

Traqula: Providing a foundation for the evolving SPARQL Ecosystem through modular Parsing, Transformation and Generation



Jitse De Smet



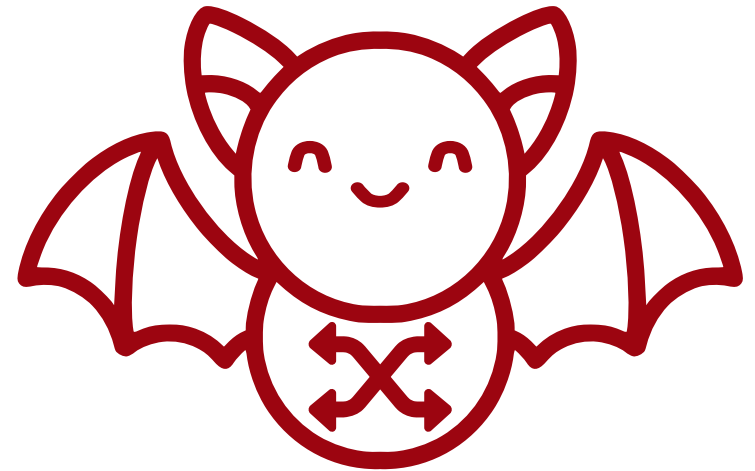
Ruben Taelman

Ghent University – imec – IDLab, Belgium

Research Foundation - Flanders



Paper at:
traqula-resource.jitsedesmet.be



TRAQULA

Available on npm, TypeScript and JavaScript, ESM and CJS
Managed by the Comunica Association, already used within Comunica

