UFRG
UNIVERSIDADE FED
DO RIO GRANDE D

.inf
INSTITUTO
DE INFORMÁTICA
UFRGS

Verigraph:
A System for Specification and Analysis of Graph Grammars

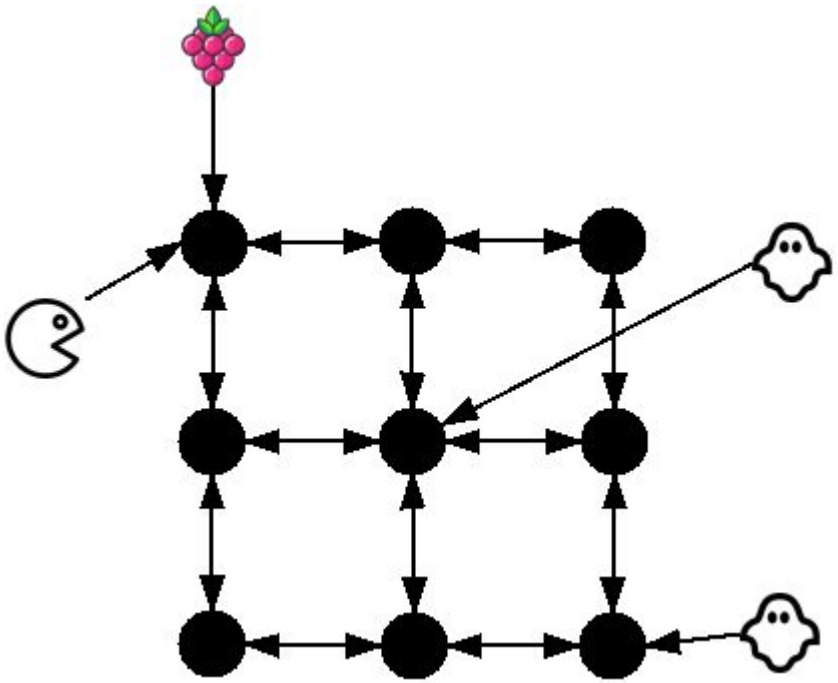
Andrei Costa, Jonas Bezerra, *Guilherme Azzi*, Leonardo Rodrigues, Thiago Becker, Ricardo Herdt, Rodrigo Machado

