Solving the TTC'16 Class Responsability Assignment with SIGMA

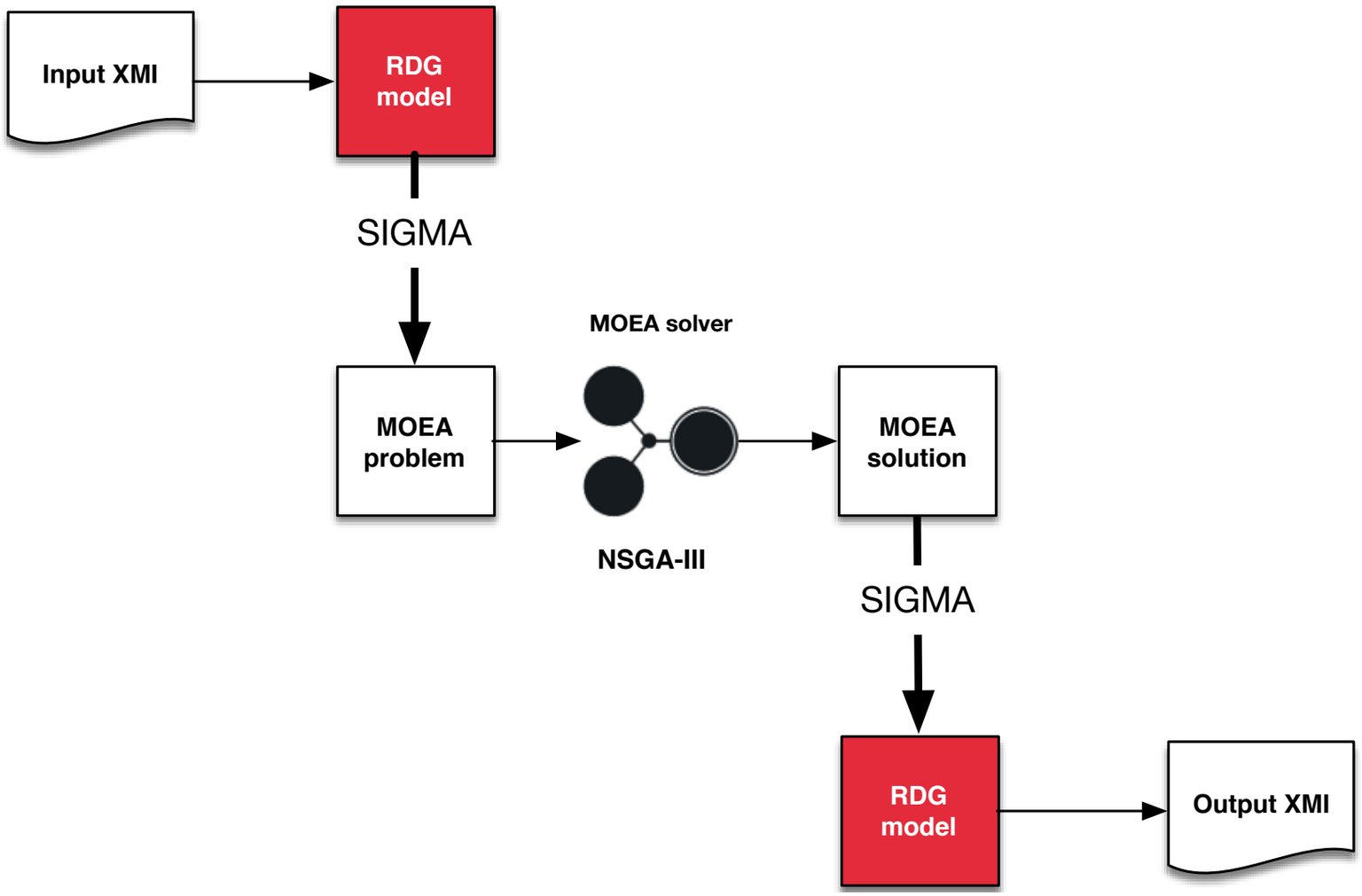
Filip Křikava

Czech Technical University

filip.krikava@fit.cvut.cz

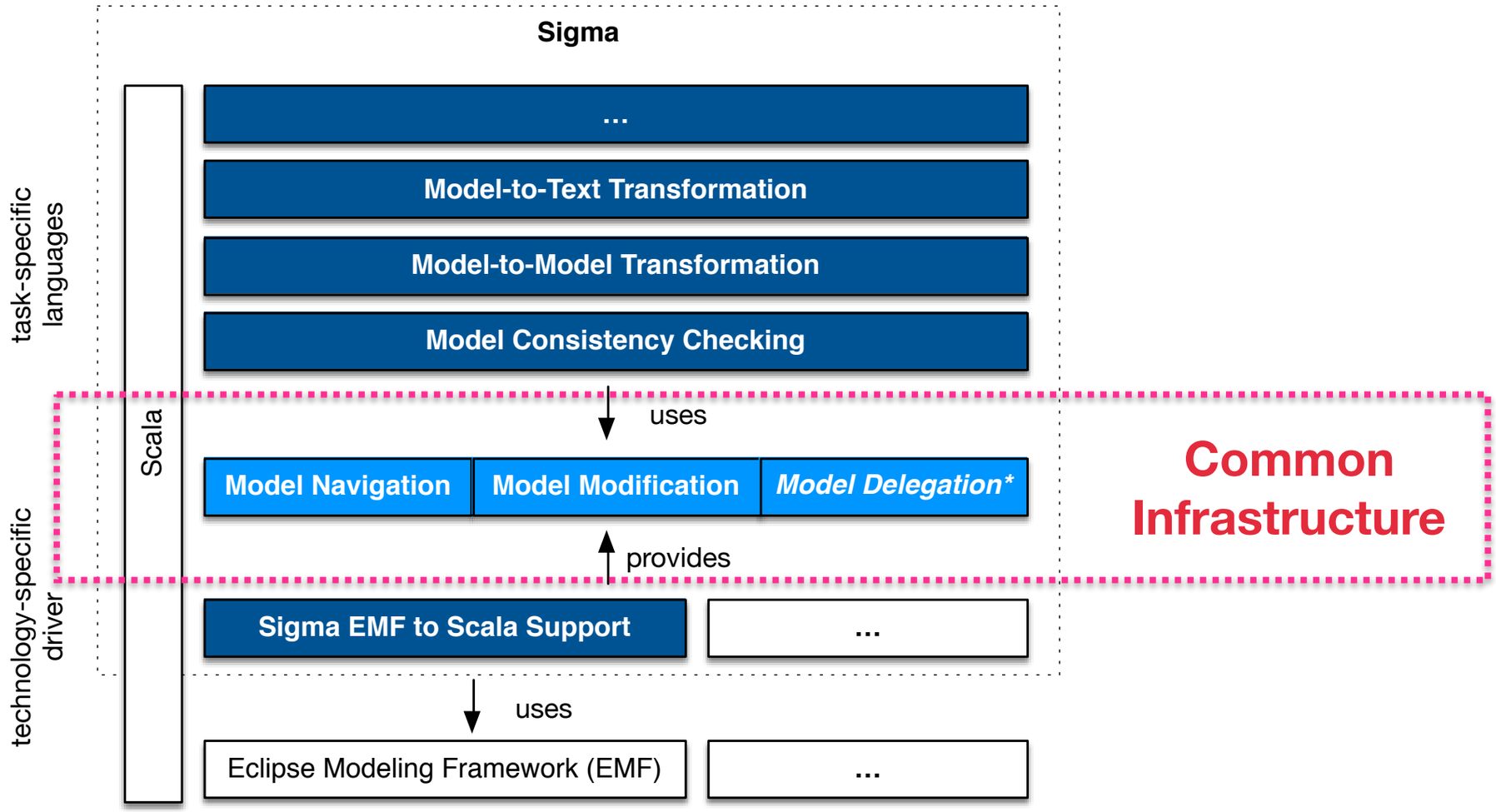
Solution Overview

- Solution for the CRA problem
- SIGMA as the transformation tool, MOEA as the solver



https://github.com/fikovnik/ttc16-cra-sigma

SIGMA[1] Overview



Scala internal DSL for practical model manipulations within a familiar environment with improved usability and performance.

---

[1] F. Křikava, P. Collet, R. France, *SIGMA: Scala Internal Domain-Specific Languages for Model Manipulations*, In Proceedings of the 17th International Conference on Model Driven Engineering Languages and Systems (MODELS '14), 2014

- Mixes function programming with OOP
- JVM compatible
- Designed to host DSLs
- Statically typed with type inference
- “*look-and-feel*” of a dynamic language
- Well supported by major tool vendor

