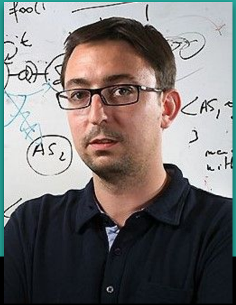


The (software) Language Extension Problem - LEP

Leduc, M., Degueule, T., Van Wyk, E., and Combemale B. The Software Language Extension Problem.
Softw Syst Model 19, 263–267 (2020). https://doi.org/10.1007/s10270-019-00772-7

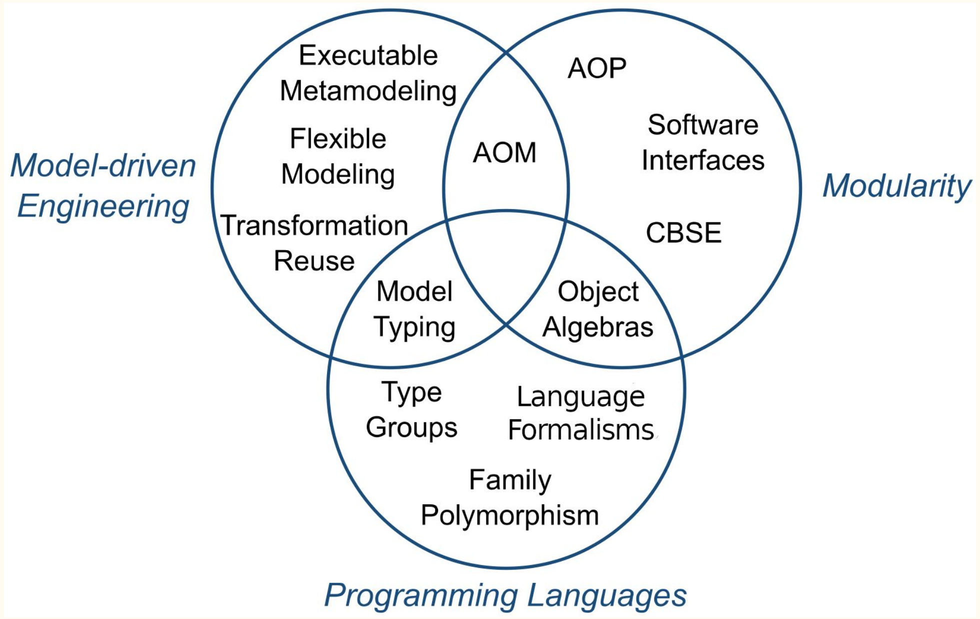
Preprint: https://hal.inria.fr/hal-02399166

Language Extension Problem?

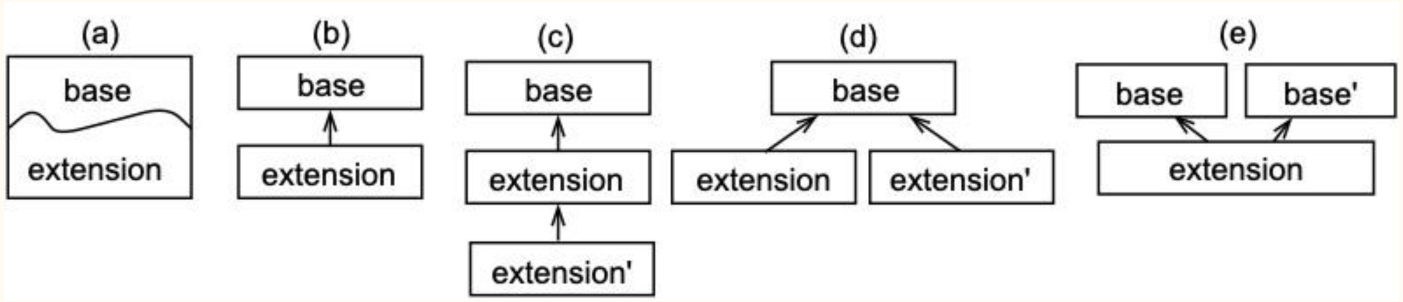
This problem informally refers to the capability of extending the syntax and semantics of an existing language while reusing its specification (e.g., grammars, semantic inference rules) and implementation (e.g., parsers, interpreters).

Approaches for
software language extension?

Existing Approaches for Language Extension



Existing Approaches for Language Extension



All applicable at the specification and implementation levels. (a) mixes up the extension into the base language, while (b)-(e) keep them separated and use explicit operators (e.g., references, static/dynamic introduction) or glue code.