Overview

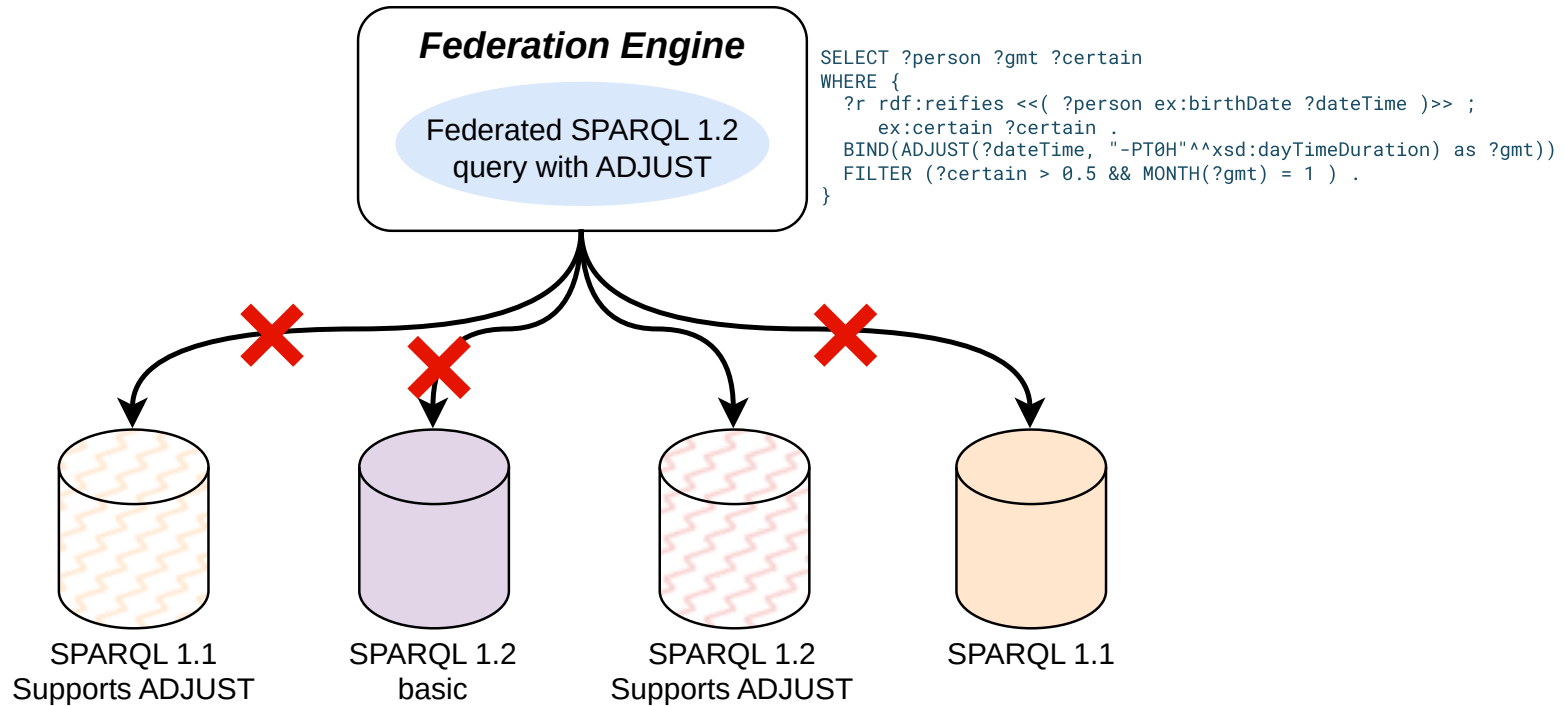
Problem

Traqula

Performance



One Query — Many Languages



TRAQULA

Bridging is essential

Three key challenges

Query Evaluation

A SPARQL 1.2 query may fail on one endpoint and work on another

SPARQL 1.2 query
→ sent to SPARQL 1.0 endpoint
→ **Error**

Tooling

Linters, editors, and formatters assume a single SPARQL version

IDE, linter, language server
→ built for SPARQL 1.1
→ **breaks on SPARQL 1.2 syntax**

Maintainability

Multi-version support
→ messy, hard-to-maintain code

```
if SPARQL_1_1 ...  
else if SPARQL 1.2  
→ duplicate code  
→ Fragile codebase
```

SPARQL 1.2 and the expected maintenance mode will push new features, likely causing language diversity to grow.



Overview

Problem

Traqula

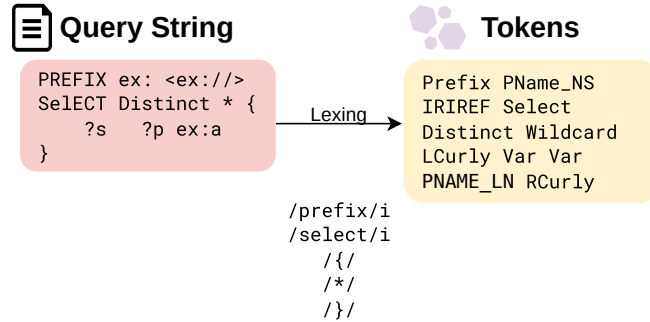
Performance



Demystifying parsers



Demystifying parsers



Demystifying parsers



Query String

```
PREFIX ex: <ex://>
SELECT Distinct * {
  ?s ?p ex:a
}
```

Lexing →

```
/prefix/i
/select/i
/{}
/*
/}/
```

Tokens

```
Prefix PName_NS
IRIREF Select
Distinct Wildcard
LCurly Var Var
PNAME_LN RCurly
```

Parsing →

AST

```
subType: select
distinct: true
loc: (location info)
context:
  contextDef (list)
    subtype: prefix
    key: ex
    value: 'ex://'
    loc: (location info)
where:
  pattern
    subtype: group
    loc: (location info)
    patterns:
      pattern (list)
        subType: bgp
        loc: (location info)
        triples:
          triple
            subject:
              type: term
              subType: variable
              value: s
              loc: (location info)
            predicate: ...
            object: ...
            loc: (location info)
variables:
  wildcard
    loc: (location info)
```