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

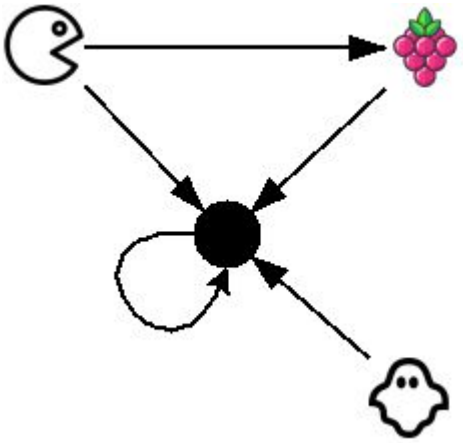
VERITES CERTIFIED

CAPES

Graph Grammars

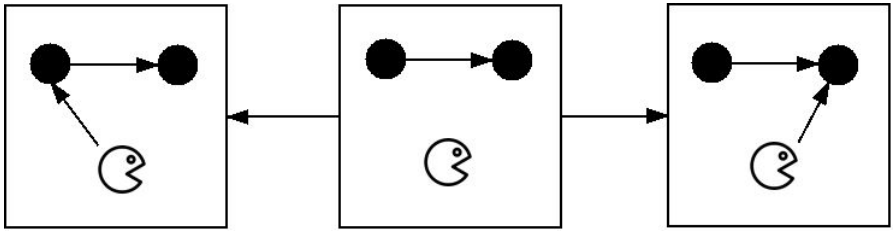


Initial graph

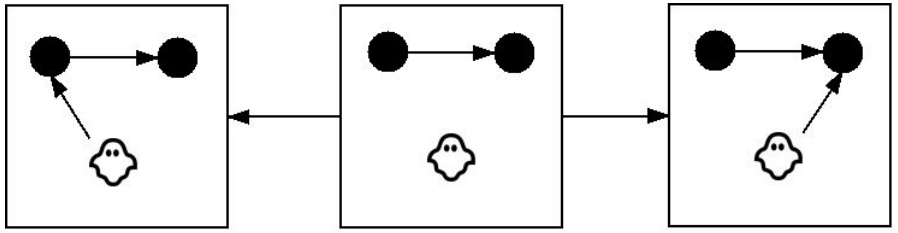


Type graph

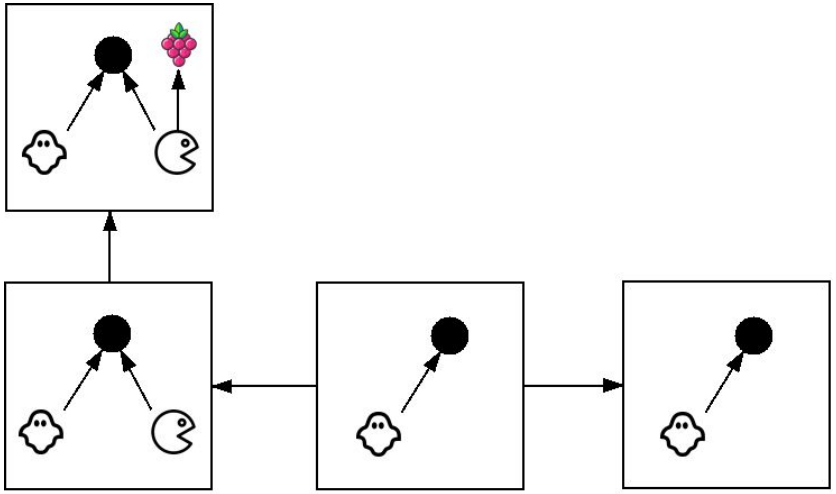
Graph Grammars



Move Pacman

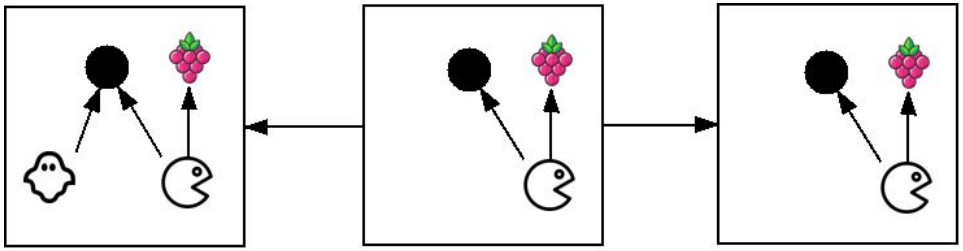


Move Ghost