---

http://www.scala-lang.org

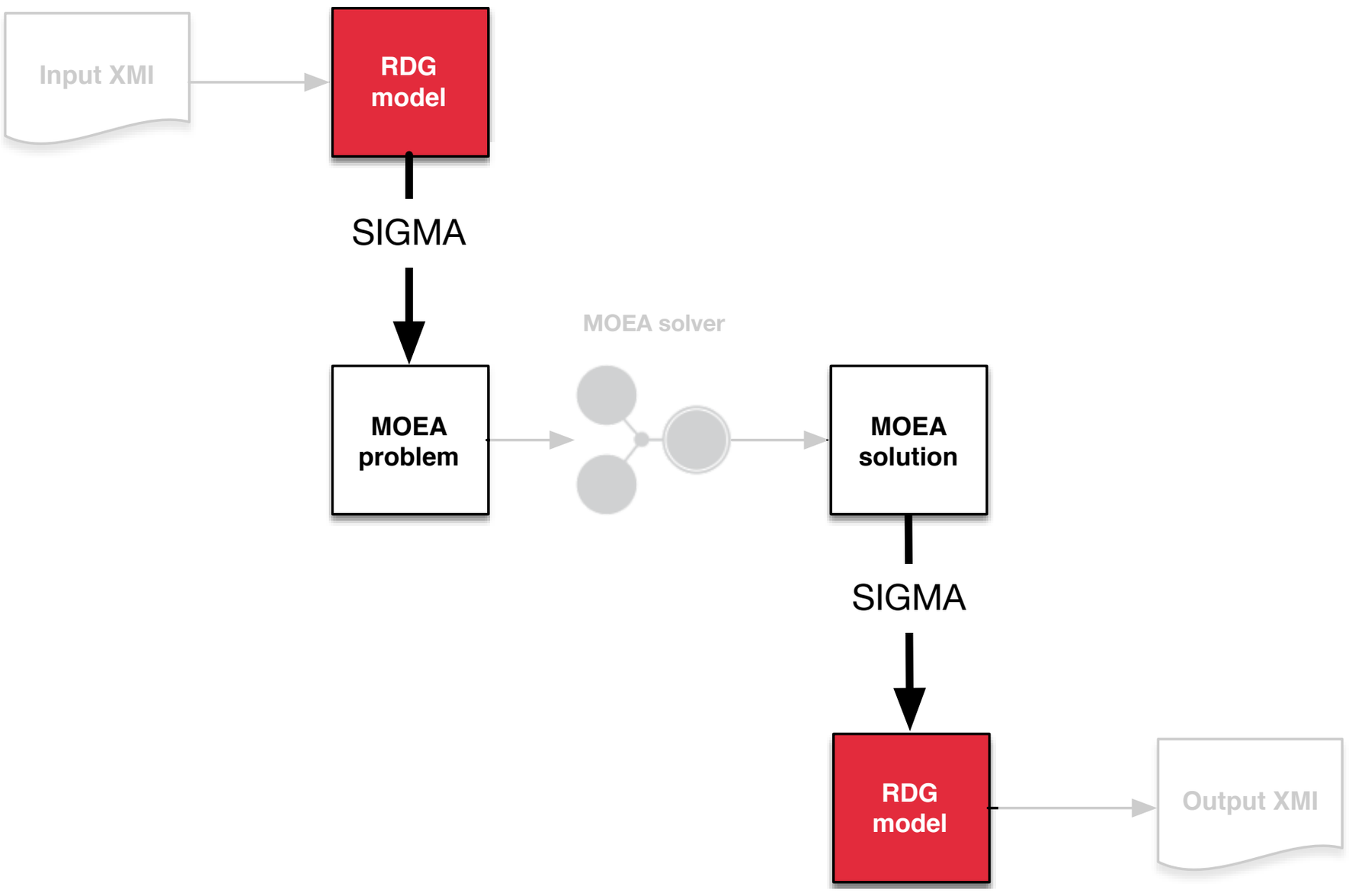
MOEA Framework

- Java library for experimenting with multiobjective evolutionary algorithms
- Open source
- Easy to use, stable, available
- Includes number of algorithms
- MOEA is just one possibility
  - **Use model transformation to transform problem from one domain to another**