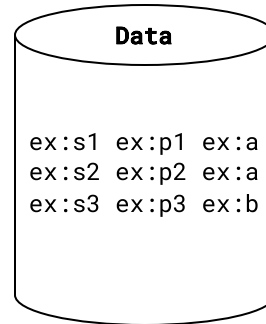


Demystifying parsers



$f(x)$ Algebra

```
distinct
input:
  project
  variables:
    - termType: Variable
      value: s
    - termType: Variable
      value: p
  input:
    bgp
    patterns:
      pattern (list)
      termType: Quad
      subject:
        termType: Variable
        value: s
      predicate:
        termType: variable
        value: p
      object:
        termType: NamedNode
        value: 'ex://a'
```



$f(x)$ Algebra

```
distinct
input:
  project
  variables:
    - termType: Variable
      value: s
    - termType: Variable
      value: p
  input:
    bindings
    - ?s:
        termType: NamedNode
        value: 'ex://s1'
      ?p:
        termType: NamedNode
        value: 'ex://p1'
    - ?s:
        termType: NamedNode
        value: 'ex://s2'
      ?p:
        termType: NamedNode
        value: 'ex://p2'
```



Demystifying parsers



$f(x)$ Algebra

distinct

input:

project

variables:

- termType: Variable
value: s
- termType: Variable
value: p

input:

bindings

- ?s:
termType: NamedNode
value: 'ex://s1'
?p:
termType: NamedNode
value: 'ex://p1'
- ?s:
termType: NamedNode
value: 'ex://s2'
?p:
termType: NamedNode
value: 'ex://p2'

$f(x)$ Algebra

distinct

input:

bindings

- ?s:
termType: NamedNode
value: 'ex://s1'
?p:
termType: NamedNode
value: 'ex://p1'
- ?s:
termType: NamedNode
value: 'ex://s2'
?p:
termType: NamedNode
value: 'ex://p2'