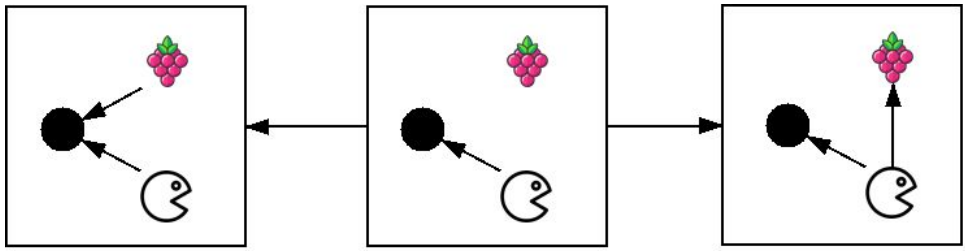


Kill Pacman

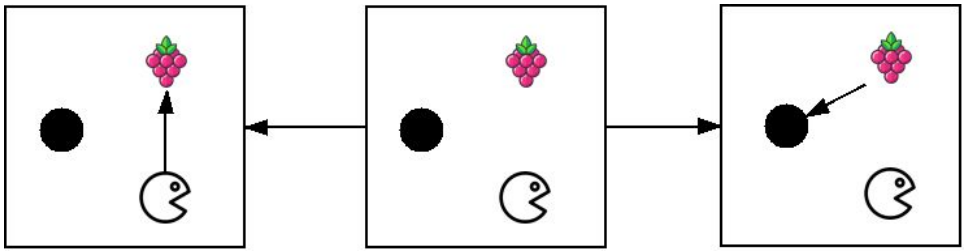
Graph Grammars



Kill Ghost



Get berry



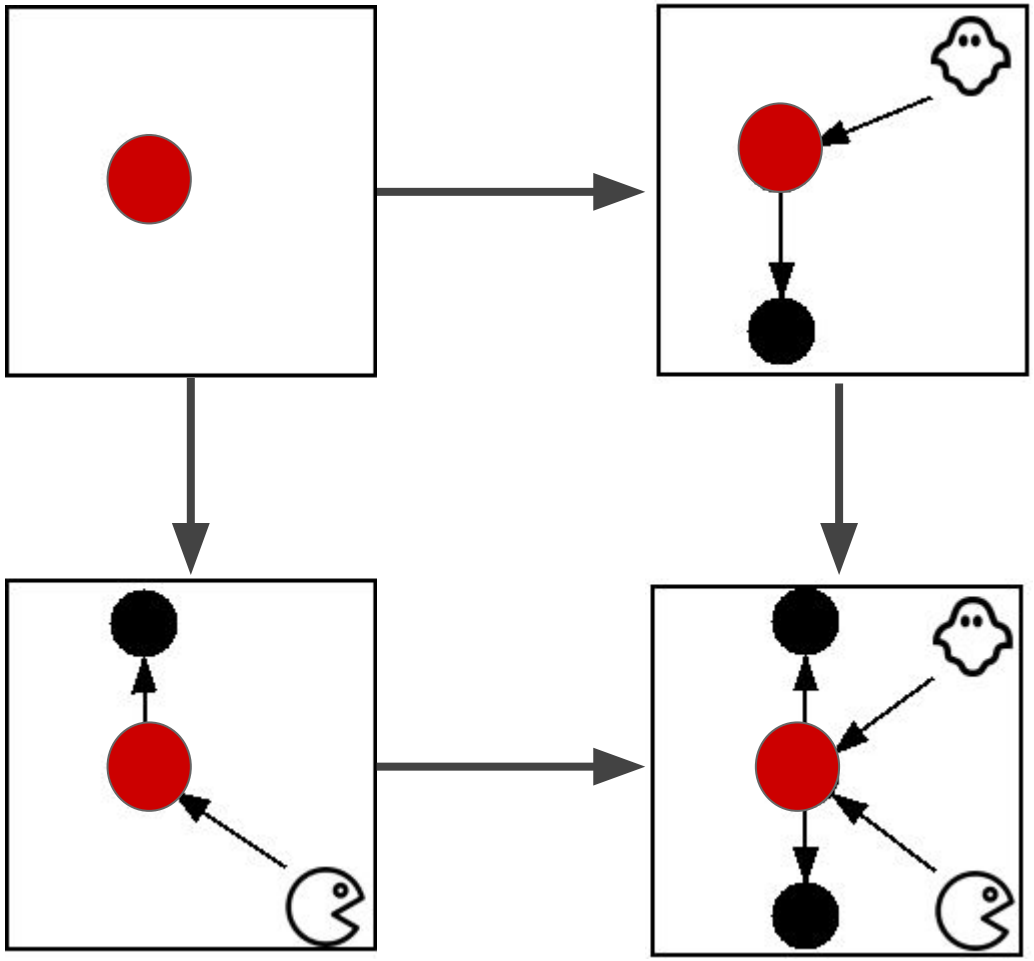
Drop berry

Algebraic Approach

- Definitions based on category theory
- Applicable to: sets, graphs, petri nets, algebraic specs...
- Depends on few categorial constructions: pushouts, pullbacks...

