Shaping semantic models with Langium

Irina Artemeva

What is Langium?

- Language engineering framework
- Spiritual successor to Xtext
- TypeScript + NodeJS
- Powered by Chevrotain
- High out-of-box functionality

Features of Langium

- Cross-references
- Workspace Management
- Language Server Protocol
- Semantic Model

Semantic Model

TypeFox

Content

- What is a semantic model?
- What to use a semantic model for?
- How does a semantic model look like in Langium?
- How Langium shapes a semantic model?
- **Demo**: how can I use a semantic model?
- Comparison with Xtext

Content

- What is a semantic model?
- What to use a semantic model for?
- How does a semantic model look like in Langium?
- How Langium shapes a semantic model?
- **Demo**: how can I use a semantic model?
- Comparison with Xtext

Semantic Model

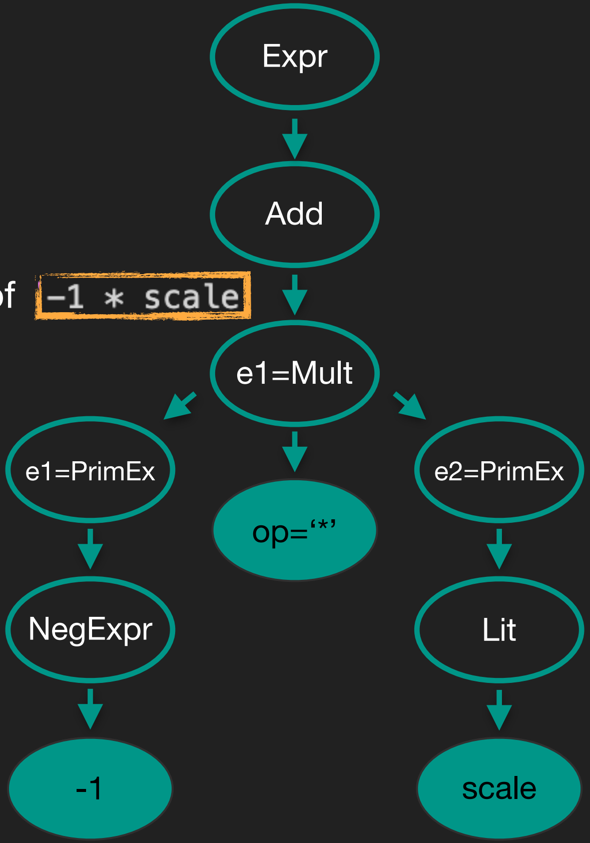
miniLogo source

```
def square(x, y, scale) {
    ...
    move(-1 * scale,0)
```

grammar of miniLogo expressions

```
Expr: Add;

Add:
    e1=Mult (op=('+'|'-') e2=Mult)*;
Mult:
    e1=PrimExpr (op=('*'|'/') e2=PrimExpr)*;

PrimExpr: Lit | Ref | Group | NegExpr;
```

AST of `-1 * scale`

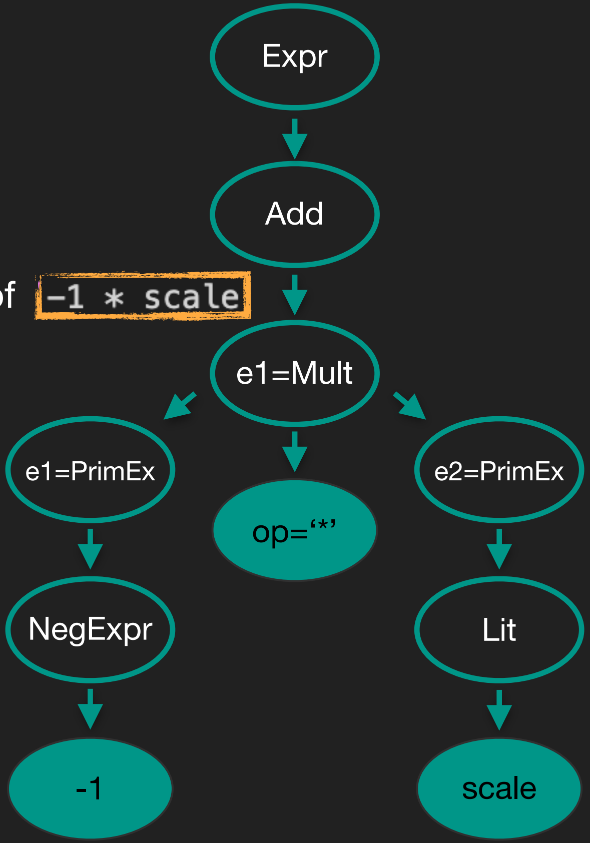


Semantic Model

semantic model of the miniLogo expressions

```
export type Expr = Add;

export interface Add extends AstNode {
    e1: Mult
    e2?: Mult
    op?: '+' | '-'
}

export interface Mult extends AstNode {
    e1: PrimExpr
    e2?: PrimExpr
    op?: '*' | '/'
}

export type PrimExpr = Group | Lit | NegExpr | Ref;
```

