An Algorithmic Solution to the Vector Challenge

Arne Lange | Romain Pascual

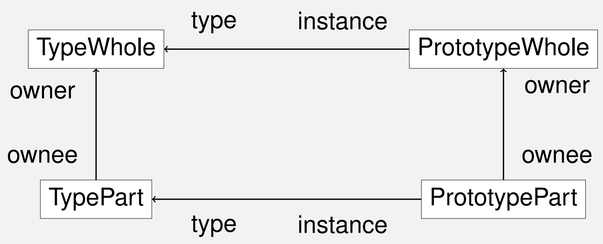
Karlsruhe Institute of Technology

CentraleSupélec | université PARIS-SACLAY

Vector's Multi Challenge

Given some STO declarations in M_2 and an instance model in M_1, check if every Type-Square is a valid STO instance.

Type-Square given by:



Contributions

1. Formal characterization of STO (nested application conditions).
2. STO reduced to three local constraints.
3. Generic EMF checker derived from the formalization.
4. Evaluation on models up to 1.8M Type-Squares.

STO Formal Description

Models are represented as attributed typed graphs. An STO is a pattern involving two orthogonal relations:

- Classification (Type Instance)
- Composition (Part Whole)

STO-conformance: An instance of a Type-Square is a well-formed STO iff if satisfies the following constraints:

Existence Every Prototype Whole owns at least one Prototype Part

Congruence The following equation holds for every Prototype Part:
typeof(owner(part)) = owner(typeof(part))

Unique Realization There is a unique Prototype Part for a Prototype Whole and a Type Part sharing the same Type Whole.

Algorithm

Input: STO declarations in the metamodel & instance model.

Procedure:

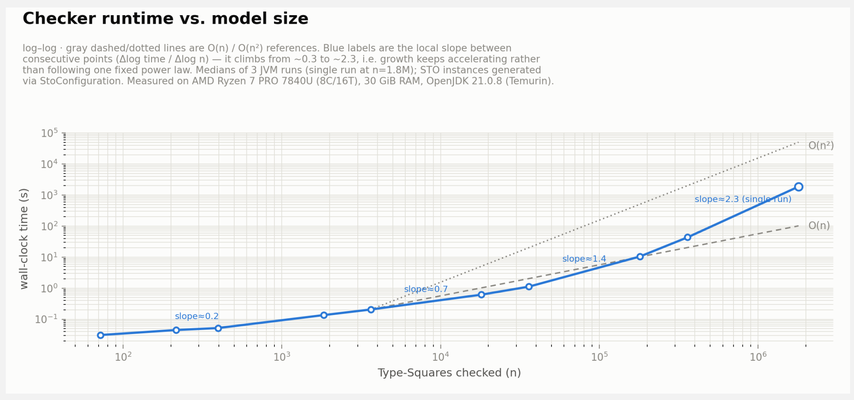
1. Discover Type-Squares:
   - traverse the containment tree
   - identify every known STO Type-Square
2. Check structural preconditions (stop if any precondition fails):
   - every instance has at most one type
   - every part has at most one whole
3. Check STO conformance for every Type-Square:
   - existence → congruence → unique realization.

Implementation

- Built on Karl Kegel's EMF utilities (base repo) for generic EObject navigation
- List of STO extracted from the metamodel
- Global two-phase verification: preconditions then STO conformance.
- Checker built as a layered rule engine:
  existence → congruence → unique realization.
- Independent rule architecture:
  Each STO constraint is implemented as its own validation rule: either record all violations or stop at the first one, easy to extend with additional constraints.

Scan to access the checker

Evaluation



- Benchmarks: 72 up to 1.8M Type-Squares (10 sizes, log-spaced)
- Runtime seems to grow exponentially but still below $O(n^2)$ on the largest sizes
- Dominant cost is the uniqueness constraint, which re-fetches the full instance list of a shared type for every square

Discussion & Perspectives

1. **Well-formedness principles**: STO conformance reduces to three graph constraints: checker directly derived from the constraints.
2. **Integrity conditions**: what & when: All rules (preconditions and STO conformance) can be checked at any time: immediate feedback during model construction.
3. **Language constructs/patterns**: Presented approach is (E)MOF compliant and relies on attributed typed graphs with nested application conditions as formal background.
4. **Attributes**: Focus on the structural aspects of STO. Attribute propagation would follow the existing MULTI typing semantics without STO-specific mechanisms.
5. **Model evolution**: Changes to instance models (M_1): STO conformance can be checked after edit. Changes to the metamodel: STO structure can be altered; conformance of the affected instances may require model edits.

Future Work: Parallelization of the checker and Incremental checking via verification of edit rules (calculus of nested application conditions)