Abstract vs Concrete

- Pushout as gluing construction



Tool Support

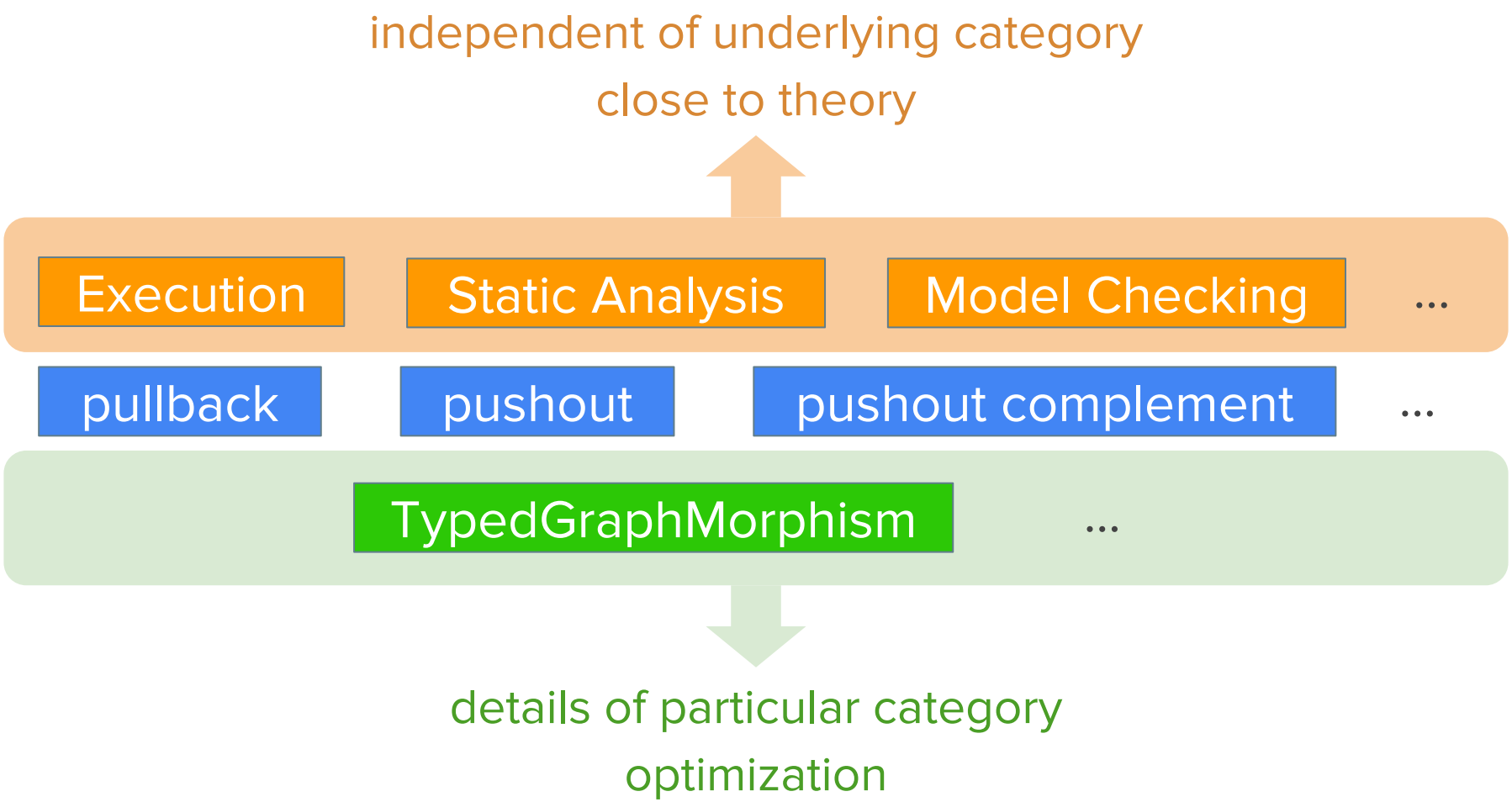
- Execution/simulation (GrGen.Net, GROOVE, AGG…)
- Model Checking (GROOVE…)
- Static Analysis (AGG, Henshin…)
- Model Driven Engineering (Henshin, Fujaba…)

Why another tool?

