

Language Design for Everyone

Matteo Cimini
University of Massachusetts Lowell

September, 29th 2023
Lowell, MA, USA

Programming Languages (PLs)

As the theory and practice of PL advances:

New PLs:

TypeScript (Microsoft), Hack (Facebook), Go (Google), Dart (Google), Swift (Apple), Rust (Mozilla), Julia, Elixir, and so on.

New Domain-specific Languages (DSLs):

**Halide. MIT (with help from Stanford, Google, Adobe.) for GPU
Turi. (formerly GraphLab) CMU/UoW acquired by Apple, for ML
GraphIt for Graph analytics.
and so on**

This Talk: Some of My Work on

Empowering people to quickly deploy their own programming languages and domain-specific languages

Flexibility:

- Lang-n-play
- Lang-n-change
- Lang-n-send

Safety:

- Lang-n-check
- Lang-SQL (or Lang-n-query)
- Lang-n-prove

Lang-n-play (Languages as first-class)

C-like array access `a[10]` **access anything, unsafe**
Java-like array access `a[10]` **checks length, safe**

```
addCArrays(Language lan):  
  return lan + { evaluator of C-like _[_] }  
addJavaArrays(Language lan):  
  return lan + { evaluator of Java-like _[_] }  
  
mylan := {syntax + evaluator }  
if trustworthy(code):  
  then addCArrays(mylan) exec code  
  else addJavaArrays(mylan) exec code
```

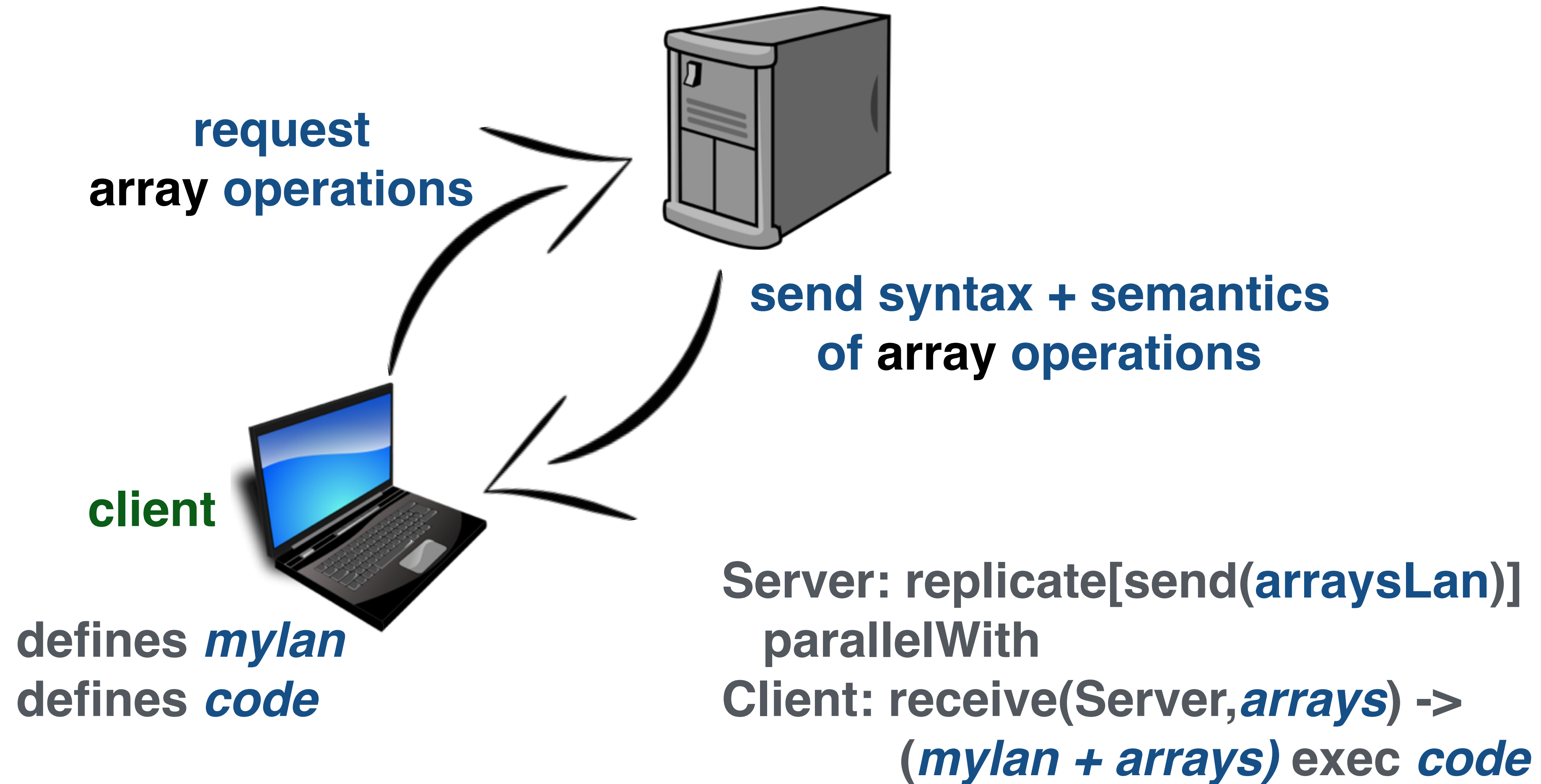
Matteo Cimini. *A Calculus for Multi-language Operational Semantics*. VSTTE 2021.