AST of `-1 * scale`



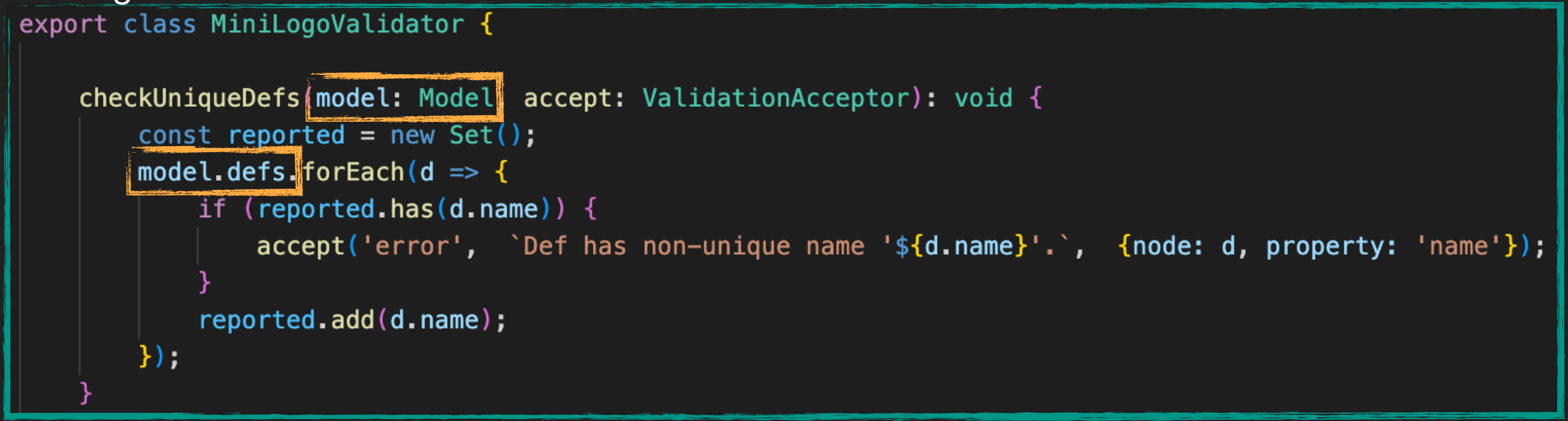
Content

- What is a semantic model?
- What to use a semantic model for?
- How does a semantic model look like in Langium?
- How Langium shapes a semantic model?
- **Demo**: how can I use a semantic model?
- Comparison with Xtext

Motivation for a Semantic Model

- Navigate over an AST
- Fix an AST structure for services implementation

miniLogo Validator



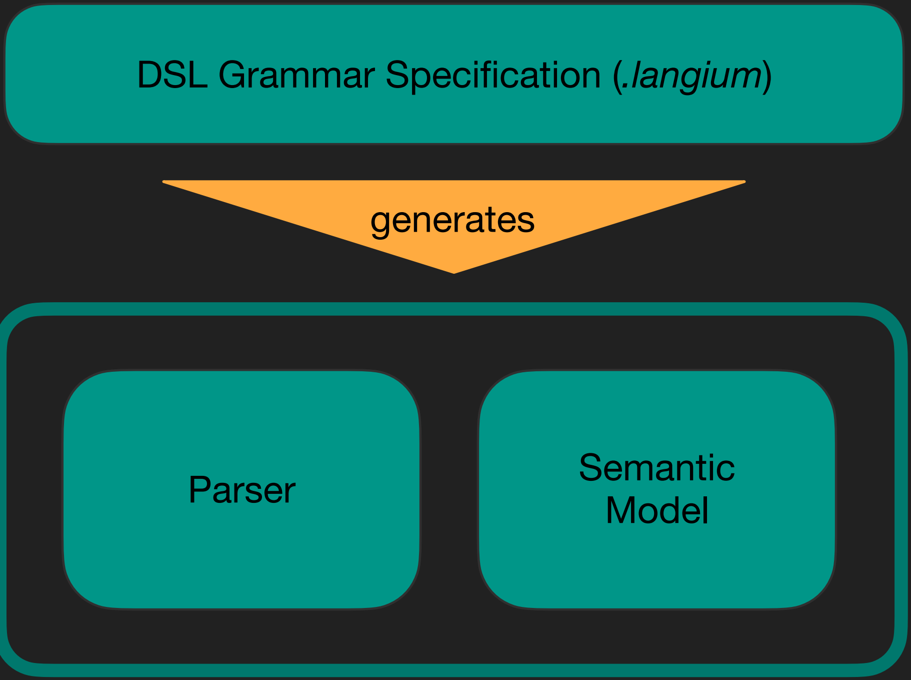
```
export class MiniLogoValidator {

    checkUniqueDefs(model: Model, accept: ValidationAcceptor): void {
        const reported = new Set();
        model.defs.forEach(d => {
            if (reported.has(d.name)) {
                accept('error', `Def has non-unique name '${d.name}'.`, {node: d, property: 'name'});
            }
            reported.add(d.name);
        });
    }
```