- Most tools have one notion of graph and transformation approach
- Verigraph: a **sandbox** for graph transformation
  - Open source
  - Written in **Haskell**
  - Architected for **flexibility** and **proximity to the theory**
    - Easier to prove **Correctness**
  - Reasonable **performance**

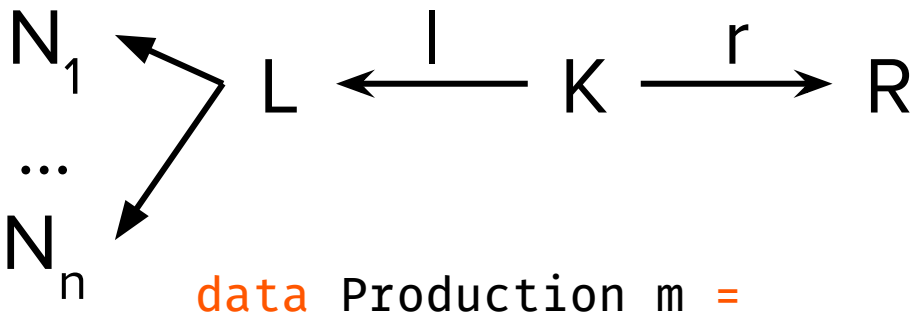
Architected for Flexibility

Clear separation between **abstract** and **concrete** layers



Proximity to Theory

- A **production** is a pair of morphisms with the same origin, along with a collection of NACs

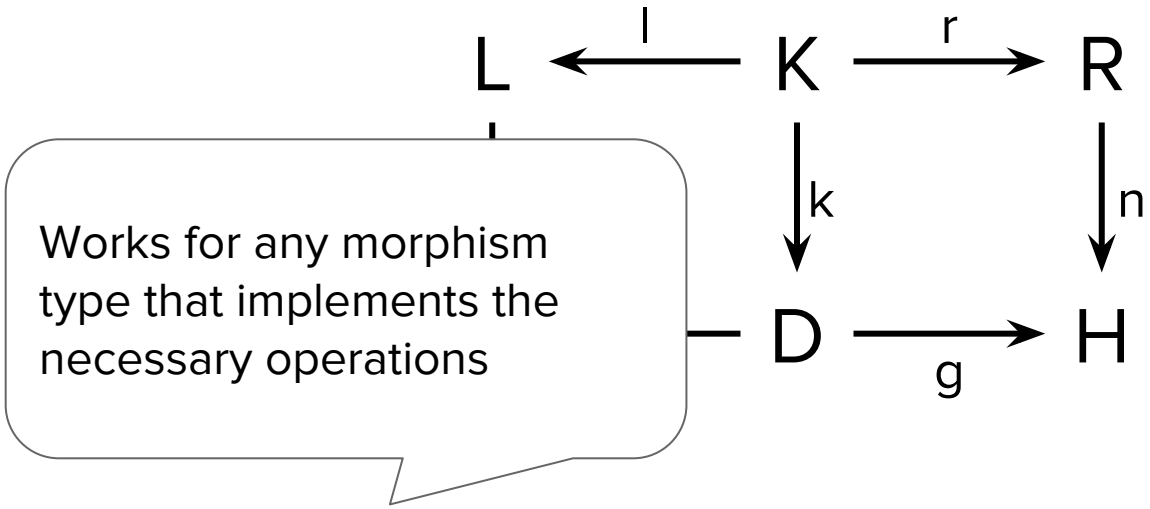



```
data Production m =
  Production {
    left  :: m,
    right :: m,
    nacs :: [m]
  }
```

Generic w.r.t type of morphism

Proximity to Theory

- The double-pushout (DPO) **rewriting** is done by calculating a pushout complement, then a pushout.