Taxonomy inspired from Erdweg, S., Giarrusso, P.G., Rendel, T.: *Language composition untangled*. In: LDTA 2012.

Specific ins and outs of software language extension?

Ins and Outs???

Why !

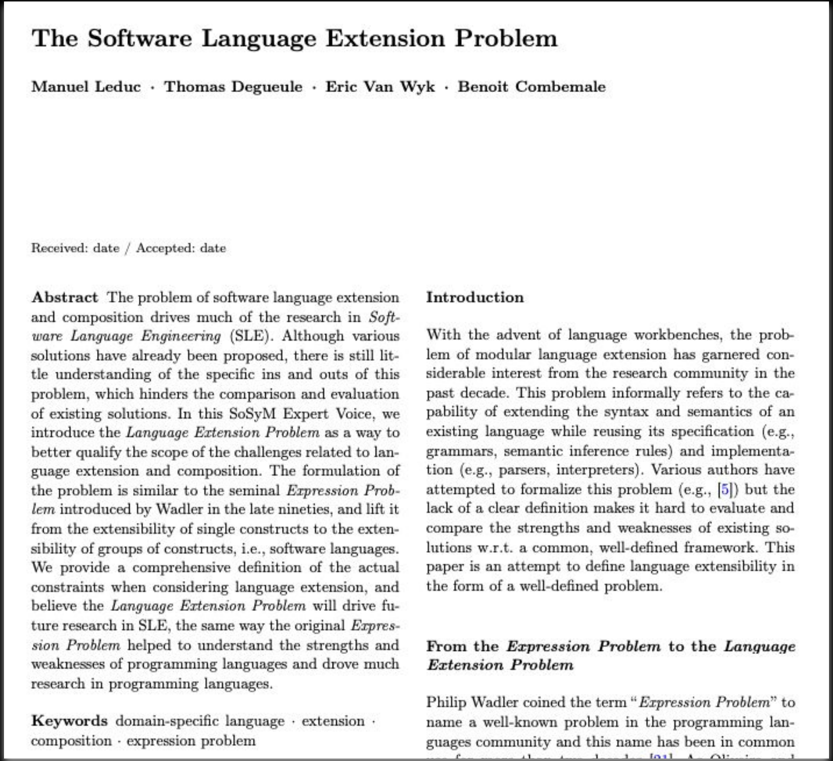
Ins and Outs???

Why !

Hard to evaluate and compare the strengths and weaknesses of existing solutions.

Ins and Outs???

LEP is an attempt to define language extensibility in the form of a well-defined problem.



The Software Language Extension Problem

Manuel Leduc · Thomas Degueule · Eric Van Wyk · Benoit Combemale

Received: date / Accepted: date

Abstract The problem of software language extension and composition drives much of the research in *Software Language Engineering* (SLE). Although various solutions have already been proposed, there is still little understanding of the specific ins and outs of this problem, which hinders the comparison and evaluation of existing solutions. In this SoSyM Expert Voice, we introduce the *Language Extension Problem* as a way to better qualify the scope of the challenges related to language extension and composition. The formulation of the problem is similar to the seminal *Expression Problem* introduced by Wadler in the late nineties, and lift it from the extensibility of single constructs to the extensibility of groups of constructs, i.e., software languages. We provide a comprehensive definition of the actual constraints when considering language extension, and believe the *Language Extension Problem* will drive future research in SLE, the same way the original *Expression Problem* helped to understand the strengths and weaknesses of programming languages and drove much research in programming languages.

Keywords domain-specific language · extension · composition · expression problem

Introduction

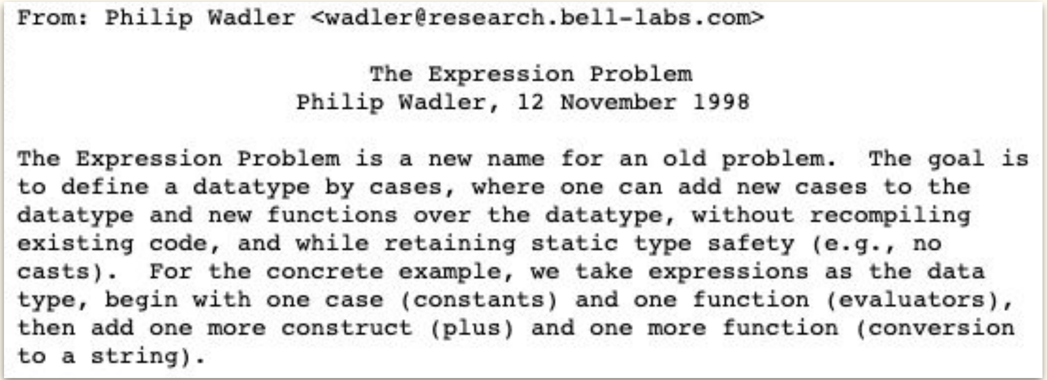
With the advent of language workbenches, the problem of modular language extension has garnered considerable interest from the research community in the past decade. This problem informally refers to the capability of extending the syntax and semantics of an existing language while reusing its specification (e.g., grammars, semantic inference rules) and implementation (e.g., parsers, interpreters). Various authors have attempted to formalize this problem (e.g., [5]) but the lack of a clear definition makes it hard to evaluate and compare the strengths and weaknesses of existing solutions w.r.t. a common, well-defined framework. This paper is an attempt to define language extensibility in the form of a well-defined problem.