TypeFox

Content

- What is a semantic model?
- What to use a semantic model for?
- How does a semantic model look like in Langium?
- How Langium shapes a semantic model?
- **Demo**: how can I use a semantic model?
- Comparison with Xtext

Semantic Model in Langium



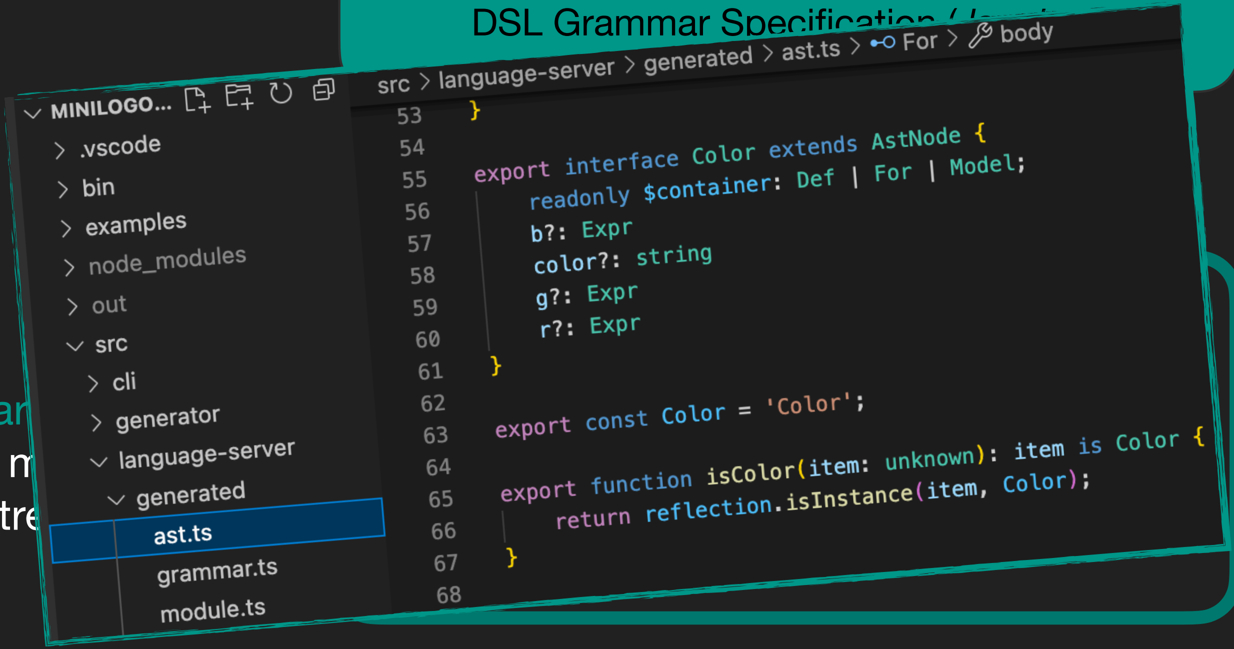
- Parser
  in memory:
  a tree of callbacks

