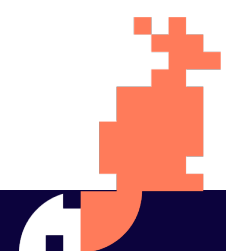
```
Add: Mul ({infer Expr.left=current} op='+' right=Mul)*;  
Mul: Num ({infer Expr.left=current} op='*' right=Num)*;  
Num: c=INT;
```



Grammar

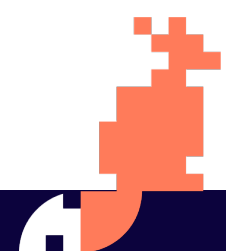
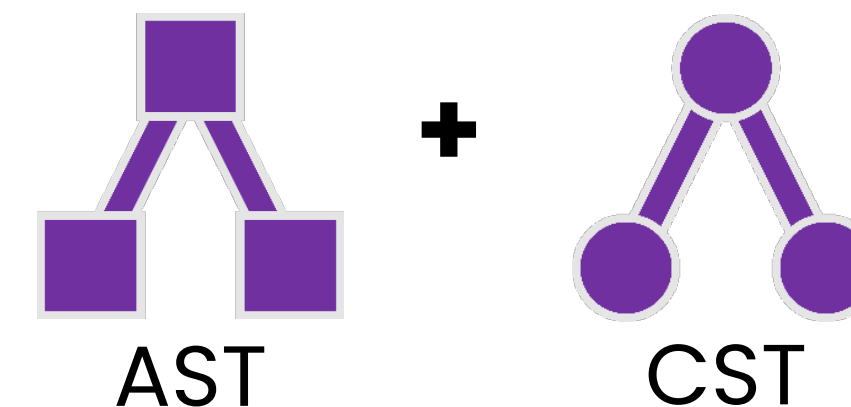
\$type of the **current** AstNode,
no need to revert the tree rewrite action now

```
serialize_Add_02_RuleCall_03_Group_Loop(): VisitLoopDecision<Expr | Num, Expr> {  
  return (context): context is Context<Expr> => {  
    return context.node.$type === 'Expr';  
  }  
}
```



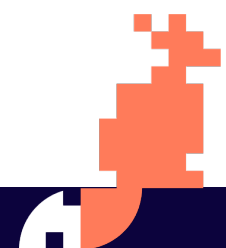
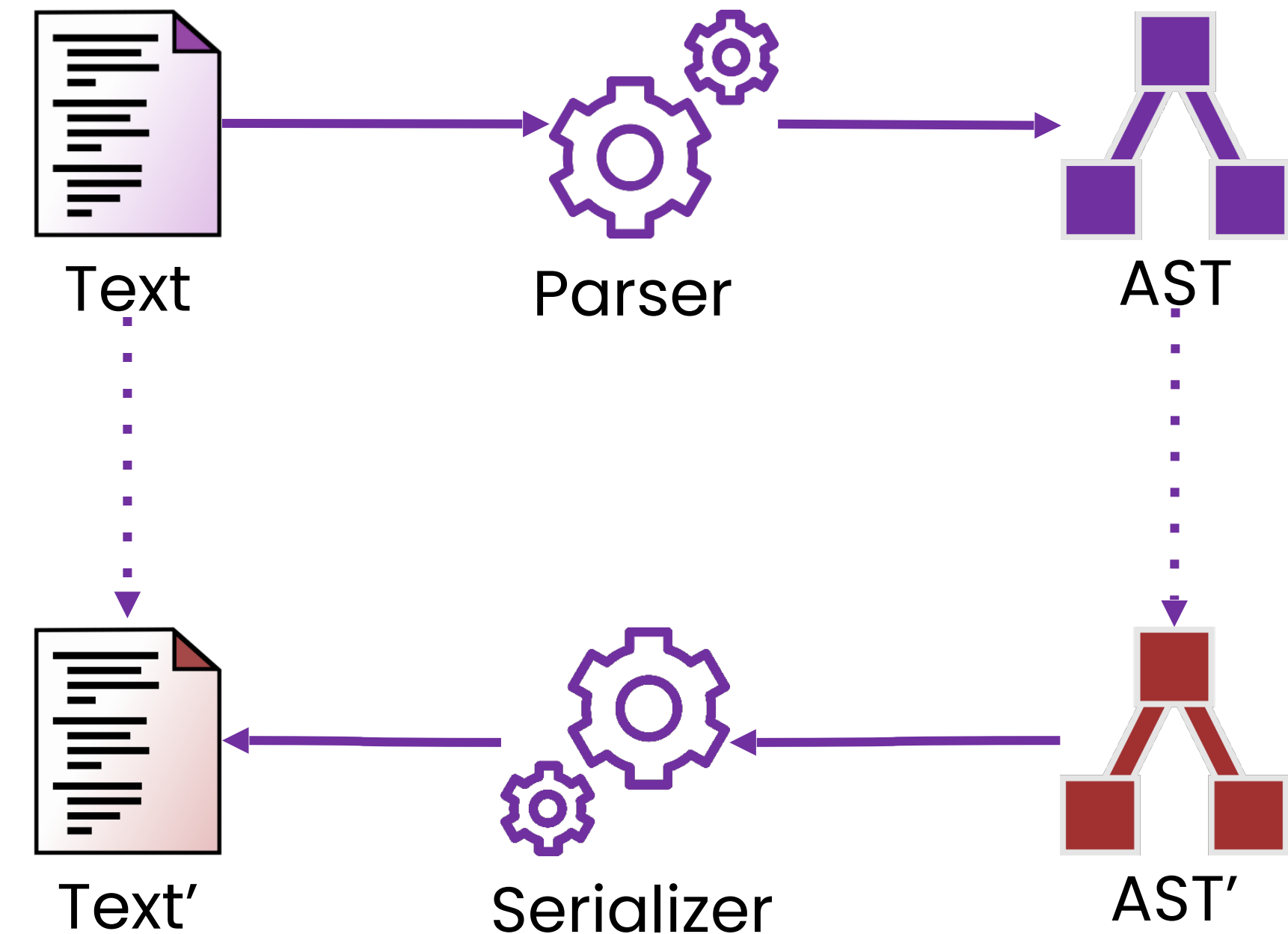
Roadmap

- Version 0.1: Serialize new AST, without CST (use case 1)
- Heuristics for serializer ambiguities
 - e.g. optional keyword, looped alternatives, unordered groups, ...
- Serialize changed AST with CST to keep comments and formatting (use case 2)
 1. Don't print tokens directly, but produce CST instead
 2. During serialization, traverse AST **and** existing CST together: add missing CstNodes, fix existing CstNodes
 3. "Pretty-print" the resulting CST



Summary

- Default serialization for given grammars
- Generated serializer code for easier ...
 - understanding
 - debugging
 - customising
- Fast approach
 - AST is not analyzed in advance
 - Decide directly on criteria derived from the grammar (\$type, assignments)



TypeFox

 LangDev
CON2026

Questions & Discussions!

typefox.io