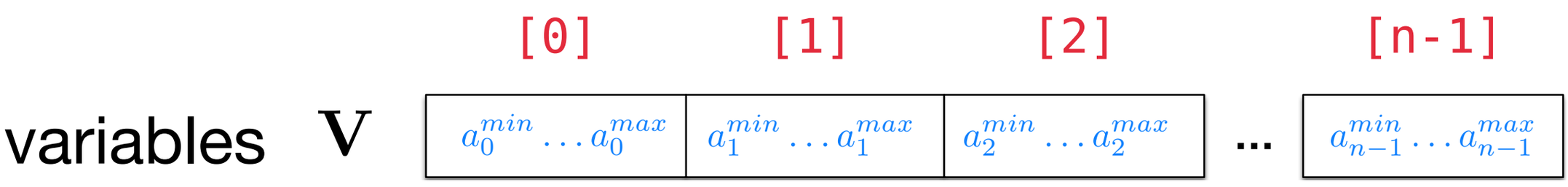
---

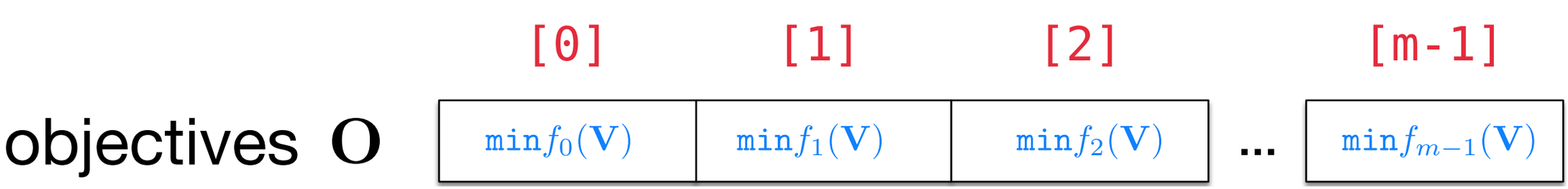
http://moeaframework.org/

Approach


Input XMI
RDG model
SIGMA
MOEA solver
MOEA problem
MOEA solution
SIGMA
RDG model
Output XMI