- Semantic Model
  ast.ts file

Semantic Model in Langium

- Par
  in m
  a tre

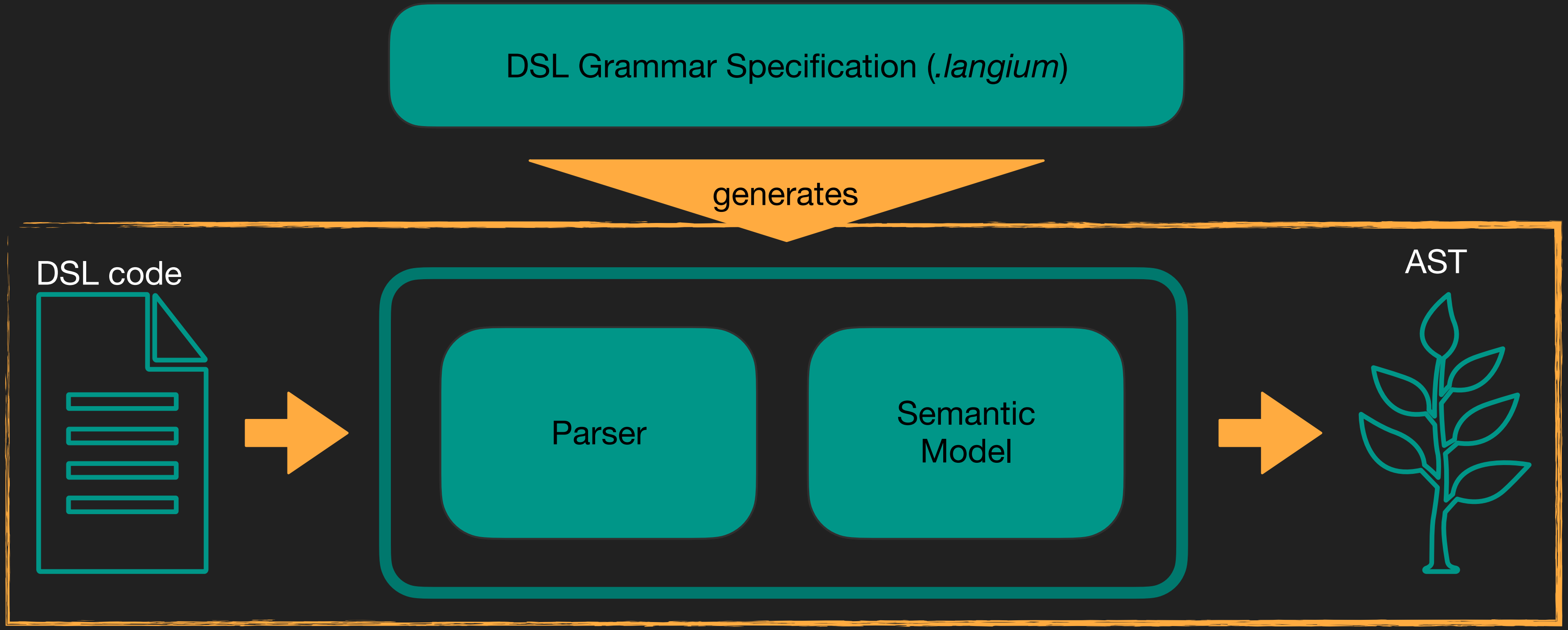
DSL Grammar Specification (.l


src > language-server > generated > ast.ts > For > body

```
}

export interface Color extends AstNode {
    readonly $container: Def | For | Model;
    b?: Expr
    color?: string
    g?: Expr
    r?: Expr
}

export const Color = 'Color';

export function isColor(item: unknown): item is Color {
    return reflection.isInstance(item, Color);
}