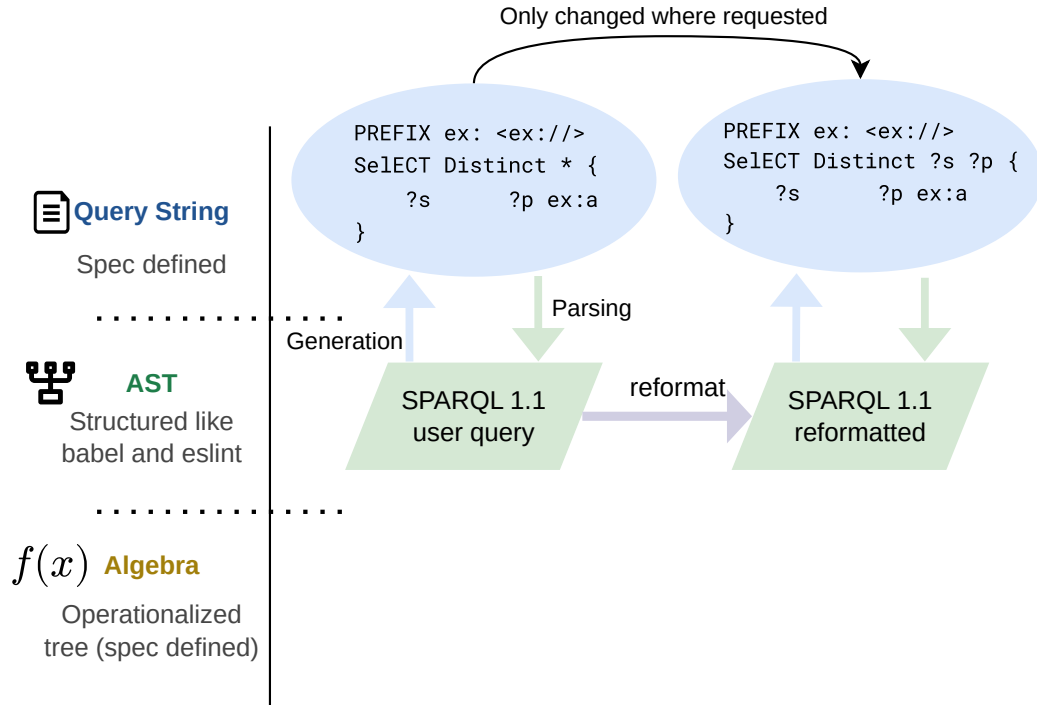
$f(x)$ Algebra

bindings

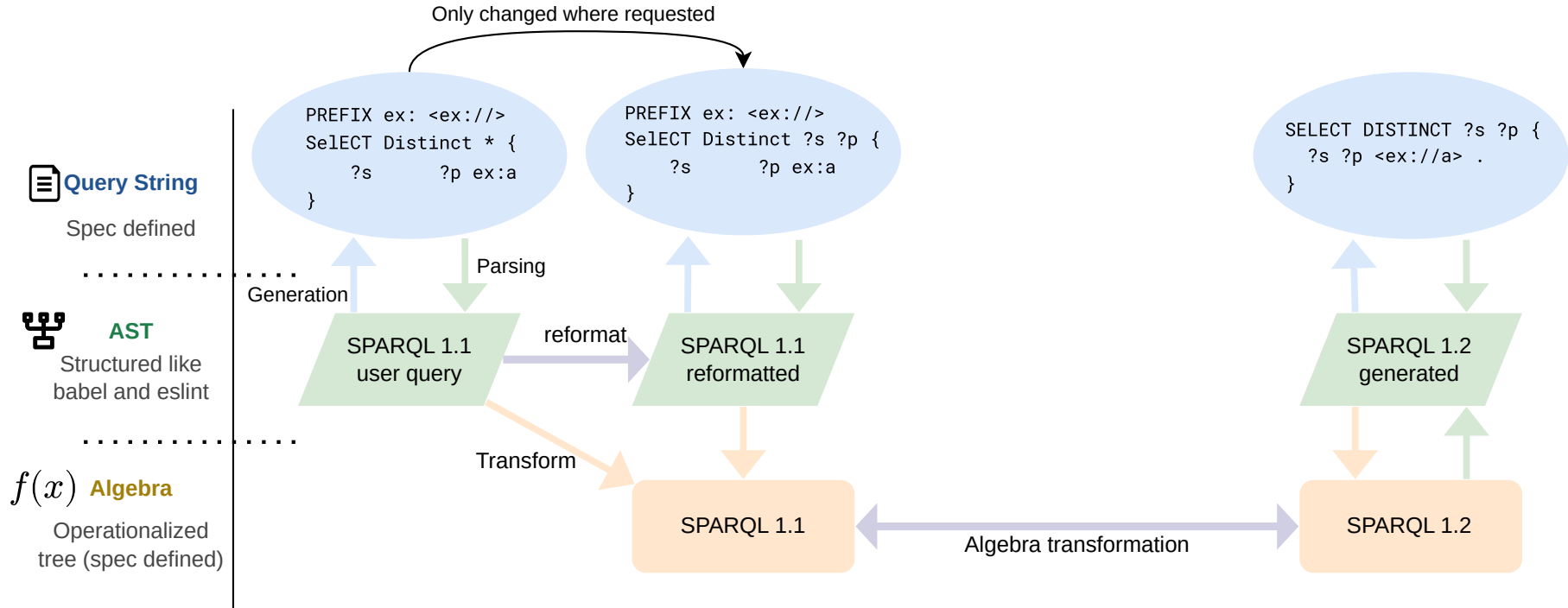
- ?s:
termType: NamedNode
value: 'ex://s1'
?p:
termType: NamedNode
value: 'ex://p1'
- ?s:
termType: NamedNode
value: 'ex://s2'
?p:
termType: NamedNode
value: 'ex://p2'