From the *Expression Problem* to the *Language Extension Problem*

Philip Wadler coined the term "*Expression Problem*" to name a well-known problem in the programming languages community and this name has been in common

LEP is an attempt to define **language extensibility** in the form of a **well-defined problem.**

The Expression Problem



```
From: Philip Wadler <wadler@research.bell-labs.com>

                    The Expression Problem
               Philip Wadler, 12 November 1998

The Expression Problem is a new name for an old problem.  The goal is
to define a datatype by cases, where one can add new cases to the
datatype and new functions over the datatype, without recompiling
existing code, and while retaining static type safety (e.g., no
casts).  For the concrete example, we take expressions as the data
type, begin with one case (constants) and one function (evaluators),
then add one more construct (plus) and one more function (conversion
to a string).
```

http://homepages.inf.ed.ac.uk/wadler/papers/expression/expression.txt

An old problem going as far back as John Reynolds in 1975, given this name in 1998.

"The "expression problem" (EP) is now a classical problem in programming languages. It refers to the difficulty of writing data abstractions that can be easily extended with both new operations and new data variants."

-- B. Oliveira and W. Cook. "Extensibility for the masses: practical extensibility with object algebras". In ECOOP'12.

EP Constraints

1. **Extensibility in both dimensions:** It should be possible to add new data variants and adapt existing operations accordingly. Furthermore, it should be possible to introduce new operations.

2. **Strong static type safety**: It should be impossible to apply an operation to a data variant that the operation cannot handle.

3. **No modification or duplication**: Existing code should neither be modified nor duplicated.

4. **Separate compilation**: Compiling datatype extensions or adding new operations should not encompass re-type-checking the original datatype or existing operations.

5. **Independent extensibility***: It should be possible to combine independently developed extensions so that they can be used jointly.

* M. Zenger and M. Odersky. 2005. Independently extensible solutions to the expression problem. In FOOL '12.

While the ***EP is merely a programming problem*** *concerning programmers and focusing on the extensibility of a single datatype,*

the ***LEP is a Software Language Engineering (SLE) problem*** *concerning language engineers and focusing on the extensibility of languages (i.e., group of types).*

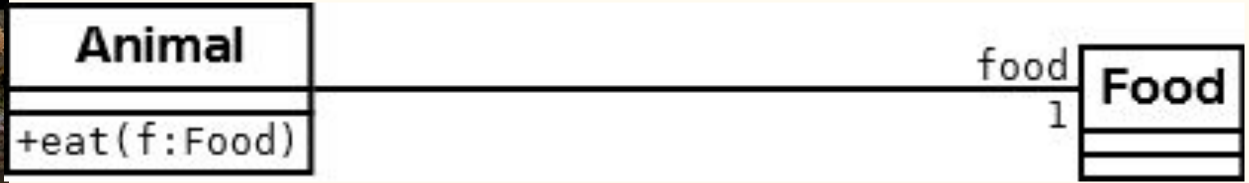
Type group?

Type group?

- Languages are group of mutually recursive (data) types
- Extension mechanisms requiert to ensure a safe and consistent extensibility of both the syntax and the semantics, at the specification and implementation (incl. language tooling).