```

- Semantic Model
  ast.ts file

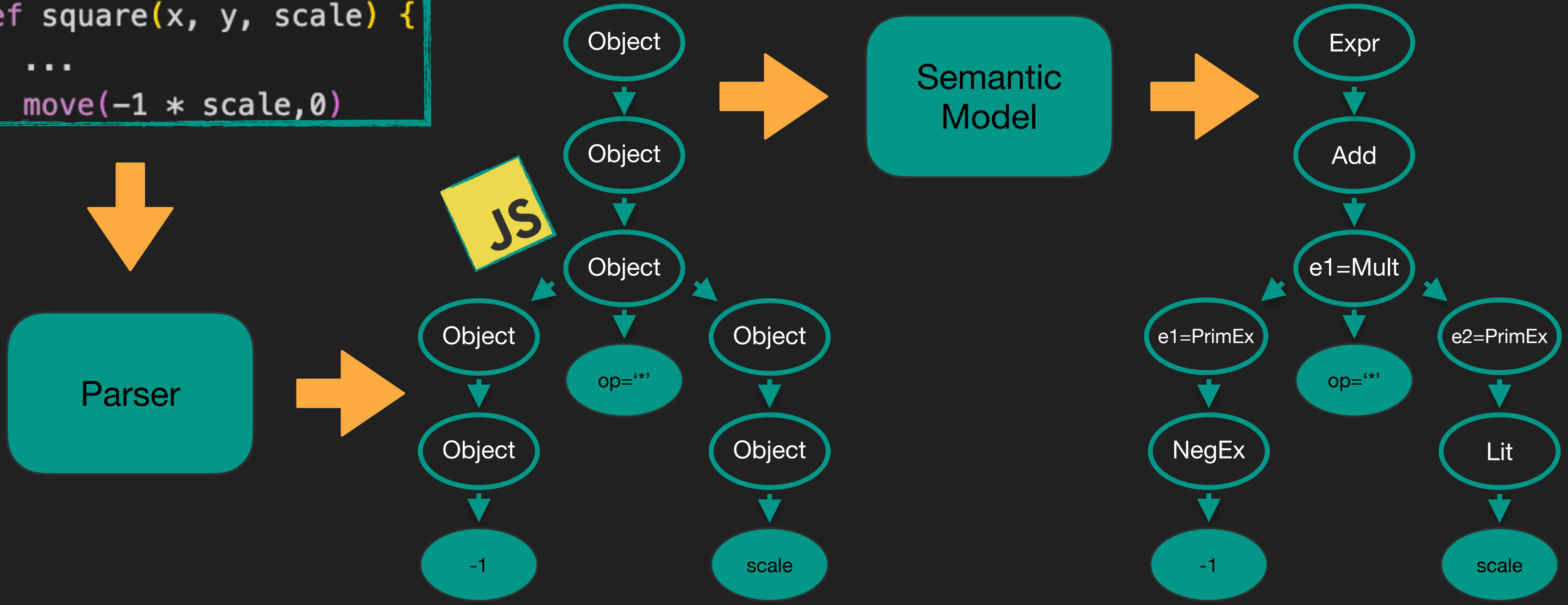
Semantic Model in Langium



Semantic Model in Langium

miniLogo source

```
def square(x, y, scale) {
   ...
   move(-1 * scale,0)
```



Content

- What is a semantic model?
- What to use a semantic model for?
- How does a semantic model look like in Langium?
- How Langium shapes a semantic model?
- **Demo**: how can I use a semantic model?
- Comparison with Xtext

Shaping Semantic Model

Inferred Types

generated from parser rules

- parser rule
- assignment
- cross-reference
- `infer` keyword
- action

Get the semantic model free — nice for brief prototyping

Declared Types

special types syntax in grammar

- interface
- type union
- `return` keyword

Fix the semantic model — nice for mature projects

Shaping Semantic Model

Inferred Types
generated from parser rules

- parser rule
- assignment
- cross-reference
- `infer` keyword
- action

Get the semantic model free — nice for brief prototyping

Declared Types
special types syntax in grammar

- interface
- type union
- word

Type inference and declaration can be used together

Fix the semantic model — nice for mature projects

Inferred Types

- parser rule
- assignment
- cross-reference
- `infer` keyword
- action

```
Expr: Add;
```

```
export type Expr = Add;
```

```
PrimExpr: Lit | Ref | Group | NegExpr;
```

```
export type PrimExpr = Group | Lit | NegExpr | Ref;
```

Inferred Types

- parser rule
- assignment
- cross-reference
- `infer` keyword
- action

```
Param: name=ID;
```

```
export interface Param extends AstNode {
    name: string
}
```

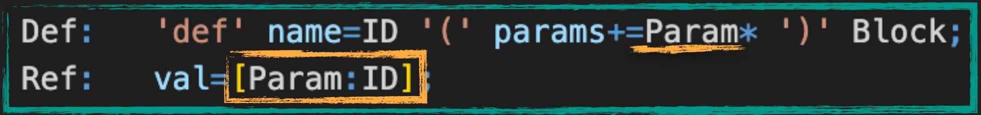
```
Def:    'def' name=ID '(' params+=Param* ')' Block;
```

```
export interface Def extends AstNode {
    body: Array<Stmt>
    name: string
    params: Array<Param>
}
```

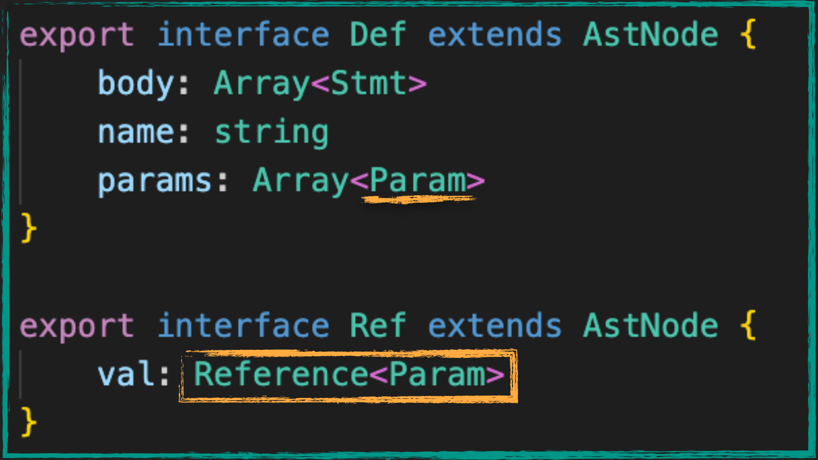
TypeFox

Inferred Types

- parser rule
- assignment
- cross-reference
- `infer` keyword
- action



```
Def:    'def' name=ID '(' params+=Param* ')' Block;
Ref:    val=[Param:ID];
```



```
export interface Def extends AstNode {
    body: Array<Stmt>
    name: string
    params: Array<Param>
}

export interface Ref extends AstNode {
    val: Reference<Param>
}
```

Inferred Types

- parser rule
- assignment
- cross-reference
- infer keyword
- action

```
Expr: Add;
Add  infers Expr:
     e1=Mult       (op=('+'|'-') e2=Mult)*;
Mult infers Expr:
     e1=PrimExpr (op=('*'|'/') e2=PrimExpr)*;
```