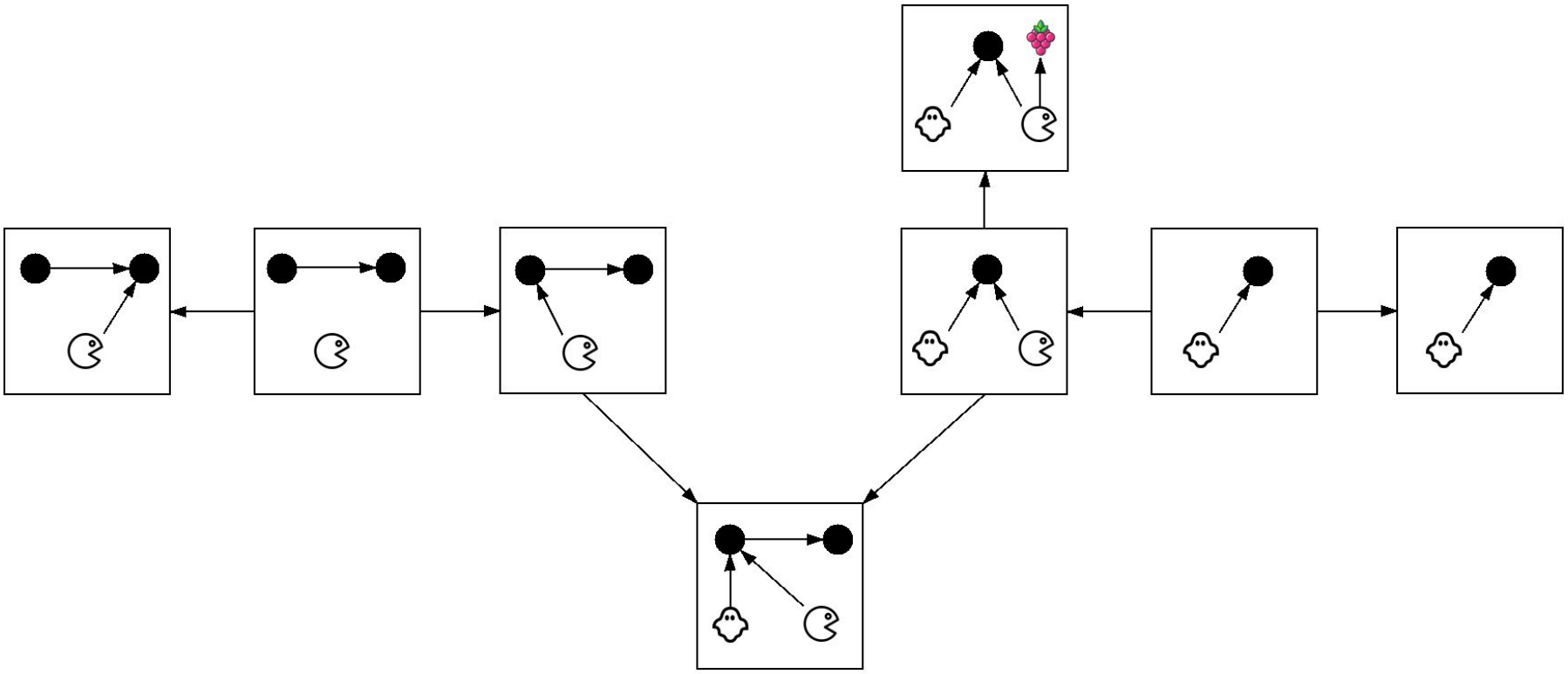


Current Features

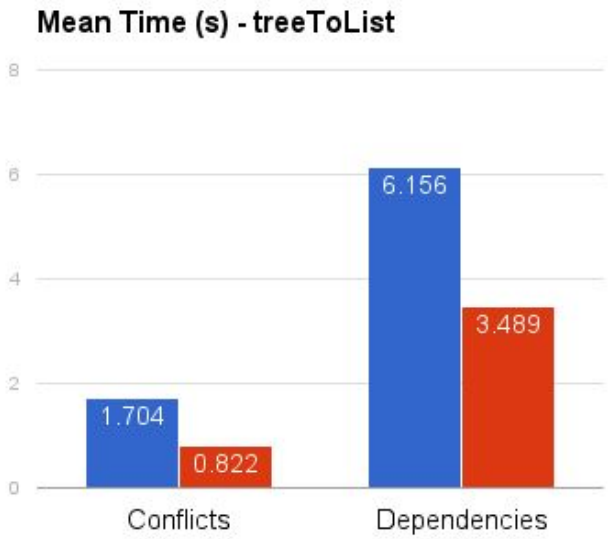
- Static Analysis
  - Conflicts
  - Dependencies
  - Concurrent Rules
- 2nd Order Rewriting