Matteo Cimini. *On the Effectiveness of Higher-Order Logic Programming in Language-Oriented Programming*. FLOPS 2020.

Matteo Cimini. *Languages as First-Class Citizens (Vision Paper)*. SLE 2018.

Lang-n-send (Processes that send languages)


Server of language definitions



Lang-n-change

(Compiling languages to languages)

```
f(float x) { truncate(x) }  
int x := 3;  
f(x);
```

 **without subtyping: reject**
with subtyping: accept

```
type_check(part_of_prg):  
  if part_of_prg is function type var body arg  
  if type == type_check(arg) then accept  
  else reject
```

must become

```
type_check_with_subtype(part_of_prg):  
  if part_of_prg is function type var body arg  
  type2 := type_check_with_subtype(arg)  
  if subtype(type2, type) then accept  
  else reject
```


Lang-n-change (Compiling languages to languages)

Lang-n-change can express:

```
retrieve type_checker,  
compile 'type1 == type2' into:  
  'newVar1 := type1;  
   newVar2 := type2;  
   if subtype(newVar1, newVar2) then accept  
   else reject'
```

Matteo Cimini, Benjamin Mourad.

Language Transformations in the Classroom. EXPRESS/SOS 2021.

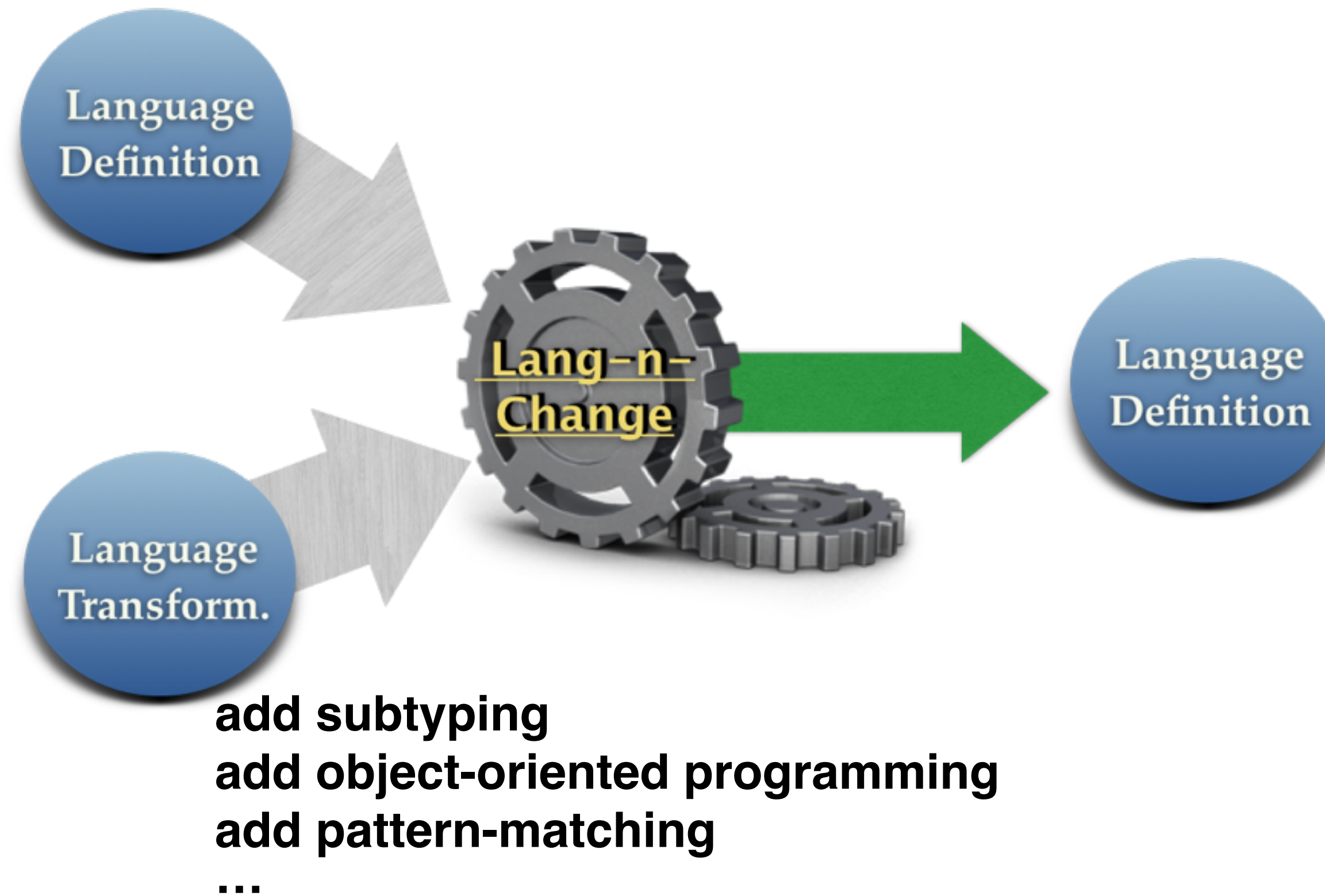
Benjamin Mourad, Matteo Cimini.

A Declarative Gradualizer with Language Transformations. IFL 2020.

Benjamin Mourad, Matteo Cimini. *System Description: Lang-n-Change - A Tool for Transforming Languages.* FLOPS 2020.

Benjamin Mourad, M. Cimini. *A Calculus for Language Transformations.* SOFSEM 2020.

Lang-n-change (Compiling languages to languages)



Matteo Cimini, Benjamin Mourad.

Language Transformations in the Classroom. EXPRESS/SOS 2021.

Benjamin Mourad, Matteo Cimini.