TypeFox

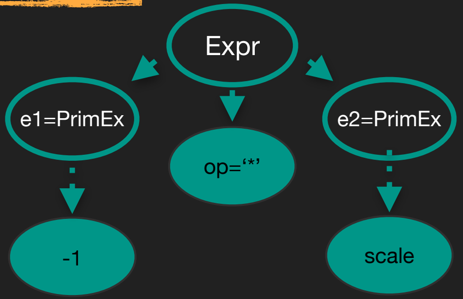
Inferred Types: `infer` keyword

```
Expr: Add;
Add  infers Expr:
    e1=Mult      (op=('+'|'-') e2=Mult)*;
Mult infers Expr:
    e1=PrimExpr (op=('*'|'/') e2=PrimExpr)*;
```

semantic model with `infer`

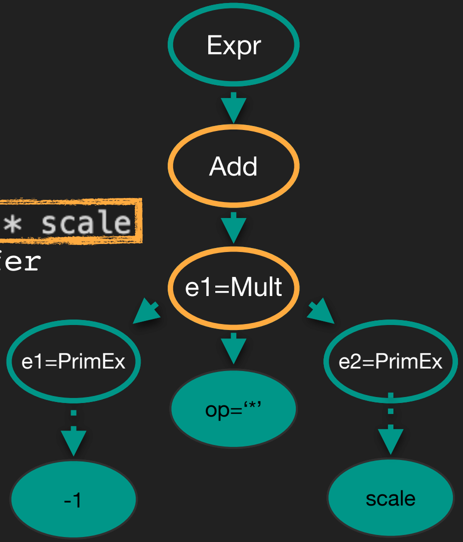
```
export interface Expr extends AstNode {
    e1: Expr | PrimExpr
    e2?: Expr | PrimExpr
    op?: '*' | '+' | '-' | '/'
}
```

AST of `-1 * scale` with `infer`

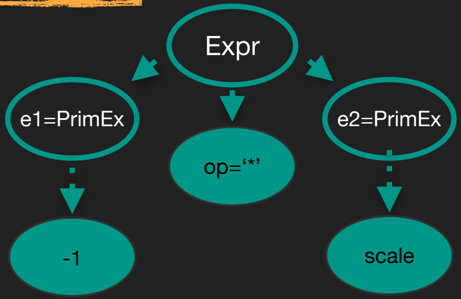


Inferred Types: `infer` keyword

AST of `-1 * scale`
without `infer`



AST of `-1 * scale`
with `infer`



Inferred Types

- parser rule
- assignment
- cross-reference
- `infer` keyword
- action

```
Expr: Add;
Add:   Mult      ({infer BinExpr.e1=current} op=('+'|'-') e2=Mult)*;
Mult: PrimExpr  ({infer BinExpr.e1=current} op=('*'|'/') e2=PrimExpr)*;
```

```
Expr: Add;
Add:  e1=Mult      (op=('+'|'-') e2=Mult)*;
Mult: e2=PrimExpr (op=('*'|'/') e2=PrimExpr)*;
```

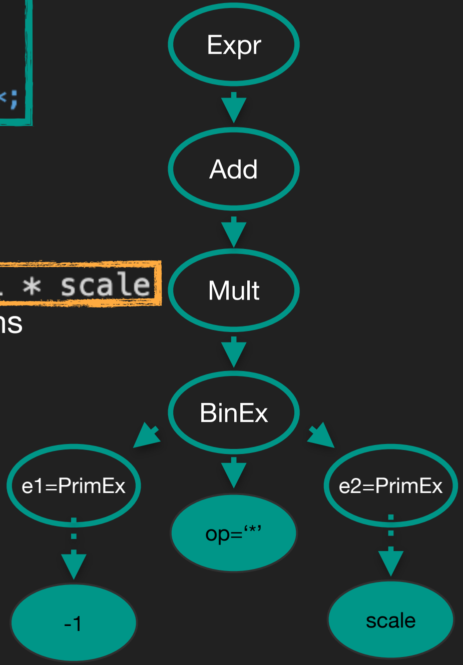
Inferred Types: action

```
Expr: Add;
Add:  Mult     ({infer BinExpr.e1=current} op=('+'|'-') e2=Mult)*;
Mult: PrimExpr ({infer BinExpr.e1=current} op=('*'|'/') e2=PrimExpr)*;
```