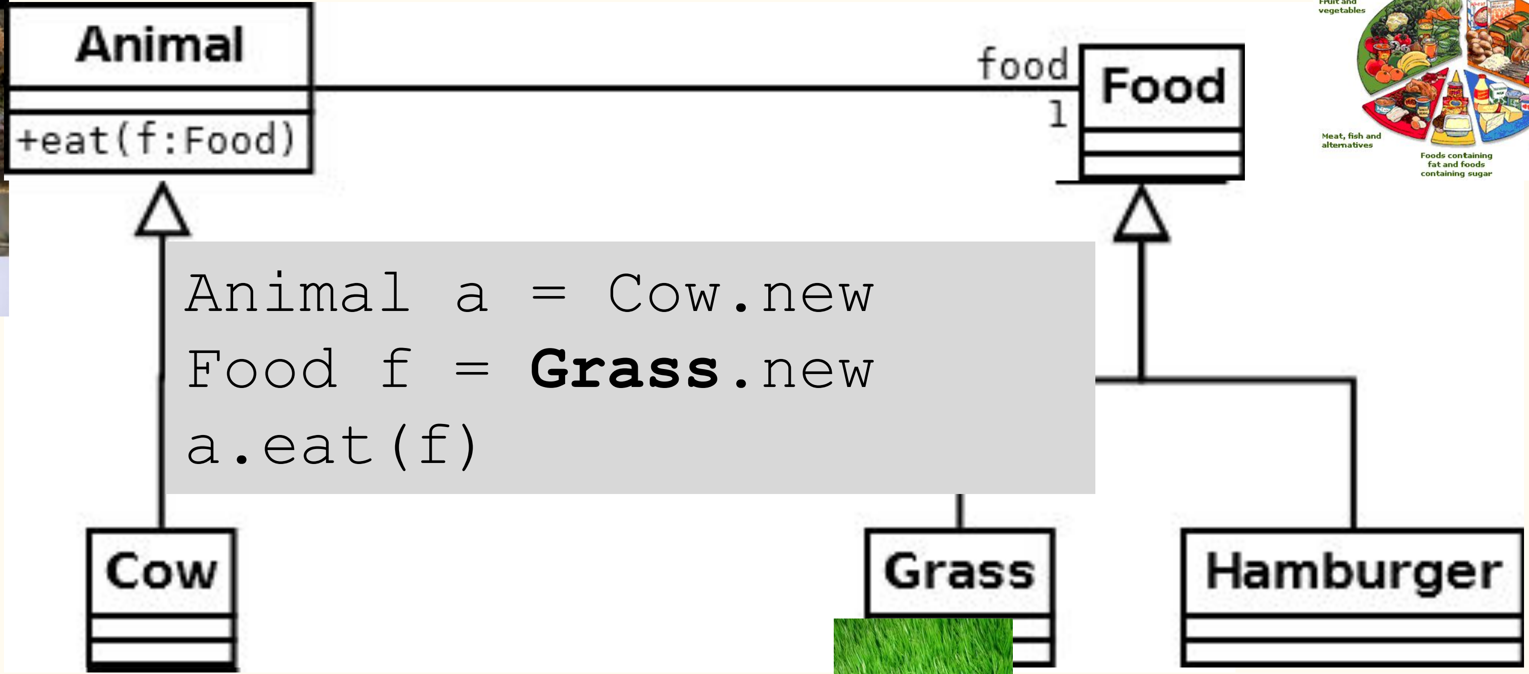
- Concrete illustration from OO? Back to the 90's: Bruce, K.B., Vanderwaart, J.: *Semantics-driven language design: Statically type-safe virtual types in object-oriented languages*. ENTCS 20 (1999) 50–75

Type group?

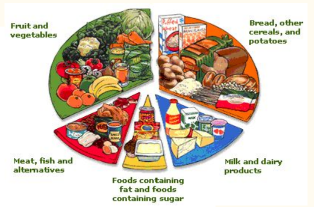

Animal
+eat(f:Food)
food
1
Food

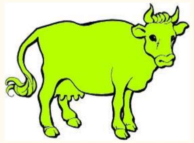
Fruit and vegetables
Bread, other cereals, and potatoes
Meat, fish and alternatives
Milk and dairy products
Foods containing fat and foods containing sugar

Type group?

