TECHNISCHE UNIVERSITÄT DRESDEN

LUND UNIVERSITY

Software Language Engineering 2018

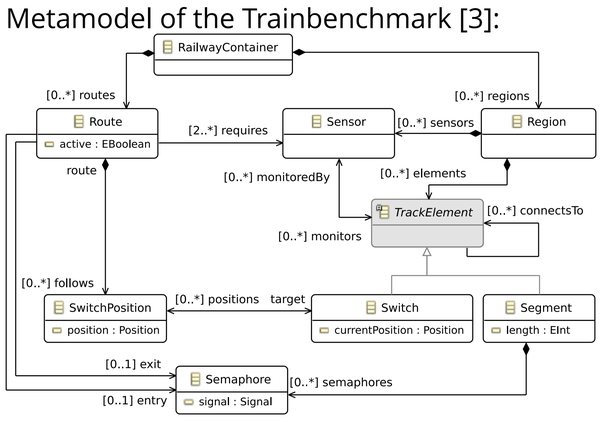
Continuous Model Validation using Reference Attribute Grammars

Johannes Mey[1], René Schöne[1], Görel Hedin[2], Emma Söderberg[2], Thomas Kühn[1], Niklas Fors[2], Jesper Öqvist[2], and Uwe Aßmann[1]

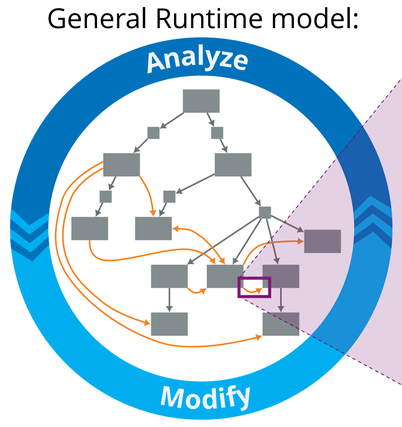
[1]Technische Universität Dresden
[2]Lund University

Problem: Efficient Navigation of Non-containment References

Metamodel of the Trainbenchmark [3]:



General Runtime model:



- RAGs [1] can be used for models@runtime providing some advantages
  - *Separation of structure and computation*
  - *Shorthands* for navigation and computation on trees
  - *Efficiency* through memoization of computed values
  - *Incremental computation*, i.e., invalidating of outdated cached values
- **Major issue:** Non-containment references can not be encoded *efficiently* w.r.t. performance and consistency

State-of-the-Art Using RAGs and JastAdd [2]

```
Region : RailwayElement ::= TrackElement* Sensor*;
Route : RailwayElement ::= <Active:boolean> SwitchPosition* requires:Ref*
    [entry:Ref] [exit:Ref];
SwitchPosition : RailwayElement ::= <Position:Position> target:Ref;
Sensor : RailwayElement ::= monitors:Ref*;
abstract TrackElement : RailwayElement ::= connectsTo:Ref*;
```

```
syn Collection<Sensor> Route.requires() {
  java.util.List<Sensor> sensors = new ArrayList<>();
  for (SensorRef sensorRef : getRequiredSensorList())
    sensors.add(sensorRef.getSensor());
  return sensors;
}
```

Name Lookup

```
Ref ::= <Value:int>
SensorRef : Ref;
```

Idiomatic Approach

```
syn Sensor SensorRef.getSensor() {
  for (Sensor s :
      getRoot().sensors())
    if (s.id() == this.getValue())
      return s;
  return null;
}
```

Idiomatic Approach with a Map

```
syn Map<Integer, Sensor> RailwayContainer.sensorMap() {
  Map<Integer, Sensor> sensors = new HashMap<>();
  for (Region region : getRegionList()) {
    for (Sensor sensor : region.getSensorList())
      sensors.put(sensor.id(), sensor);
  }
  return sensors;
}
syn Sensor SensorRef.getSensor() =
  getRoot().sensorMap().get(this.getValue());
```