semantic model with actions

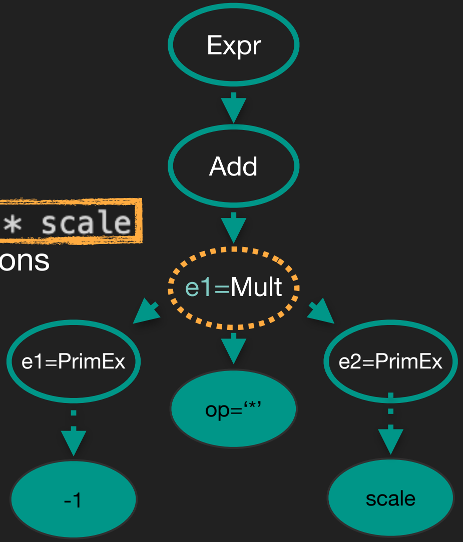
```
export type Expr = Add;
export type Add = BinExpr | Mult;
export type Mult = BinExpr | PrimExpr;

export interface BinExpr extends AstNode {
    e1: Mult | PrimExpr
    e2: Mult | PrimExpr
    op: '*' | '+' | '-' | '/'
}
```

AST of `-1 * scale` with actions

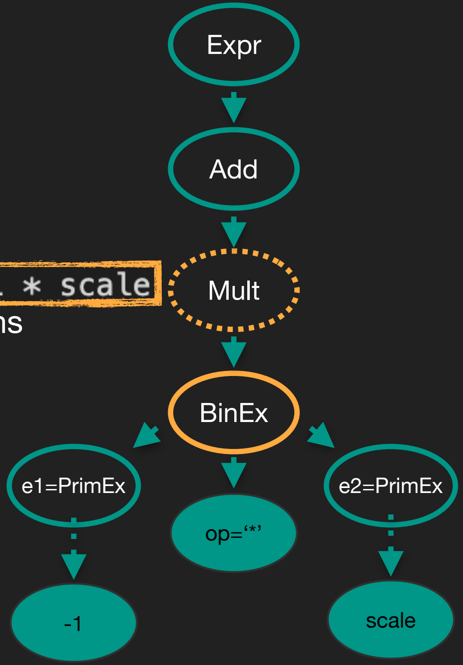


Inferred Types: action

AST of `-1 * scale`
without actions



AST of `-1 * scale`
with actions



Inferred Types

- parser rule
- assignment
- cross-reference
- `infer` keyword
- action

```
Expr: Add;
Add  infers Expr:
    Mult     ({infer BinExpr.e1=current} op=('+'|'-') e2=Mult)*;
Mult infers Expr:
    PrimExpr ({infer BinExpr.e1=current} op=('*'|'/') e2=PrimExpr)*;
```

TypeFox

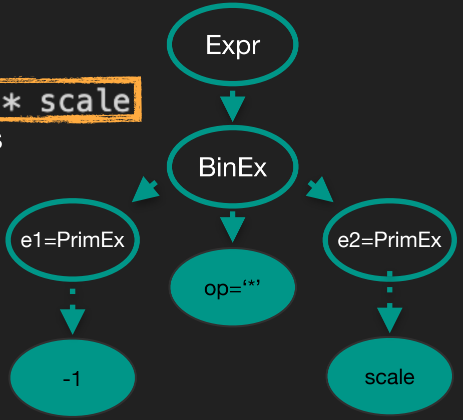
Inferred Types: action + `infer` keyword

```
Expr: Add;
Add  infers Expr:
    Mult     ({infer BinExpr.e1=current} op=('+'|'-') e2=Mult)*;
Mult infers Expr:
    PrimExpr ({infer BinExpr.e1=current} op=('*'|'/') e2=PrimExpr)*;
```

semantic model with actions and `infer`

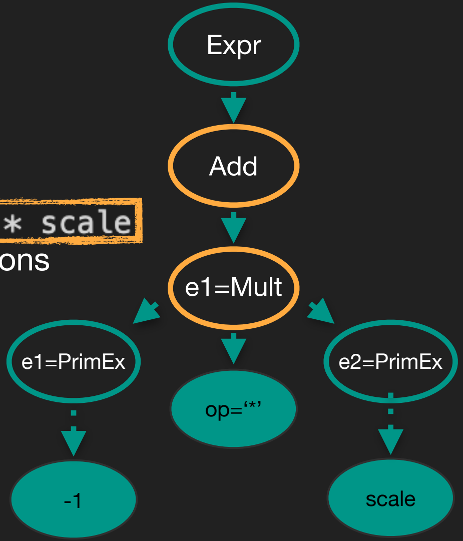
```
export type Expr = BinExpr | PrimExpr;

export interface BinExpr extends AstNode {
    e1: Expr | PrimExpr
    e2: Expr | PrimExpr
    op: '*' | '+' | '-' | '/'
}
```