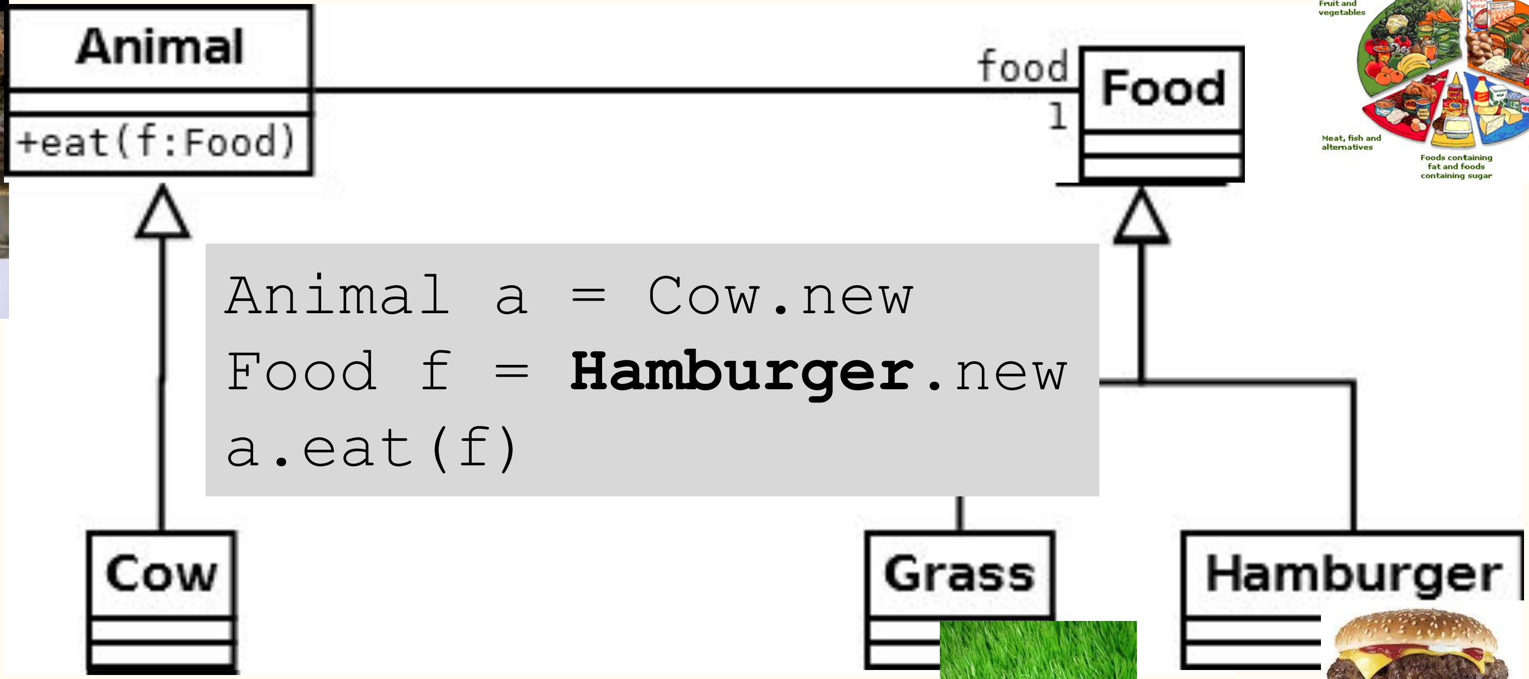


```
Animal a = Cow.new
Food f = Grass.new
a.eat(f)
```

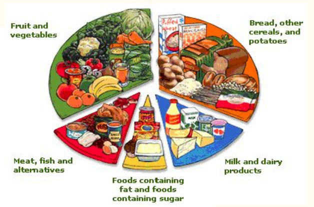

Fruit and vegetables
Bread, other cereals, and potatoes
Meat, fish and alternatives
Milk and dairy products
Foods containing fat and foods containing sugar

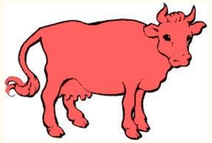


Type group?

