Implementing Software Product Lines in Python

Alejandro Russo
russo@chalmers.se

FP Workshop, Jun 2nd, 2010
Chalmers

Python?

- Is it a functional language?
  - Functions are first-class citizens
  - High order functions
  - List comprehension
  - But no static types
- What esle?
  - Imperative, Object-oriented

Software Product Lines

- A set of software systems with well-defined *commonalities* and *variabilities*
  - ***Ultimate goal: reuse of code***
  - ***Holy grail: automatic assembly of products***

SPL and OO Programming

- "... The main idea is to overcome **the limitations** of class-based inheritance with respect to **code reuse** by replacing it with trait composition" [Bettini, Damiani, Schaefer 2009]
  - Trait = set of methods
  - Class = register (state) + trait (interface), where + is substitution

SPL and Python

- In Python, with *a little bit of programming*, it is possible to **precisely** control what is inherited from a class
  - No need for new concepts in order to build programs
  - I will show a set of combinators specifically design to build SPL
  - This is a very work-in-progress

Demo

Implementing SPL

products_id(p_1,..., p_n)

produce(p_1,..., p_n)

is_produced(p)

```
@line_class
class user_class:
    …
    @exclude_func(p)
    def f(...):

    @alias_func(p,str')
    def f(...):

    @rename_func(p,str')
    def f(...):
```

@inherits(c_1,c_2,..,c_n)

```
class user_class:
    …
```

c' = exclude(c,m_1,...,m_n)

c' = rename(c,(m_1, m'_1), ...,(m_n,m'_n))

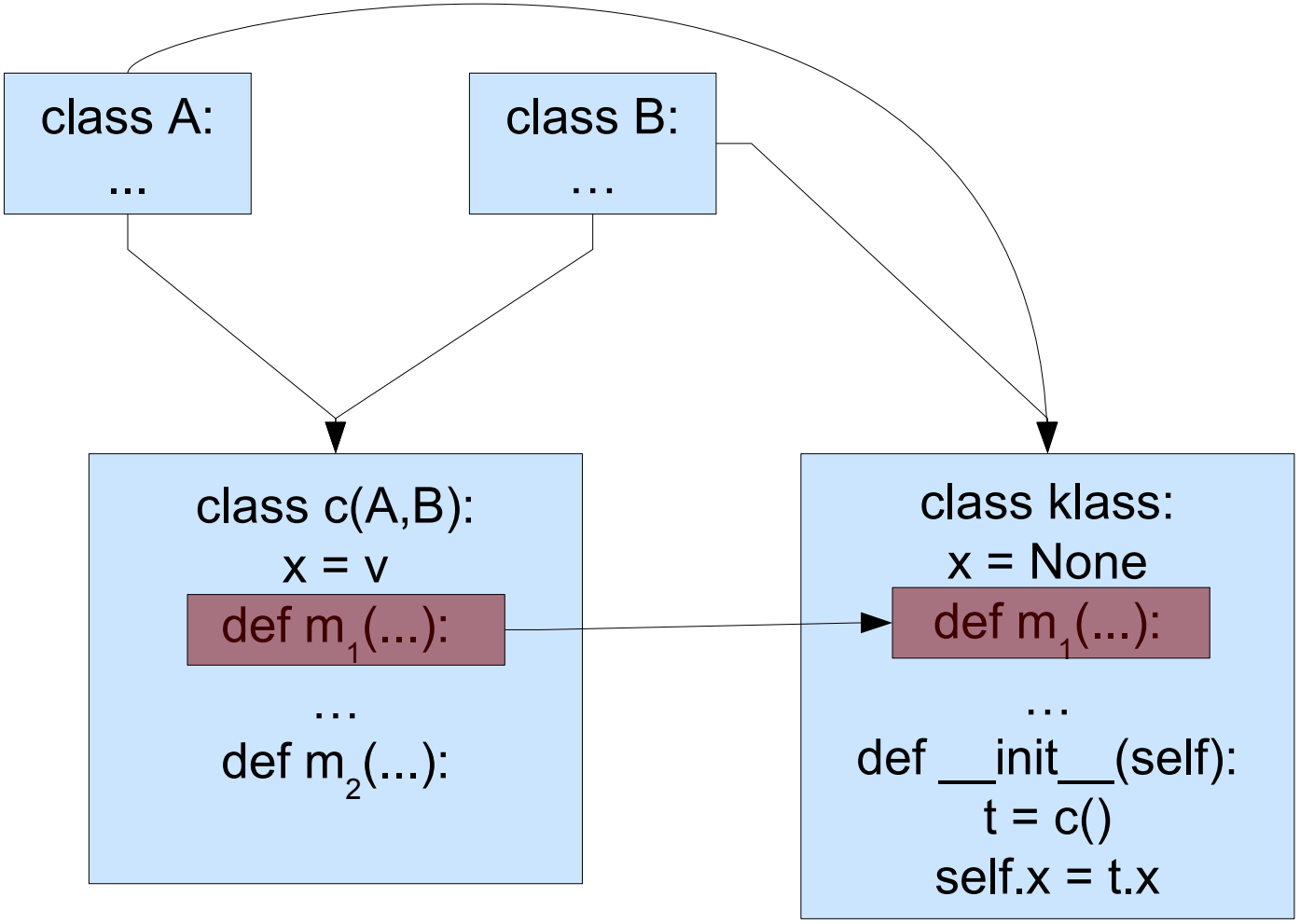
c' = alias(c,(m_1, m'_1), ...,(m_n,m'_n))

c' = line(str)(exclude(c,m_1,...,m_n))