Mapping RDG to MOEA and Back

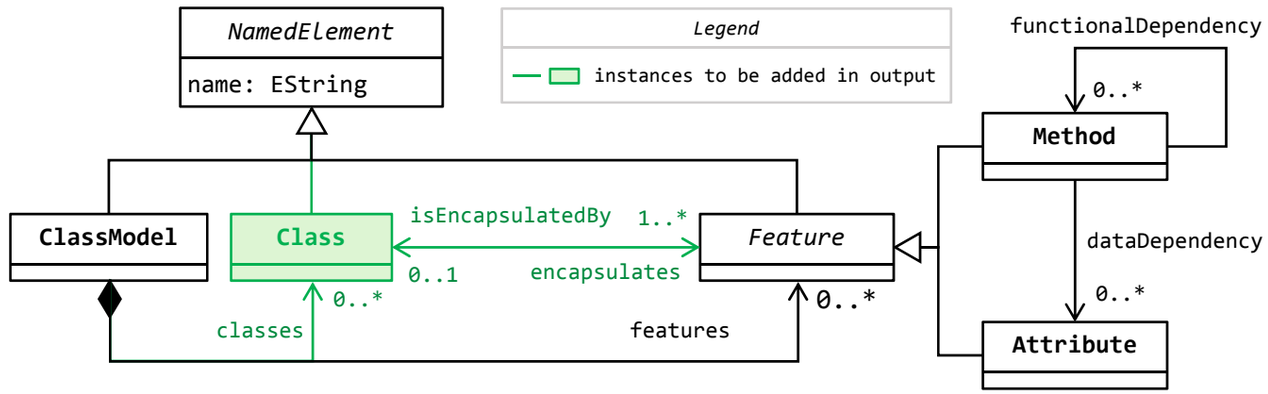




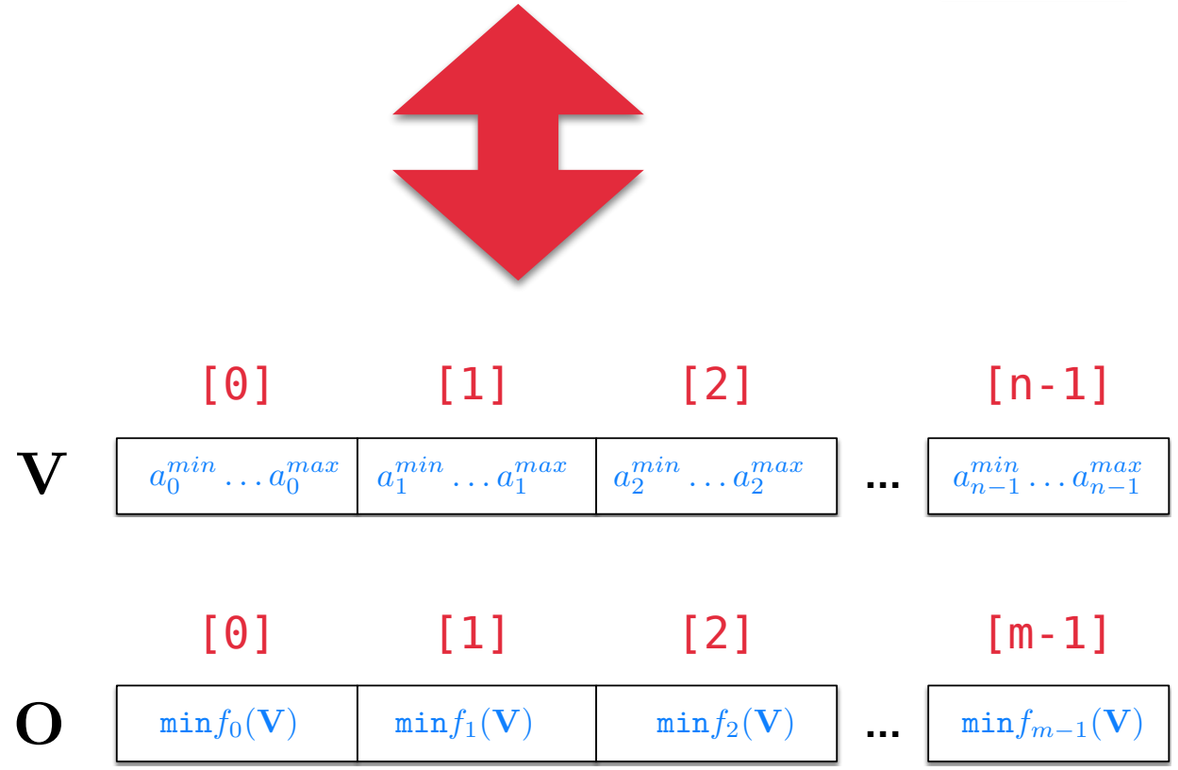
one possible MOEA Problem

Mapping RDG to MOEA and back

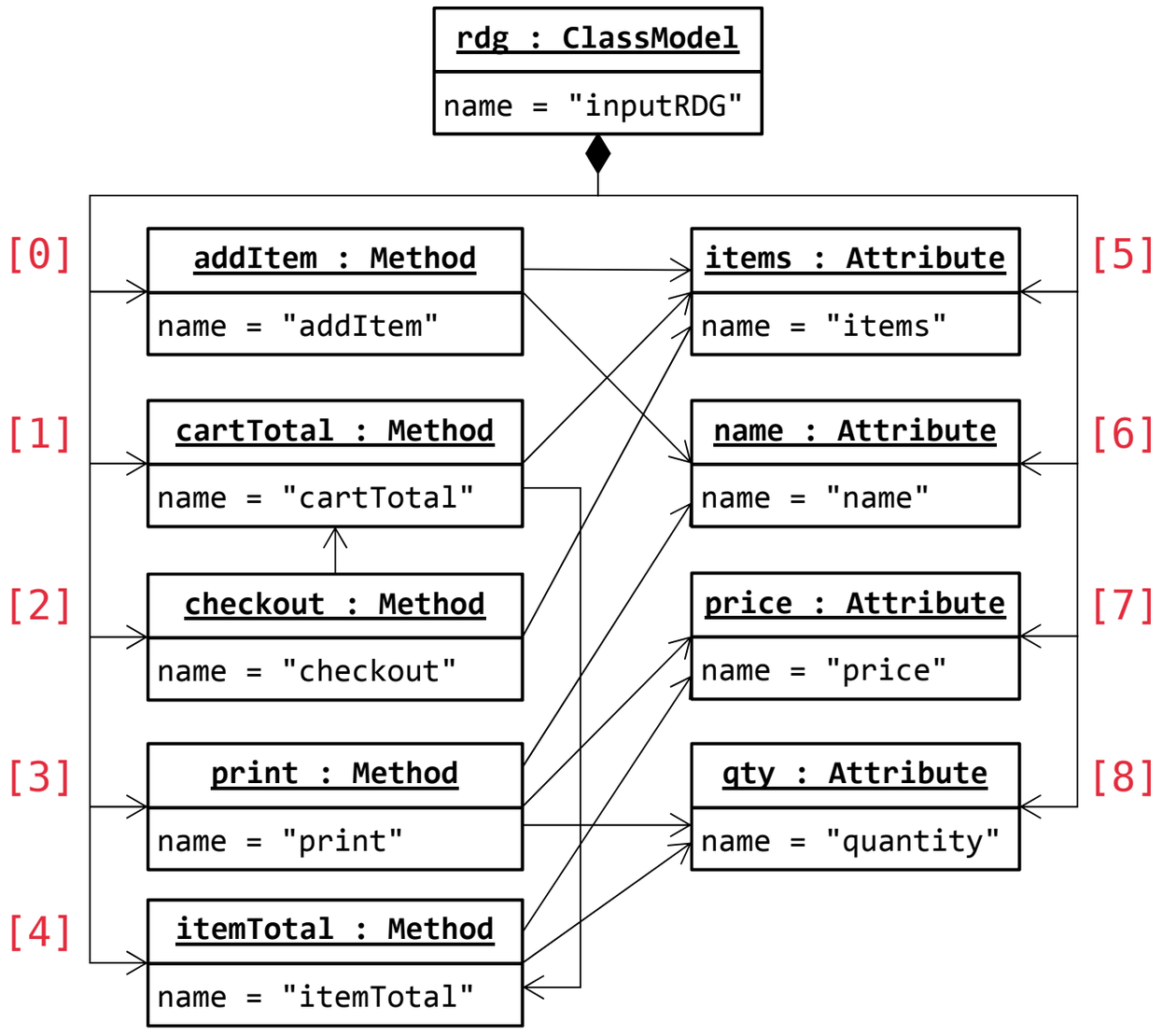
RDG Model



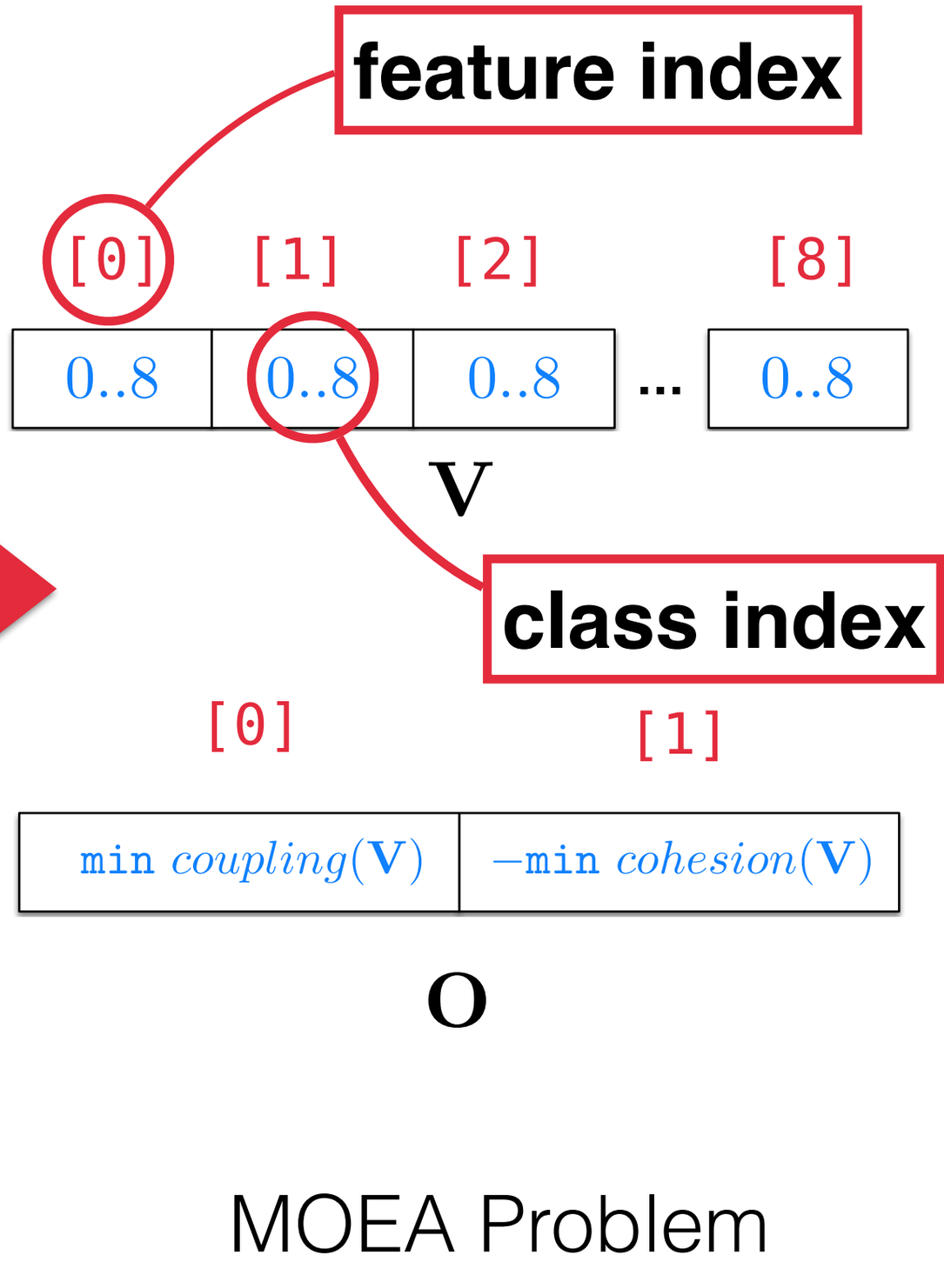
MOEA Problem



Mapping RDG to MOEA

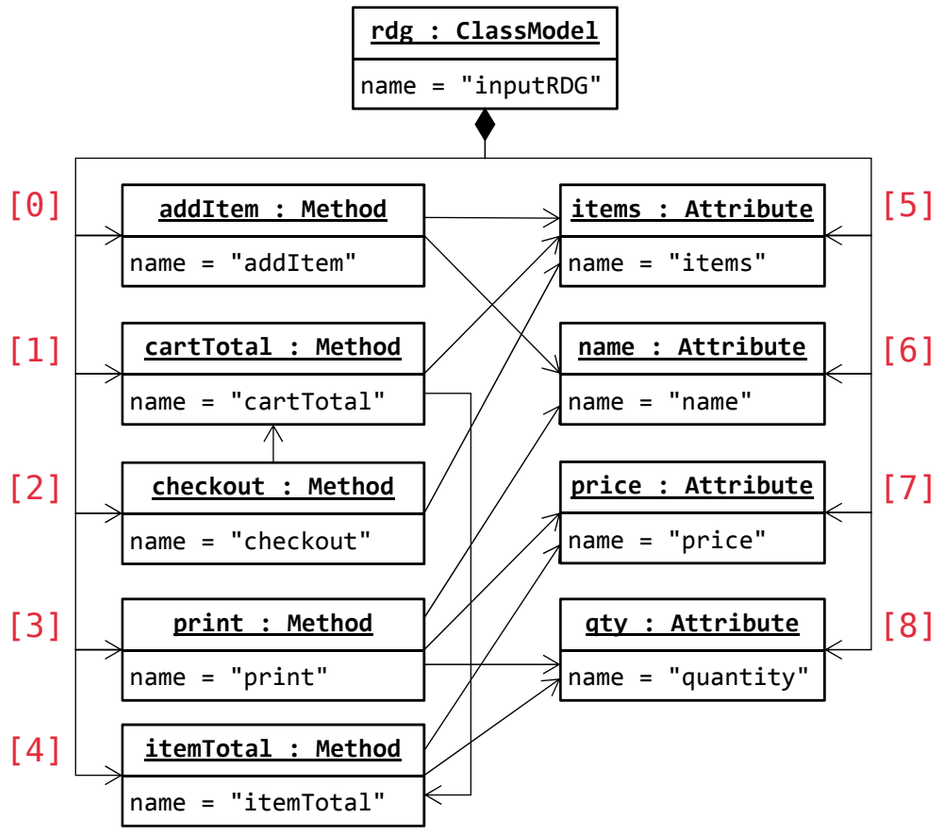


RDG model

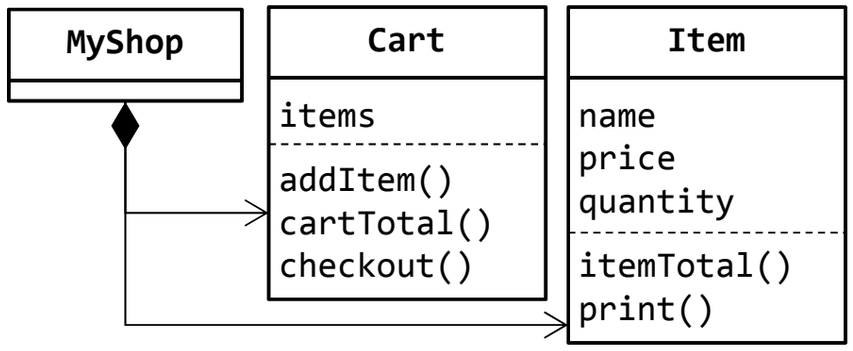


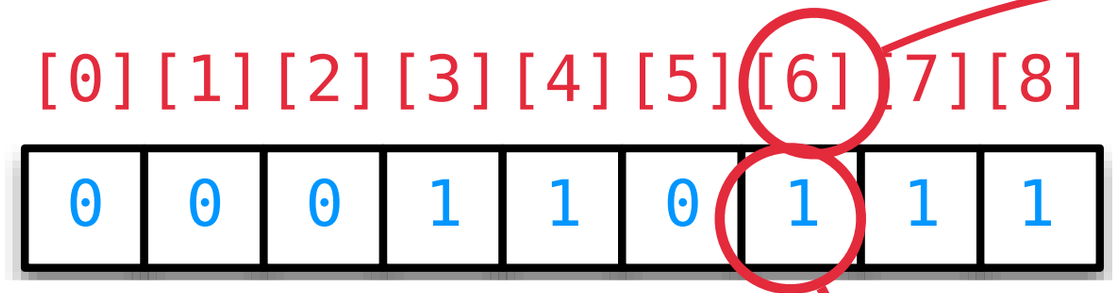
MOEA Problem

Mapping MOEA to RDG



RDG (CD) model





feature index

class index

MOEA Problem

Implementation

```
def optimize(numRuns: Int, initModel: ClassModel): ClassModel = {
  val solver = new NSGAIII(initModel)
  (1 to numRuns).map (_ => solver())
                .maxBy(model => calculateCRAIndex(model))
}
```

Implementation

```
class NSGAIII(val initModel: ClassModel) extends Solver {

  private val mutationRate = 1.0 / initModel.features.size