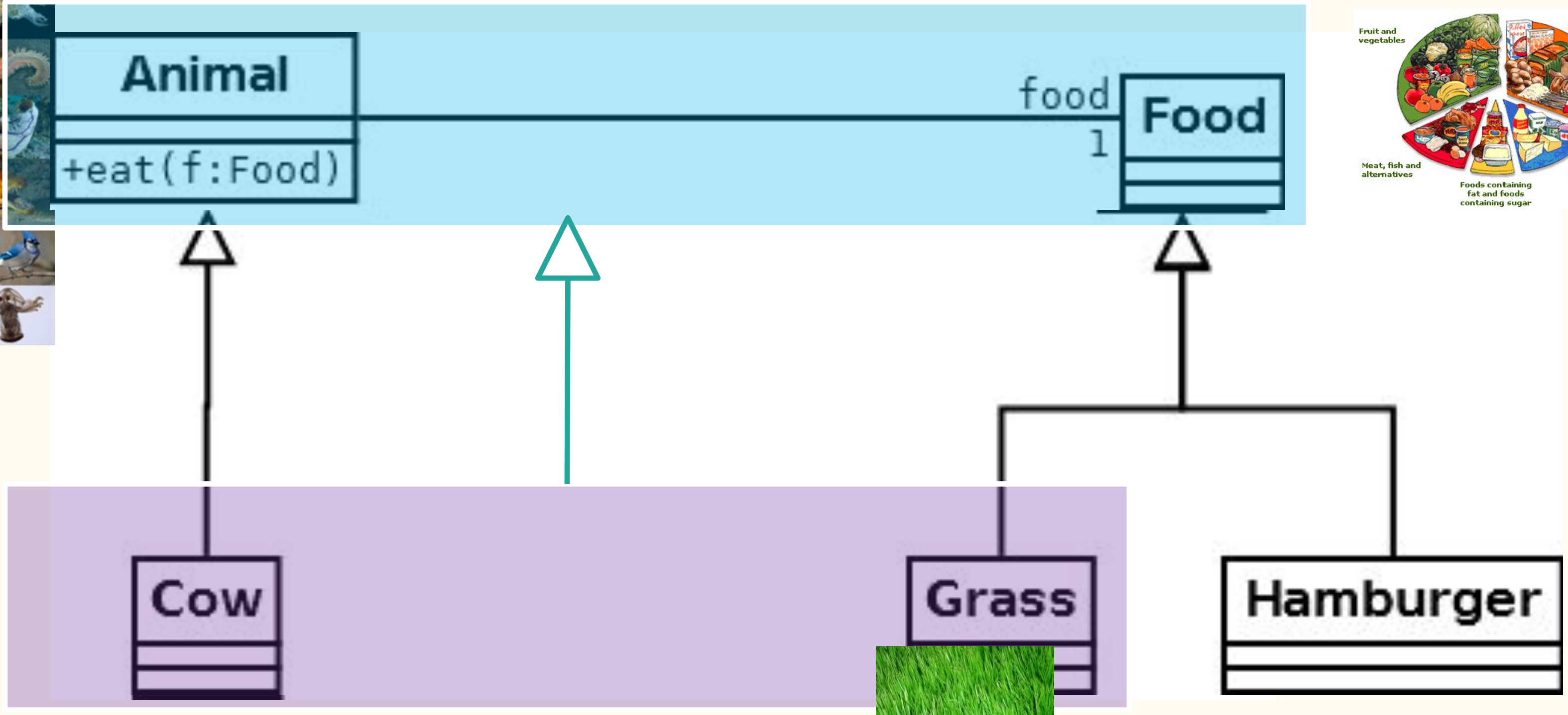


```
Animal a = Cow.new
Food f = **Hamburger**.new
a.eat(f)
```

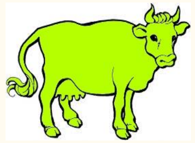

Fruit and vegetables
Bread, other cereals, and potatoes
Meat, fish and alternatives
Milk and dairy products
Foods containing fat and foods containing sugar



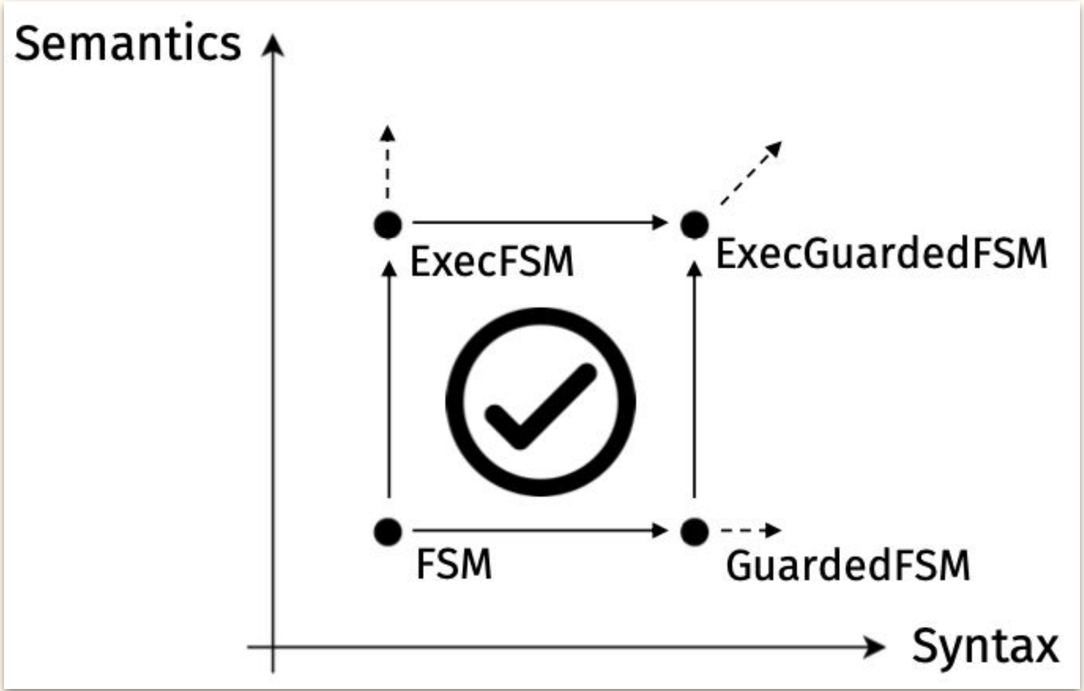
Type group?