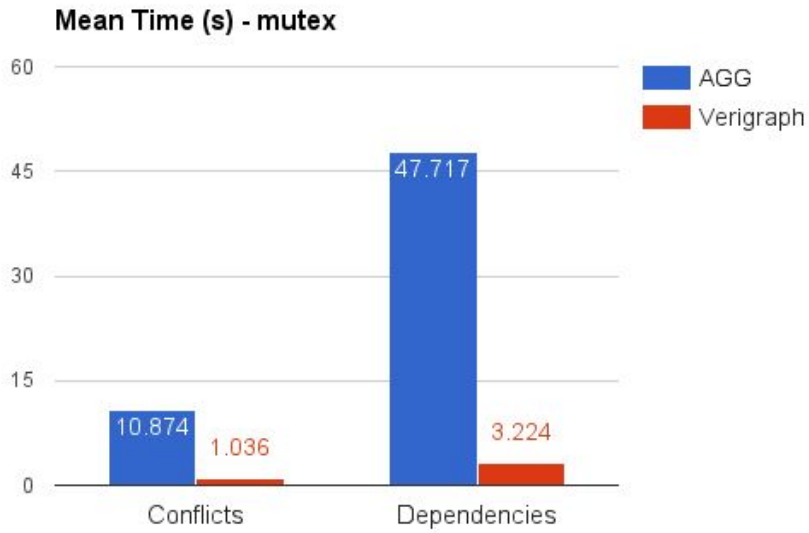
Static Analysis: Conflicts

- Conflict when operations handle **shared resources** in **incompatible ways**
- Defined in terms of **categorial** constructions