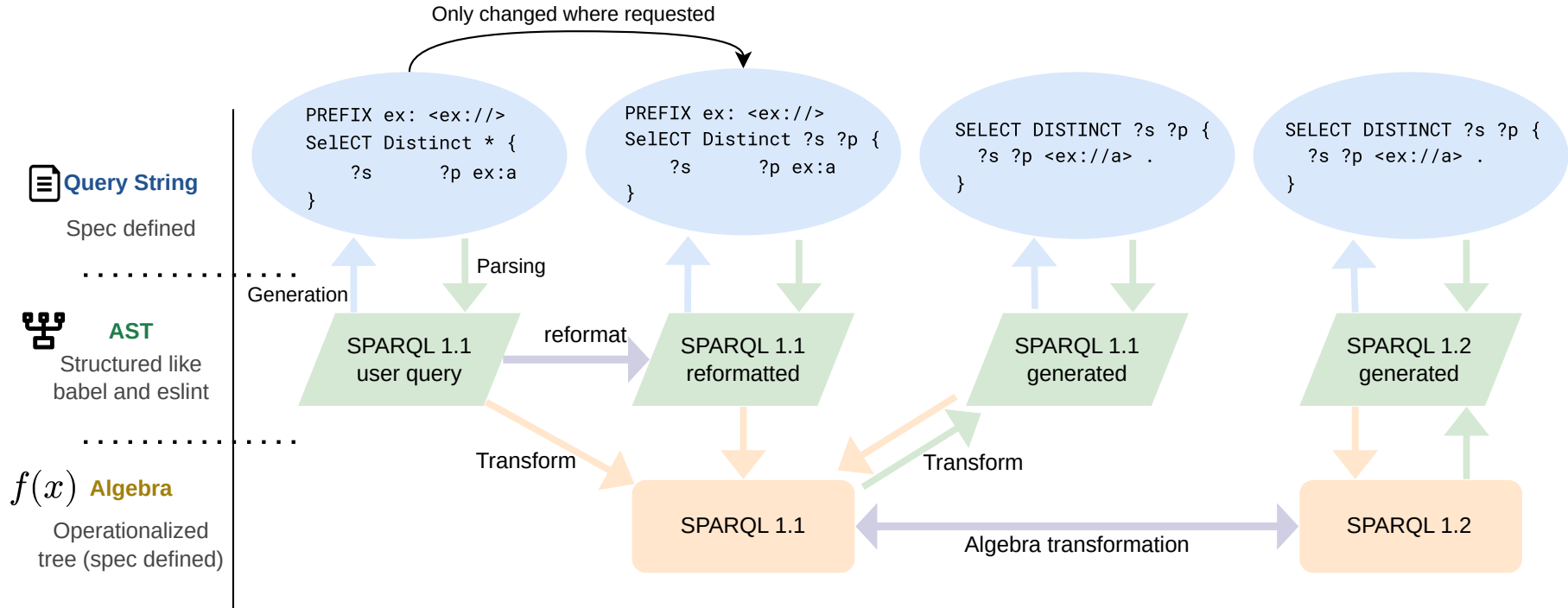
Transform across query string, AST, and algebra



Transform across query string, AST, and algebra



Transform across query string, AST, and algebra

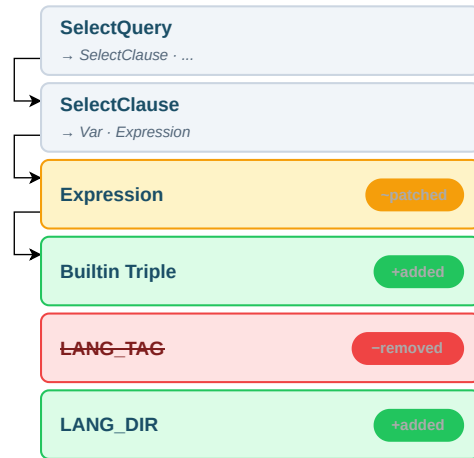


Flexibility through rule composition

www.w3.org/TR/sparql12-query/#rSelectQuery

[9]	SelectQuery	::=	SelectClause DatasetClause* WhereClause SolutionModifier
[10]	SubSelect	::=	SelectClause WhereClause SolutionModifier ValuesClause
[11]	SelectClause	::=	'SELECT' ('DISTINCT' 'REDUCED')? ((Var ('(' Expression 'AS' Var ')'))+ '*')
[12]	ConstructQuery	::=	'CONSTRUCT' (ConstructTemplate DatasetClause* WhereClause SolutionModifier DatasetClause* 'WHERE' '{' TriplesTemplate? '}' SolutionModifier)