Intrinsic References

```
abstract Ref ::= <Value:int> ;
SensorRef : Ref ::= <Sensor:Sensor>;
```

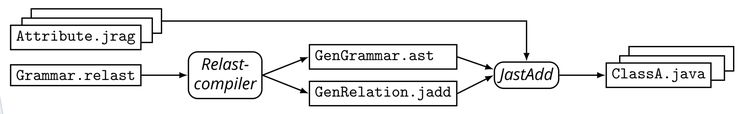
- Name-Analysis done in deserializer and after every edit
- Faster evaluation and less cache invalidation

Grammar Extension

Introducing Relations

```
rel Route.requires* -> Sensor;
rel Route.entry? -> Semaphore;
rel Route.exit? -> Semaphore;
rel SwitchPosition.target <-> Switch.positions*;
rel Sensor.monitors* <-> TrackElement.monitoredBy*;
rel TrackElement.connectsTo* -> TrackElement;
```

Adding the Preprocessor *RelAst*

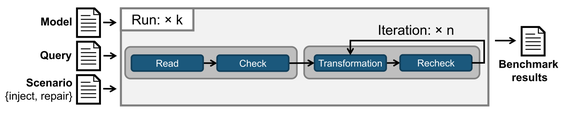


Generated API

```
public java.util.List<Sensor> Route.requires() {
  RefList<Sensor> l = get_impl_requires();
  return l != null ? Collections.unmodifiableList(l) :
      Collections.emptyList();
}
public void Route.addToRequires(Sensor o) { /* ... */ }
public void Route.removeFromRequires(Sensor o) { /* ... */ }
```