Animal
+eat(f:Food)
food
1
Food
Cow
Grass
Hamburger

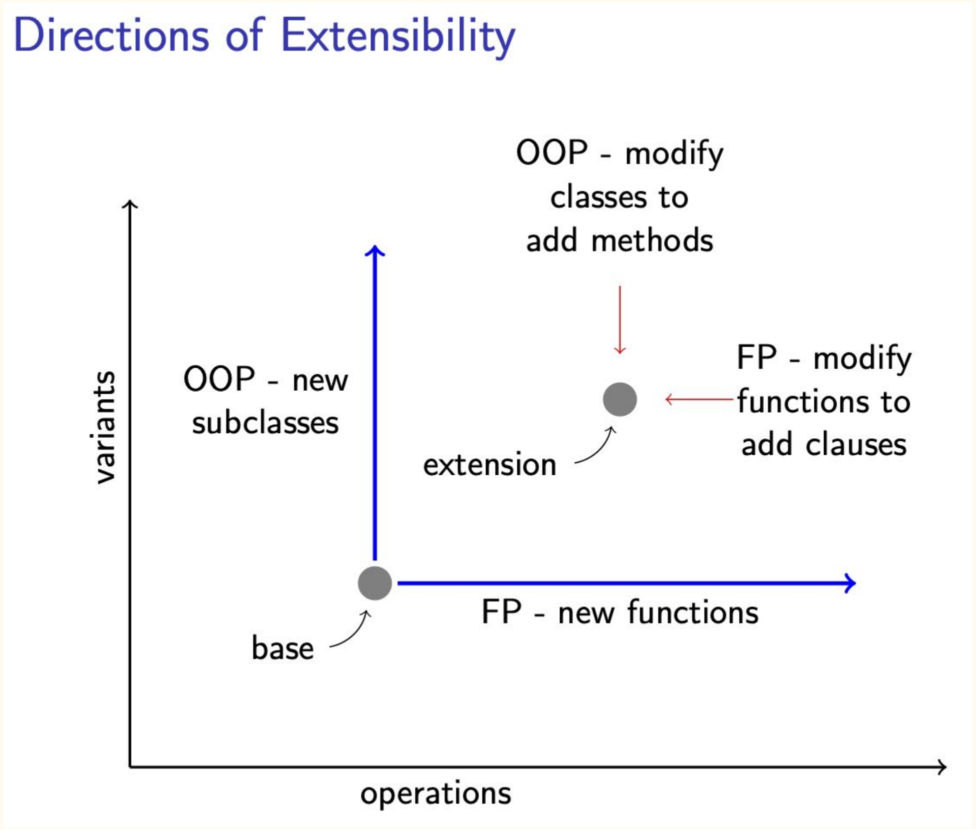
Fruit and vegetables
Bread, other cereals, and potatoes
Meat, fish and alternatives
Foods containing fat and foods containing sugar
Milk and dairy products



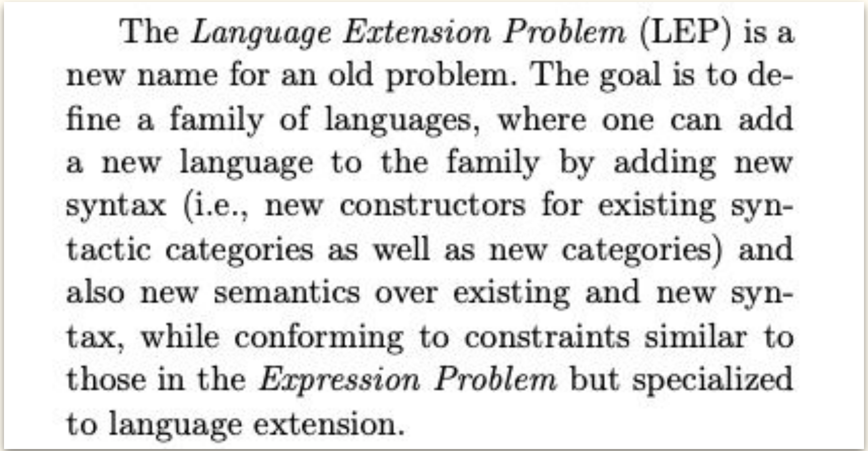
Type group?