Rule Dictionary



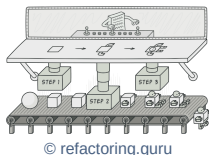
TypeScript Example

```
const parserBuilderSparql1_2 = ParserBuilder
  .create(parserBuilderSparql1_1)
  .patchRule(expression)
  .addRule(builtInTriple)
  .removeRule(langTag)
  .addRule(langDir)

const parser = parserBuilderSparql1_2
  .build({ tokenVocabulary: myTokens })

const astQuery = parser
  .queryOrUpdate(`SELECT * { ?s ?p ?o }`)

const ast = parser
  .expression(`TRIPLE ( ex:s ex:p ex:o )`)
```



Compose parsers, generators & transformers
Facilitating language experimentation & maintainability

Overview

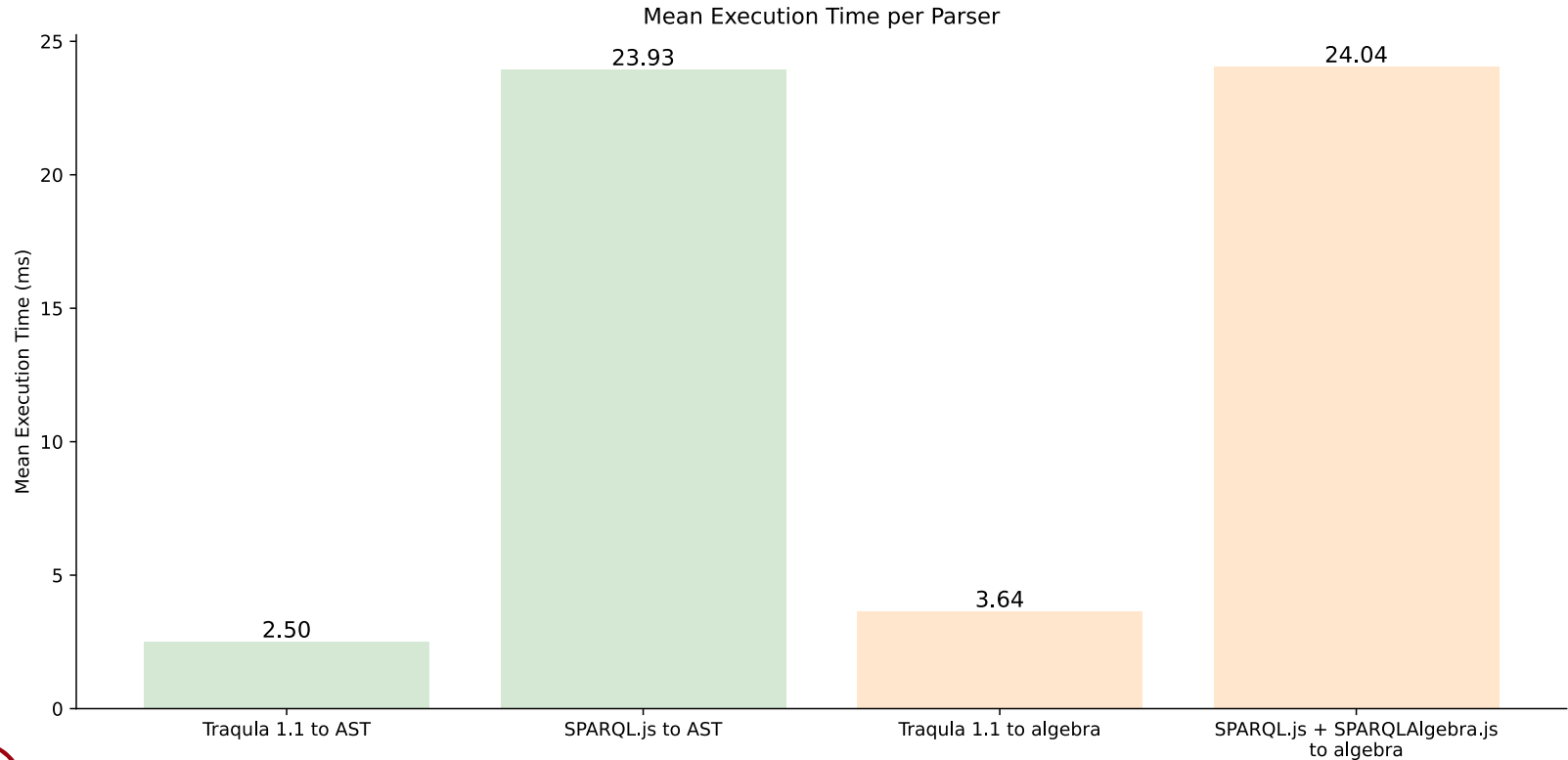
Problem

Traqula

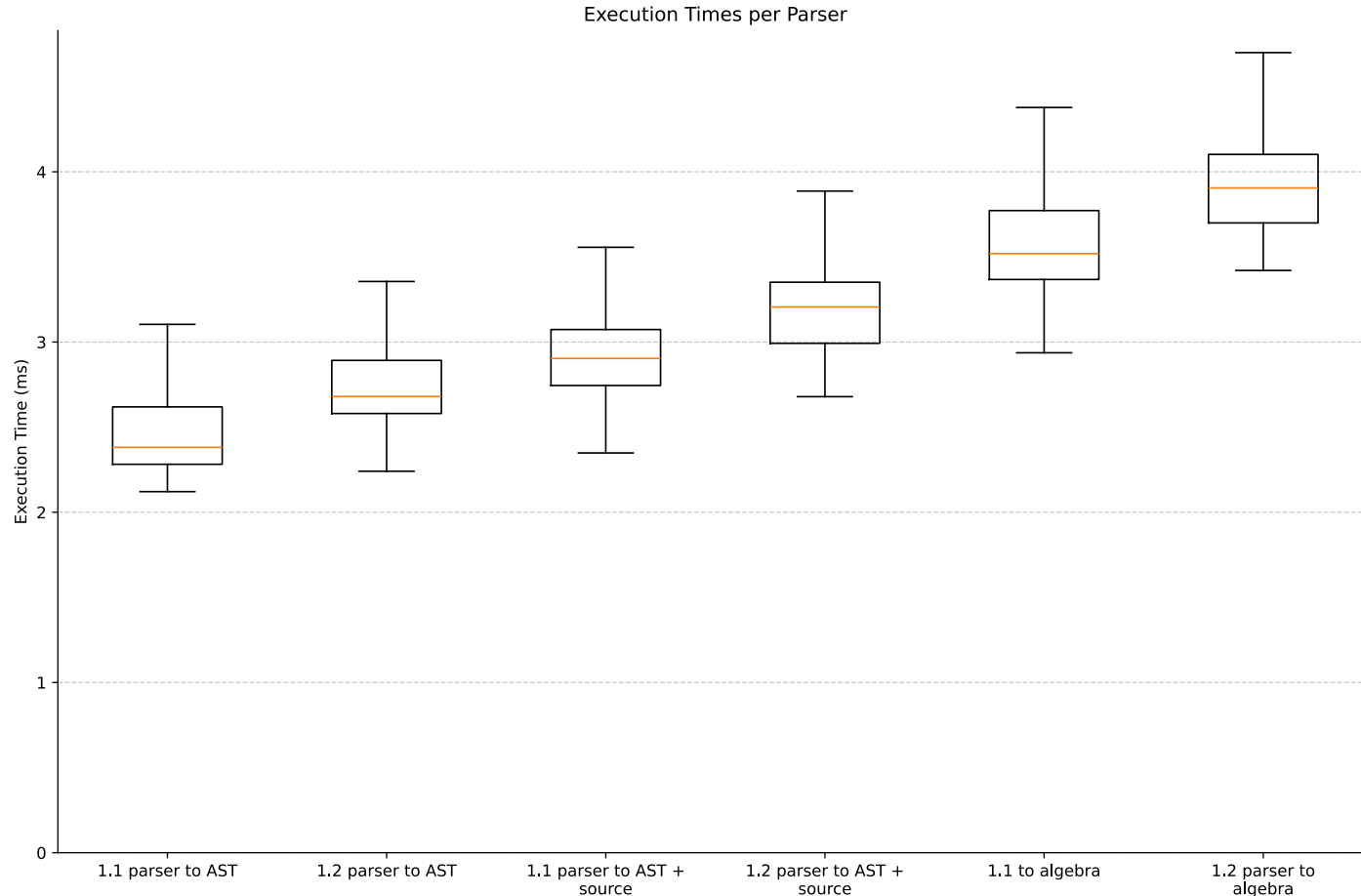
Performance



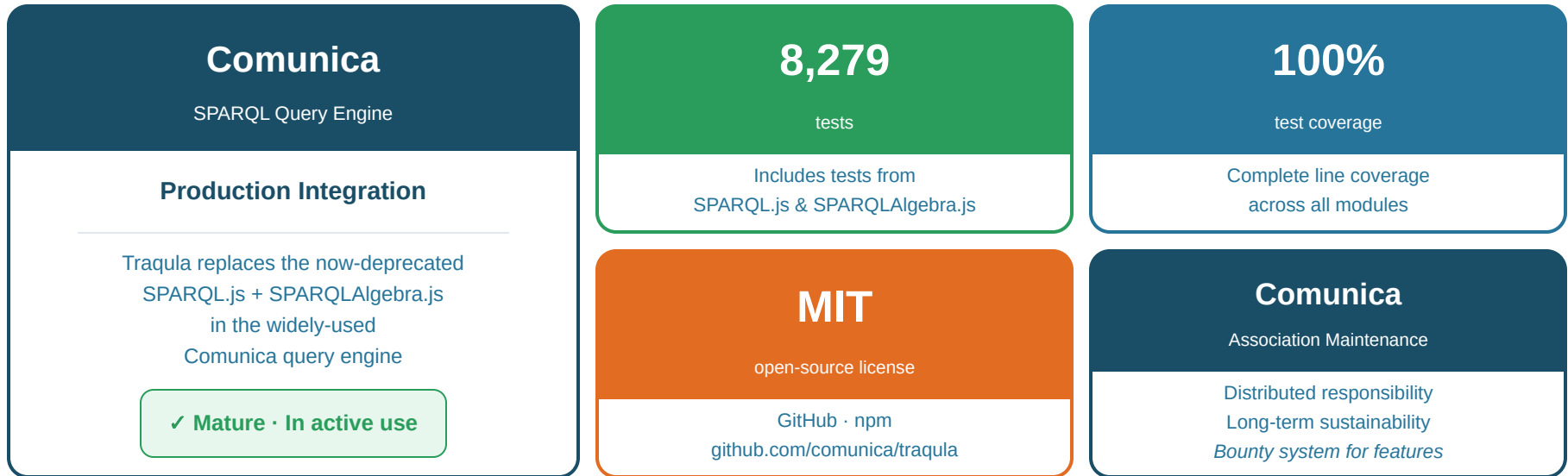
~10x faster than JavaScript/ TS tools



Language and parse targets affect speed



Already in production



Conclusion

Problem

SPARQL language heterogeneity is a **growing challenge** for query evaluation, tooling, and maintainability

Solution

Traqula: **flexible** · round-trippable · web-native · language-agnostic

Impact

Foundation for a unifying framework across languages and dialects. Ready for linters, formatters, LSPs, and future query languages



Also check out our demo

compose-parser-engine.demo.jitsedesmet.be