AST of `-1 * scale` with actions and `infer`

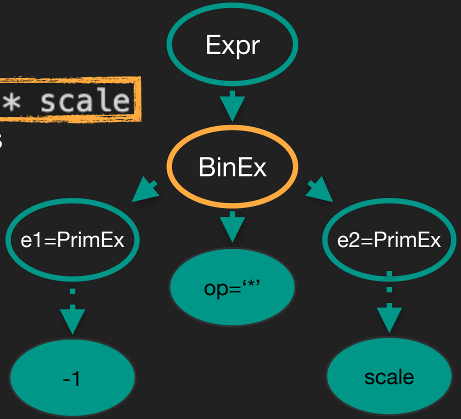


Inferred Types: action + `infer` keyword

AST of `-1 * scale` without actions and `infer`

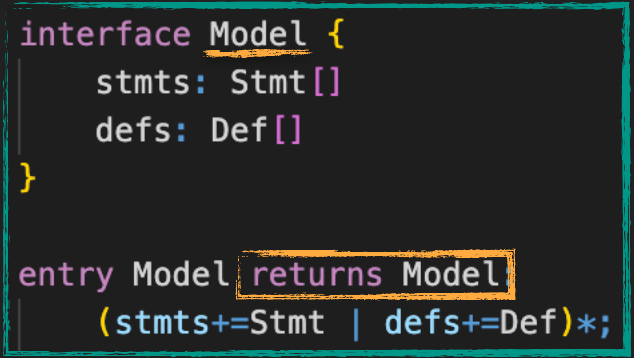


AST of `-1 * scale` with actions and `infer`

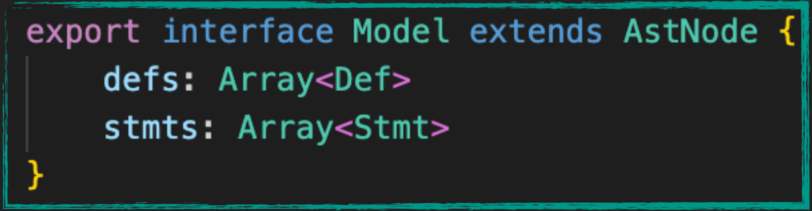


Declared Types

- interface
- type union
- `return` keyword



```
interface Model {
    stmts: Stmt[]
    defs: Def[]
}

entry Model returns Model:
    (stmts+=Stmt | defs+=Def)*;
```



```
export interface Model extends AstNode {
    defs: Array<Def>
    stmts: Array<Stmt>
}
```

Declared Types

- interface
- type union
- `return` keyword

```
type Stmt = Cmd | Macro

Stmt returns Stmt:
    Cmd | Macro;
```

```
export type Stmt = Cmd | Macro;
```

Content

- What is a semantic model?
- What to use a semantic model for?
- How does a semantic model look like in Langium?
- How Langium shapes a semantic model?
- **Demo**: how can I use a semantic model?
- Comparison with Xtext

TypeFox

Demo: *MiniLogo* Semantic Model

https://web.engr.oregonstate.edu/~walkiner/teaching/cs381-wi21/minilogo.html

https://github.com/montymxb/minilogo-langium-example

Inferred semantic model

Break *Validator:* change a parser rule

Declared semantic model

Preserve Validator: get validation errors while change a parser rule

TypeFox

Content

- What is a semantic model?
- What to use a semantic model for?
- How does a semantic model look like in Langium?
- How Langium shapes a semantic model?
- **Demo**: how can I use a semantic model?
- Comparison with Xtext

Langium vs Xtext

langium	Xtext
Declared Types	ECore + Dummy rules
For an AST navigation Fixes a semantic model	For an AST navigation For EMF
No types at runtime	Types at runtime

Langium vs Xtext

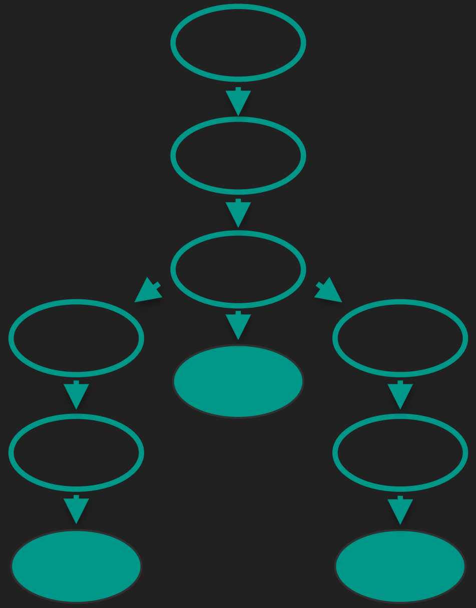
langium	Xtext
Declared Types	ECore + Dummy rules
For an AST navigation Fixes a semantic model	For an AST navigation For EMF
Testing is simpler	Types at runtime

Keynotes

Langium is a lang. engineering framework providing high out-of-box functionality

Langium has powerful tools to shape semantic models

- It can be inferred automatically from the grammar or
- fine-grained by the DSL creator



Langium Going Forward

- langium.org
  - https://langium.org/docs/ast-types/
- Currently 0.4.0 (soon 0.5.0)
- Dev Meetings every Wed. @ 16:00