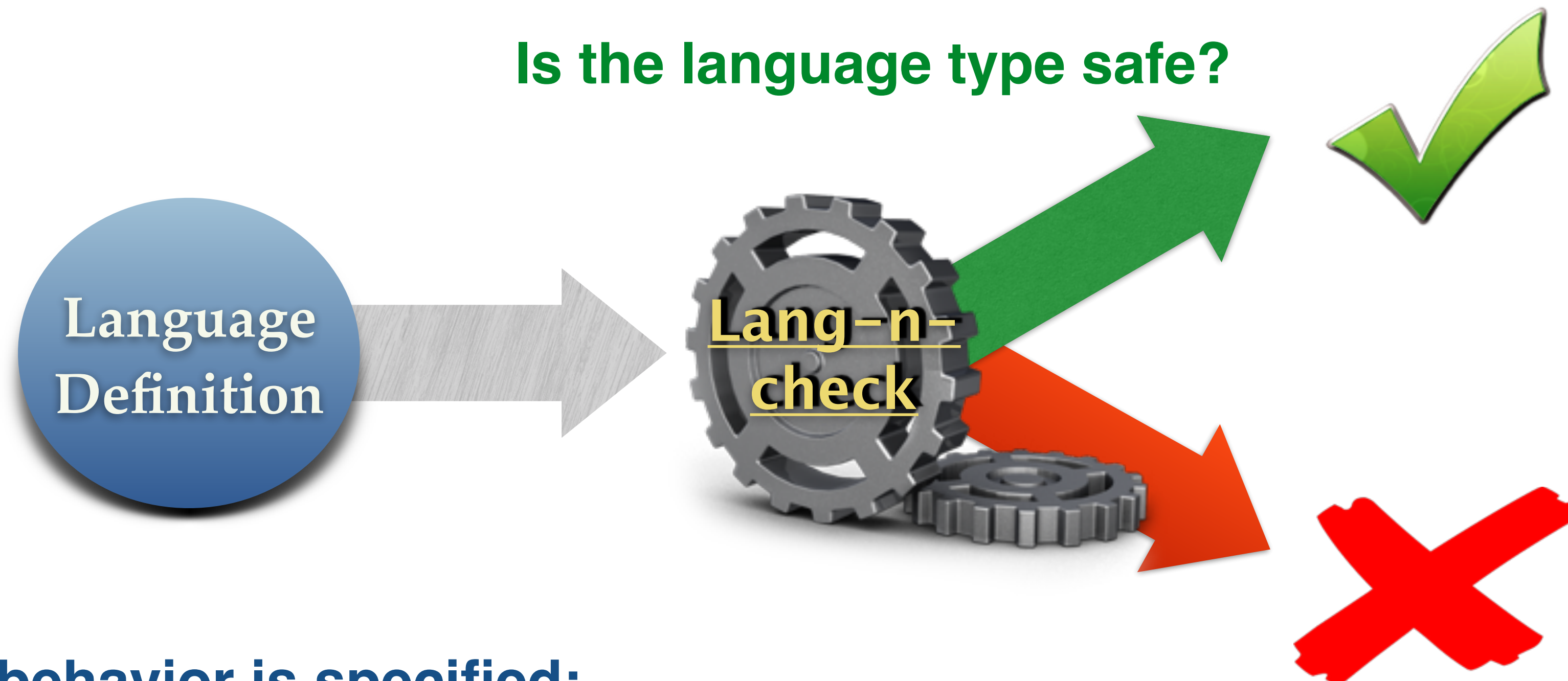
A Declarative Gradualizer with Language Transformations. IFL 2020.

Benjamin Mourad, Matteo Cimini. *System Description: Lang-n-Change - A Tool for Transforming Languages.* FLOPS 2020.

Benjamin Mourad, M. Cimini. *A Calculus for Language Transformations.* SOFSEM 2020.

Safety: Lang-n-check

(A type checker for language definitions)



all behavior is specified:

reject if `eval(n div 0)` is not specified

all behavior is consistent:

reject `int_to_string(4/2) ==> int_to_string(4) / int_to_string(2)`

Matteo Cimini, Dale Miller, Jeremy G. Siek.

Extrinsically Typed Operational Semantics for Functional Languages. SLE 2022.

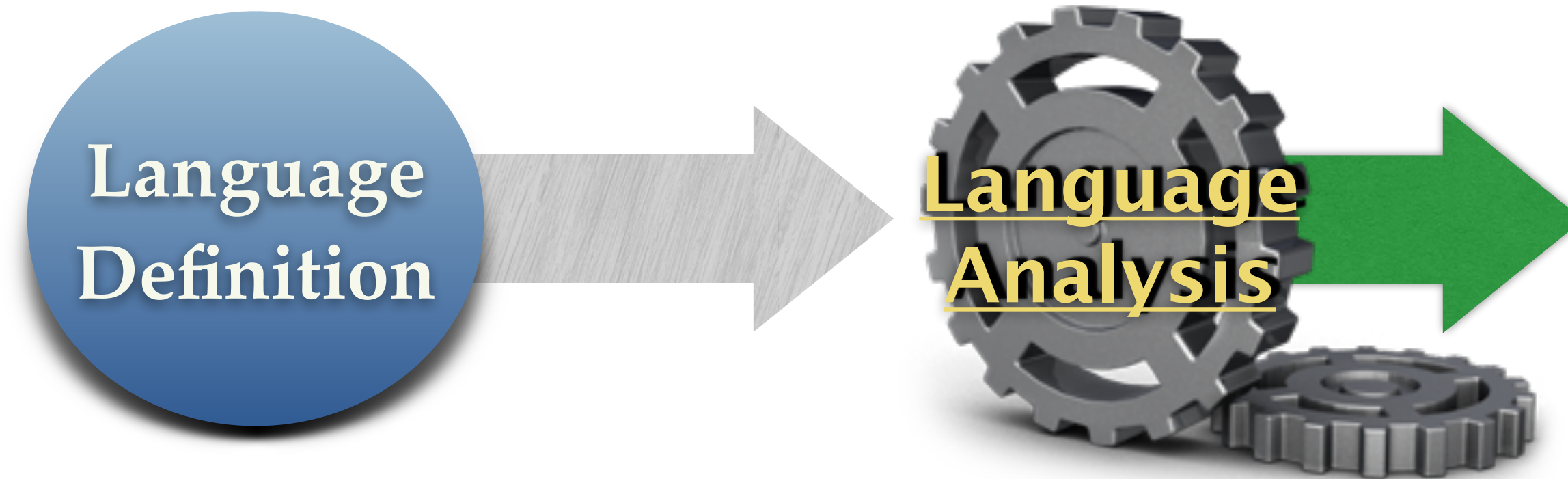
Matteo Cimini. *Early Experience in Teaching the Basics of Functional Language Design with a Language Type Checker. TFP 2019.*

