DSL: Project

Subject and details available online (see Google doc):
https://github.com/FAMILIAR-project/HackOurLanguages-SIF

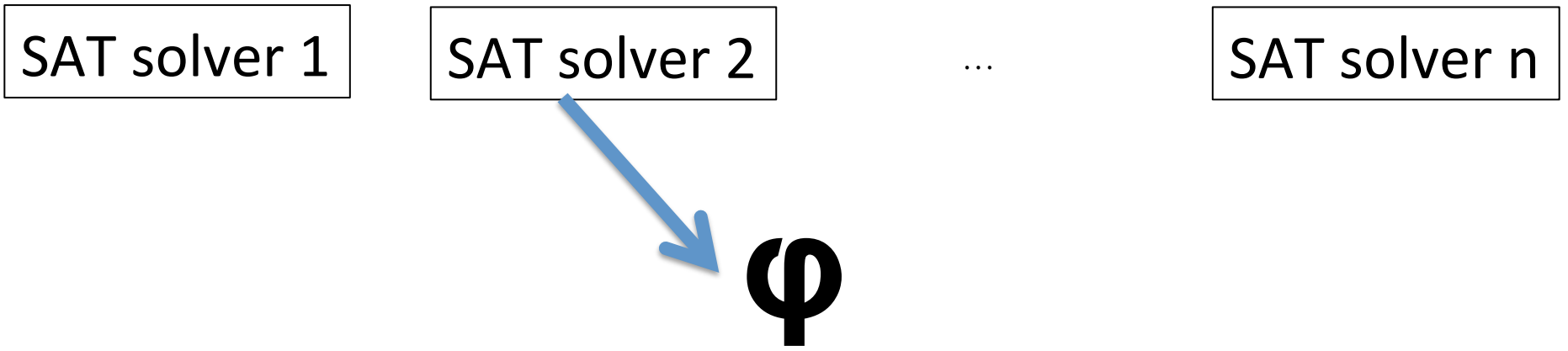
Using, validating, and comparing SAT solvers with a DSL and some compilers

Satisfiability (SAT)

- NP-complete problem
- SAT formula, usually in conjunctive normal form (CNF)
  - eg (x1 v x2) ^ (x3 v x2) ^ (~x3 v x1)
- SAT **solvers** can give two possible answers:
  - Yes and a solution
  - No
- There are many variants of SAT solvers
  - written in C, C++, Python, etc.
  - we will implement a **family** of SAT solvers
- Three challenges:
  - make SAT solvers **usable** out of a simple DSL
  - find functional and performance **bugs** of SAT solvers
  - **comparing** performances of SAT solvers

Usability (requirements of your DSL):
specification of the formula
name of the solver you want to use
answer in a unified format

