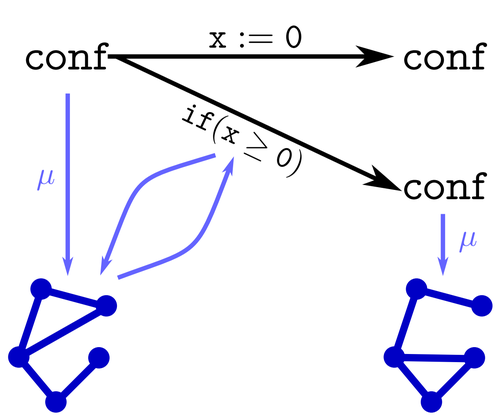
Type Checking Semantically Lifted Programs via Query Containment under Entailment Regimes

Eduard Kamburjan **Egor V. Kostylev**

Semantically Lifted Programs (SLP)

An SLP maps its program state to a knowledge graph and enable **semantic reflexion**: Performing queries on its own configuration using semantic web technologies.



The direct mapping μ results in an OWL knowledge graph, where the TBox contains axioms from two sources: Axioms $\mathcal{T}_{\mathsf{conf}}$ describing the configuration and **domain knowledge** $\mathcal{T}_{\mathsf{user}}$ provided by the user. Both TBoxes are static and do not change during execution.

Challenge

The following models a cloud system that uses the **domain knowledge** about overloaded platforms to dynamically reschedule tasks.

```
class Platform(List<Server> serverList) ... end
class Server(List<Task> taskList) ... end
class Scheduler(List<Platform> platformList)
 Unit reschedule()
  List<Platform> l
     := access("SELECT ?x WHERE {?x a Overloaded}");
  this.adaptPlatforms(l);
 end
end
```

Will the above query **always** return a list of `Platform` instances?

⇒ Depends on axioms for `Overload` and the entailment regime
⇒ Type system and TBox **interact**

How to ensure that a query access is type-safe, if part of the knowledge requires reasoning about the user-provided TBox?

Typing Semantically Lifted State Access using Query Containment

The first approach is to statically check whether the query is contained in `{?x a T}`, as this query trivially ensures type safety at runtime. This must take into account the SPARQL entailment regime er that is used for query answering. In the, slightly simplified, rule below, the first premise checks that the query (as a formula $\exists \bar{y}.\, \phi(x)$) is contained in the query that selects the target type ($S_{\mathsf{T}'}(x)$) under TBox $\mathcal{T} = \mathcal{T}_{\mathsf{conf}} \cup \mathcal{T}_{\mathsf{user}}$.

$$\textbf{(acc-type)}\ \frac{\exists \bar{y}.\phi(x) \subseteq^{\mathcal{T}}_{\mathsf{er}} S_{\mathsf{T}'}(x) \quad \Gamma \vdash \mathtt{l} : \mathtt{List<T'>} \quad \Gamma \vdash \mathtt{e}_i : \mathsf{T}_i}{\Gamma \ \vdash^{\mathcal{T}}_{\mathsf{er}}\ \mathtt{l:=access}(\exists \bar{y}.\, \phi(x), \mathtt{e}_1, \ldots, \mathtt{e}_n)\ :\ \mathtt{Unit}}$$

Typing Semantically Lifted State Access using Concept Subsumption

There are no practical algorithms for query containment under expressive entailment regimes. We, thus, resort to an approximation that reduces the check to concept subsumption with respect to a TBox. If the query can be over-approximated by some DL concept C, then it suffices to check whether C subsumed by the concept $Class_{\mathsf{T}'}$ to ensure that the query return something of type T'.

In the, again slightly simplified, rule below, the first premise is changed to express this situation. For tree-shaped conjunctive queries, such a concept C can be easily and efficiently computed directly from the query.

$$\textbf{(acc-approx-type)}\ \frac{\exists C.\ \exists \bar{y}.\phi(x) \sqsubseteq^{\mathcal{T}} C \sqsubseteq^{\mathcal{T}} Class_{\mathsf{T}'} \quad \Gamma \vdash \mathtt{l} : \mathtt{List<T'>} \quad \Gamma \vdash \mathtt{e}_i : \mathsf{T}_i}{\Gamma \ \vdash^{\mathcal{T}}_{\mathsf{er}}\ \mathtt{l:=access}(\exists \bar{y}.\, \phi(x), \mathtt{e}_1, \ldots, \mathtt{e}_n)\ :\ \mathtt{Unit}}$$

Conclusion

We are able to give a type system to semantically lifted programs and implemented it for the **Semantic Micro Object Language** (SMOL) [1]. This is a first approach to directly couple programming languages and DL reasoning with **static** guarantees about the **interactions** while still maintaining a **separation of concerns** between programming and domain modelling.

[1] Kamburjan et al. *Programming and Debugging with Semantically Lifted States*, ESWC'21

Further Research

Query Containment Type checking requires only **asymmetrical** query containment: the right-hand side is simple concept query. Can this asymmetry be used for more **practical** containment checks under entailment regimes?

Concept Approximation Can we approximate **more complex** classes of queries to approximate type checking?

Download and Contact

SLPs are implemented in the SMOL interpreter, try it out under:

github.com/Edkamb/SemanticObjects

■ Eduard Kamburjan eduard@ifi.uio.no
■ Egor V. Kostylev egork@ifi.uio.no

UiO : University of Oslo