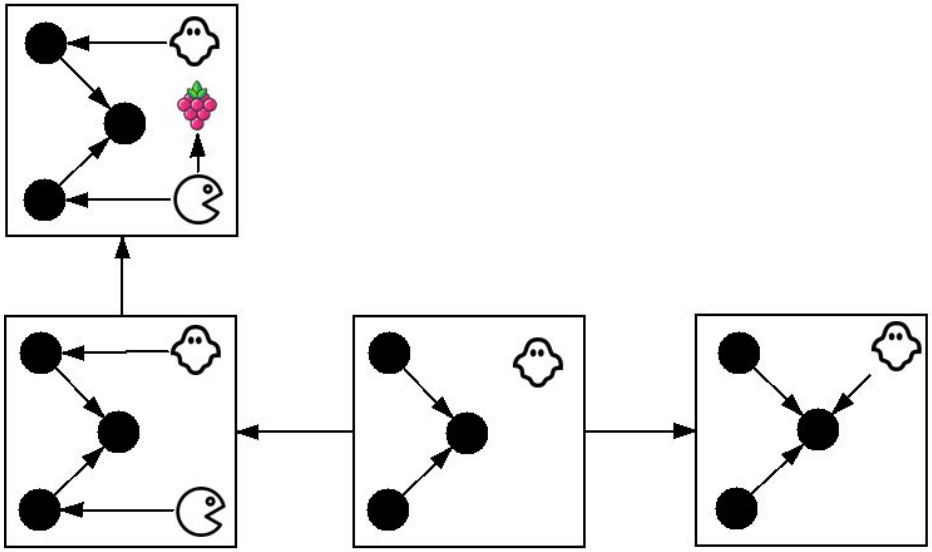
Performance Comparison





Static Analysis: Concurrent Rules

- **Summarize combined behaviour** of multiple operations
- Defined in terms of **categorial** constructions



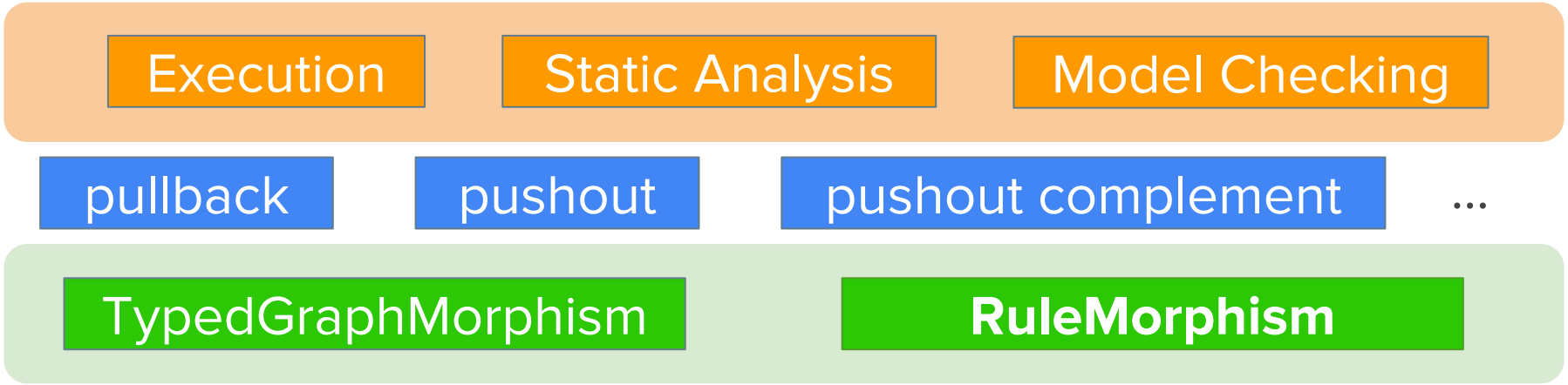
Move Pacman + Move Ghost + Kill Pacman

Modeling Evolution

- System evolves ⇒ behaviour evolves ⇒ rules evolve
- **Rules that modify rules** (Machado, 201)
- **Category** of rules and morphisms between rules
- Same transformation approach of previous rules
- New datatype: **RuleMorphism**

Extensibility: Modelling Evolution

- Morphisms between rules implemented in concrete layer
- Operations of the abstract layer “for free”!



Ongoing/Future Work

- **Rewriting approaches**: SPO, SqPO, AGREE
- **Variations of graphs**: attributed, subtyping
- Further exploration of **2nd order rewriting**
- **Model Checking**
- **Graph Processes**
- Integration with **Theorem Proving** tools
- **Graphical User Interface**

Conclusions

- Verigraph: a sandbox for graph rewriting
  - Extensible
  - Efficient
  - Probably correct

Available at

Source code:

github.com/Verites/verigraph

Tutorial:

github.com/Verites/verigraph/releases/download/1.0.1/tutorial-1.0.1.pdf

Internal API documentation:

verites.github.io/verigraph

Acknowledgement

Thank you!