Type group?


Directions of Extensibility
OOP - modify classes to add methods
FP - modify functions to add clauses
OOP - new subclasses
extension
base
FP - new functions
variants
operations

The (Software) Language Extension Problem



The *Language Extension Problem* (LEP) is a new name for an old problem. The goal is to define a family of languages, where one can add a new language to the family by adding new syntax (i.e., new constructors for existing syntactic categories as well as new categories) and also new semantics over existing and new syntax, while conforming to constraints similar to those in the *Expression Problem* but specialized to language extension.

- Lifts the vocabulary from datatypes to languages
- Add specific constraints to consider engineering practices, e.g., the distinction between the specification and the implementation

LEP Constraints

1. **Extensibility in both dimensions:** It should be possible to extend the syntax and adapt existing semantics accordingly. Furthermore, it should be possible to introduce new semantics on top of existing syntax.

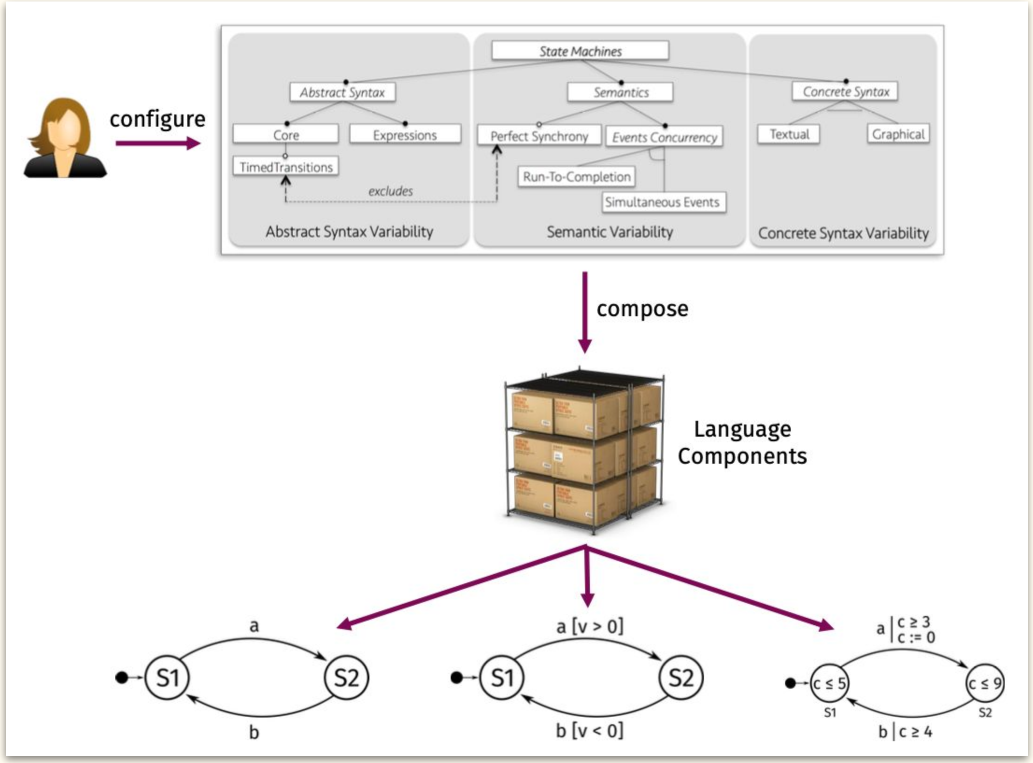
2. **Strong static type safety**: All semantics should be defined for all syntax.

3. **No modification or duplication**: Existing language specifications and implementations should neither be modified nor duplicated.

4. **Separate compilation**: Compiling a new language (e.g., syntactic extension or new semantics) should not encompass re-compiling the original syntax or semantics.

5. **Independent extensibility**: It should be possible to combine and use jointly language extensions (syntax or semantics) independently developed.

Language families: the holy grail



Take away messages

The LEP

- lifts the constraints drawn from the EP to the SLE context
- provides a framework to reason on language extension and its challenges
- helps the comparison of existing and future SLE contributions

Call for Action?

- collaborative refinement of the problem specification
- online review and classification of existing approaches

@manuel_leduc @tdegueul @ericvanwyk @bcombemale

Preprint: https://hal.inria.fr/hal-02399166