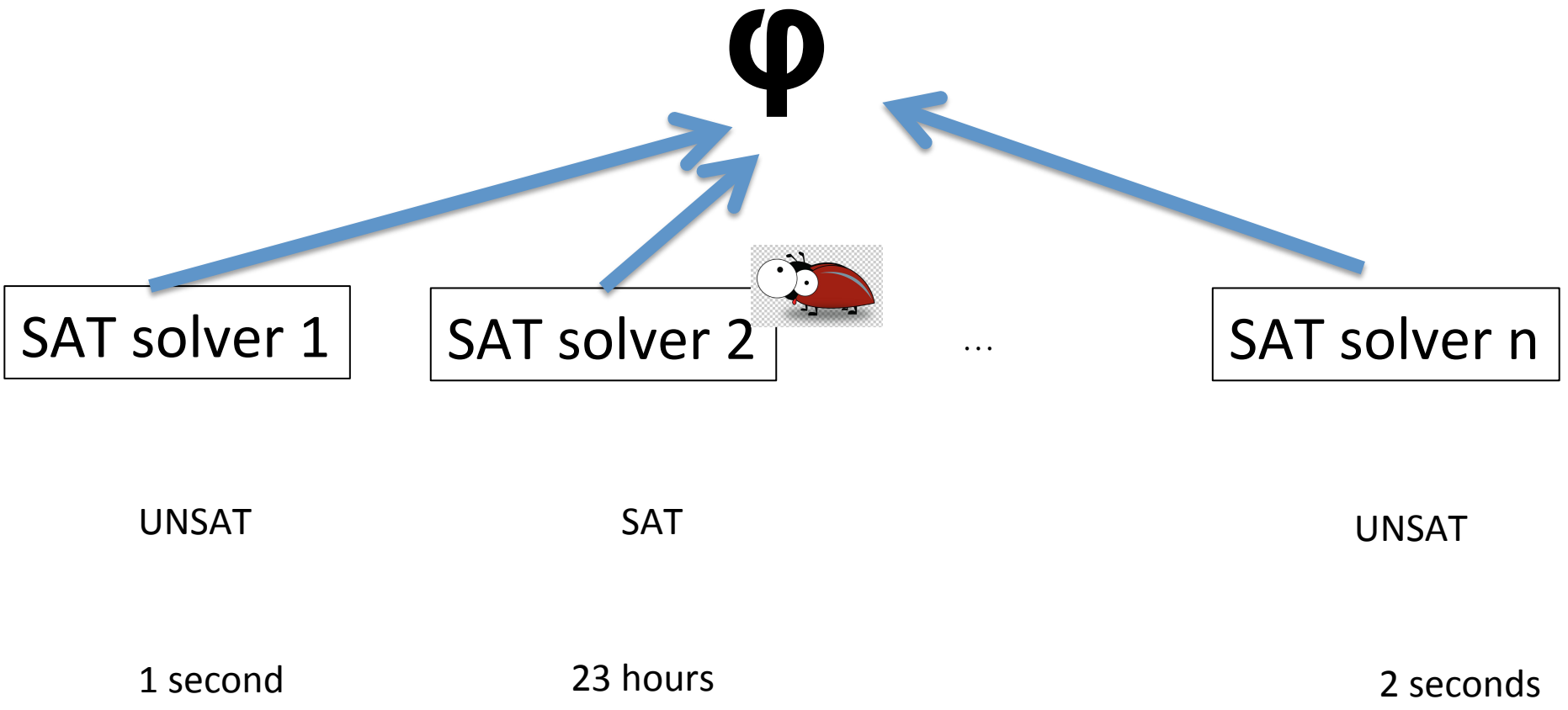
« I want to call SAT solver 2 onto formula phi »



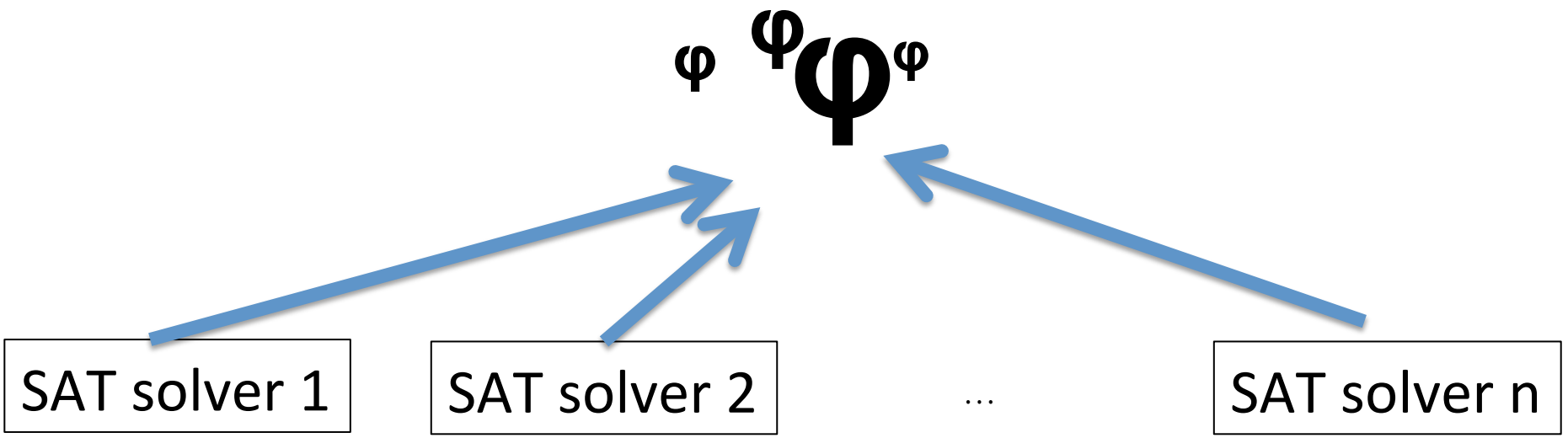
Bugs (and testing your DSL)

functional bugs

performance bugs



Comparing variants' performance



Github repository

- New this year: a common repository
  - We will have the same language
  - Many compilers with a common infrastructure (to test and benchmark them)
  - The work will be reusable
- Pull request
- Intermediate points throughout the course
  - Monitoring the progress
  - Discussing pitfalls or compiler/language design