@inherits(exclude(c,m),
rename(exclude(c,m_2),(m_1, m'_1))

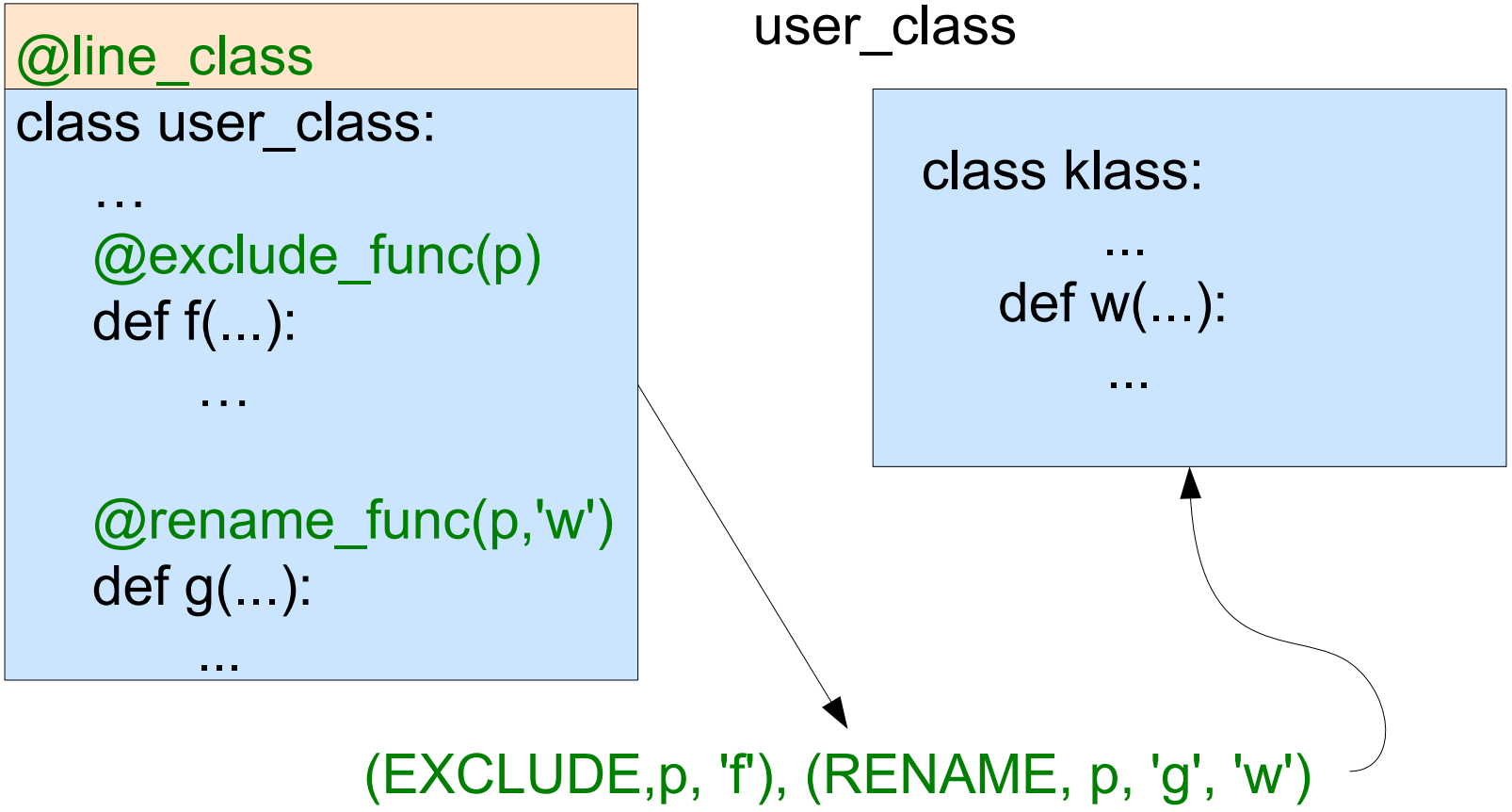
```
class user_class:
    …
```

Excluding methods



exclude(c,m_2)

Excluding methods
(similar to pre-processors)



Final remarks

- It is possible to provide decorators specifically design for building SPL in Python
  - Provided as a library
  - 150 LOC
- **Inheritance is not bad in presence of Python's expressiveness** (other languages?)
- No need to adapt an existing programming language or propose new ones!
- Different from other approaches for SPL
  - We do not provide static guarantees