Evaluation within the Trainbenchmark

Trainbenchmark Process

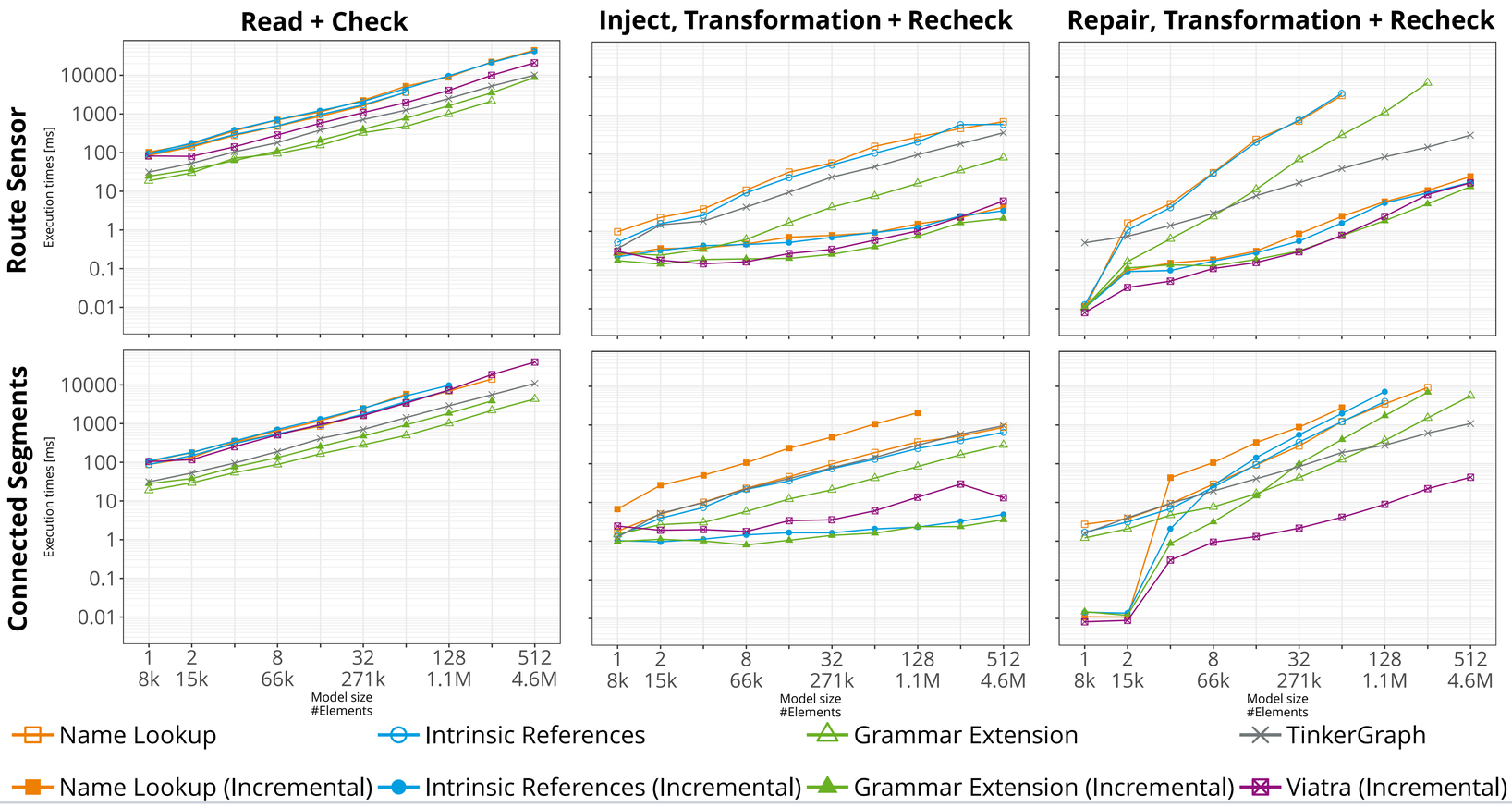


Advantages of our Grammar Extension

- Less boilerplate code to write

```
  public void inject(Collection<Match> matches) {
-   List<SensorRef> refsToRemove = new ArrayList<>();
    for (final Match match : matches) {
-     for (SensorRef ref : match.getRoute()
-         .getRequiredSensors()) {
-       if (ref.getSensor() == match.getSensor()) {
-         refsToRemove.add(ref);
-       }
-     }
-   }
-   for(SensorRef ref : refsToRemove) {
-     ref.removeSelf();
+     match.getRoute().removeRequiredSensor(
+       match.getSensor());
    }
    driver.flushCache();
  }
```

- More consistency
- Better performance



[1] Görel Hedin. Reference attributed grammars. *Informatica (Slovenia)*, 24(3):301–317, 2000.
[2] Görel Hedin and Eva Magnusson. JastAdd: an aspect-oriented compiler construction system. *Science of Computer Programming*, 47(1):37–58, 2003.
[3] Gábor Szárnyas, Benedek Izsó, István Ráth, and Dániel Varró. The Train Benchmark: cross-technology performance evaluation of continuous model queries. *Software & Systems Modeling*, pages 1–29, January 2017.

Acknowledgments: This work is partly supported by the German Research Foundation (DFG) in the SFB 912 "Highly Adaptive Energy-Efficient Computing", the project "RISCOS" and within the Research Training Group "Role-based Software Infrastructures for continuous-context-sensitive Systems" (GRK 1907), and by the German Federal Ministry of Education and Research within the project "OpenLicht". This work is also partly supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) in the PIIA project 2017-02371 and by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation (KAW).

Contact:
Johannes Mey johannes.mey@tu-dresden.de
René Schöne rene.schoene@tu-dresden.de
Thomas Kühn thomas.kuehn3@tu-dresden.de
Uwe Aßmann uwe.assmann@tu-dresden.de
Görel Hedin gorel.hedin@cs.lth.se
Emma Söderberg emma.soderberg@cs.lth.se
Niklas Fors niklas.fors@cs.lth.se
Jesper Öqvist jesper.oqvist@cs.lth.se