  private val crossoverRate = .7

  def apply() =
    create().withAlgorithm("NSGAIII")
            .withProperty("pm.rate", mutationRate)
            .withProperty("sbx.rate", crossoverRate)
            .run()
}
```

Implementation

```
class CRAProblem(val initModel: ClassModel)
  extends AbstractProblem(initModel.features.size, 2) {

  def newSolution(): Solution = {
    val solution = new Solution(numberOfVariables, numberOfObjectives)

    (0 until numberOfVariables).foreach { x =>
       solution.setVariable(x, newInt(0, numberOfVariables - 1))
    }

    solution
  }

  def evaluate(solution: Solution): Unit = {
    val model = CRAProblem.solutionToClassModel(initModel, solution)

    solution.setObjective(0, CRAIndexCalculator.calculateCoupling(model))
    solution.setObjective(1, -CRAIndexCalculator.calculateCohesion(model))
  }
}
```

Implementation

```
def solutionToClassModel(
  initModel: ClassModel,
  solution: Solution): ClassModel = {

  val model = initModel.sCopy
  val rawSolution = EncodingUtils.getInt(solution)
  val classes = (0 to rawSolution.max) map (x => Class(name = s"Class $x"))

  rawSolution.zipWithIndex.foreach {
    case (classIdx, featureIdx) =>
      model.features(featureIdx).isEncapsulatedBy = classes(classIdx)
  }

  model.classes ++= classes filterNot (x => x.getEncapsulates.isEmpty)
  model
}
```

Results - SHARE

Input	Cohesion	Coupling	CRA
A	4	1	3
B	6.5	2.5	4
C	6.37	3.63	2.74
D	4.83	7.94	-3.11
E	7.38	17.99	-10.60
F	9.85	44.74	-34.88

Input	Time [s]
A	19.17
B	34.78
C	72.53
D	300.49
E	1110.74
F	6289.75

Results - Laptop

Table 1

Model	Cohesion	Coupling	CRA	Time [s]	Runs
TTC_InputRDG_A	4	1	3	10	10
TTC_InputRDG_B	6.67	2.58	4.08	13	10
TTC_InputRDG_C	4.99	2.57	2.41	39	10
TTC_InputRDG_D	9.36	8.50	0.87	91	10
TTC_InputRDG_E	7.14	16.25	-9.11	1416	10
TTC_InputRDG_F	9.87	39.37	-29.50	234	1
TTC_InputRDG_G	9.76	369.44	-359.68	1213	1
TTC_InputRDG_H	13.49	1393.56	-1380.06	79778	1

All runs on java version "1.8.0_74", OSX 10.11.5, 3,1 GHz Intel Core i7

Conclusion

- A concise solution with reasonably good performance
- Lot of opportunities for tuning NSGA-III
- Uses SIGMA for EMF model manipulation

https://github.com/fikovnik/ttc16-cra-sigma

Internal DSL for Model Manipulation

- External model manipulation DSLs
  - embed a GPL into a model-manipulation DSL
- Internal model manipulation DSLs
  - embed model manipulation DSL into a GPL
    - increased level of abstraction
    - similar features, expressiveness and versatility
    - improved tool support, interoperability and performance
    - low engineering cost

but all depends on the host language