Language Testing

small questions



what constructors bind names?
what polymorphism?
what are the canonical forms?
...



canonical forms

value	type
<i>true</i>	<i>bool</i>
<i>false</i>	<i>bool</i>
<i>function</i>	<i>arrow</i>
$[a_1, \dots, a_n]$	<i>list</i>

helpful for debugging

Lang-SQL

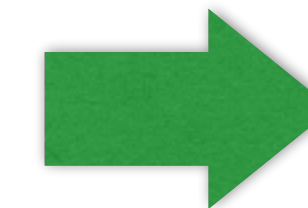
(A query language for querying languages)

“Interrogating languages should be akin to interrogating databases”

- **languages-as-databases.** *Stores languages as database tables.*
- **Lang-SQL,** *tailored for interrogating languages*

canonical forms

```
SELECT GET-OPNAME(term) AS value,  
       GET-OPNAME(EXPRESSION(args)) AS type  
FROM Value, rule  
WHERE predname = type_check AND role = CONCL  
AND GET-OPNAME(term) = GET-OPNAME(OUTPUT-TYPE(args)))
```



value	type
<i>true</i>	<i>bool</i>
<i>false</i>	<i>bool</i>
<i>fun</i>	<i>arrow</i>
<i>[...]</i>	<i>list</i>

Matteo Cimini. *Testing Languages with a Languages-as-Databases Approach.* TAP 2023.

Matteo Cimini. *A Declarative Validator for GSOS Languages.* PLACES 2023.

Matteo Cimini. *A Query Language for Language Analysis.* SEFM 2022.

Lang-n-prove: a proof language for language proofs

1. Theorem *progress* : $\forall e, T. \text{Main} : \vdash e : T \Rightarrow e \text{ is a result } \vee \exists e'. e \longrightarrow e'$.
2. Proof
3. *induction on Main*.
4. *apply inductive hypothesis on the typing of the arg. (e of (head e) and (fst e)).*
5. *case analysis on the progress of that argument.*
 - (a) head e
 6. *apply canonical-form-list.*
 7. *appeal to existence of a rule for nil.*
 8. *appeal to existence of a rule for cons*
 9. *proof continues.*
 - (b) fst e
 6. *apply canonical-form- \times . (product)*
 7. *appeal to existence of a rule for $\langle v, v \rangle$.*
 8. *proof continues.*

Lang-n-prove can express:

if e handles type t then

apply canonical-form_t.

[for each v in valuesOf(t): QED “rule exists”.

Matteo Cimini.

Towards the Complexity Analysis of Programming Language Proof Methods. ICTAC 2023.

Matteo Cimini. *Lang-n-Prove: A DSL for Language Proofs.* SLE 2022.

Thankful to NSF for grant *Language-agnostic proofs*. PI: Matteo Cimini.

Conclusion: Some of My Work on

Empowering people to quickly deploy their own programming languages and domain-specific languages

Flexibility:

- Lang-n-play
- Lang-n-change
- Lang-n-send

Safety:

- Lang-n-check
- Lang-SQL (or Lang-n-query)
- Lang-n-prove

Thank you!