Turtoise: Interfacing hierarchical data with knowledge graph

Position of the problem: Hierarchical and distributed representation of symbolic data

Many data sets are defined using hierarchical data structures, what is called a “record” or “named tuple”, often collected in “spreadsheet” or “table”. The JavaScript Object Notation (JSON) for instance, under the vocable of “object”, defines the notion of record, i.e., collection of unordered name and value pairs, each value being either a Boolean, string or numeric litteral or a sub-structure. Here we use the wJSON semantic variant for which, for instance, a list is one to one correspondence with a record with numbers indexing the list values. Furthermore, each data is specified with a type. Such representation is universal in the sense that we are used to define almost all data (e.g., everyday structured data, software parameters and other configuration data, digital object metadata, etc) in such a format.

On the other hand, general purpose language for representing semantic information to define ontology, linked data and metadata, such as semantic web contents [Hoekstra, 2009] are highly distributed. The Resource Description Framework (RDF) decomposes the knowledge in atoms of the form `(subject predicate object)`, an object being either a data or another subject. For an individual, a data property correspondis to qualitative or quantitative feature, while an object property (i.e. with another subject as object) corresponds to a relation. This corresponding to a graph data structure with predicate as labeled edge and subject or object as labeled node, i.e., “linked data”. Data modeling is implemented via properties stated on the individual, specified via predicates, such as the RDF schema[1] or OWL2[2], which formally defines meaning over the defined facts, while we can also use derivation rules (e.g., using SWRL[3] rules). See, e.g. [Mercier et al., 2021] for an introduction in this context.

Representing data in such a distributed manner corresponds to A-box[4] of a knowledge graph. Data modeling using such ontology predicates or derivation rules define a T-box[5] which will generates *deduced* qualities, offering the possibility to perform inferences, thus implementing dynamic features at a pure symbolic level.

We thus need to map hierachical data structure onto a A-box and reintroduce the deduced results into the original data structure to enrich it.

[1]`https://www.w3.org/TR/rdf-schema`

[2]`https://www.w3.org/TR/owl2-overview` with https://www.w3.org/TR/owl2-primer for an introduction.

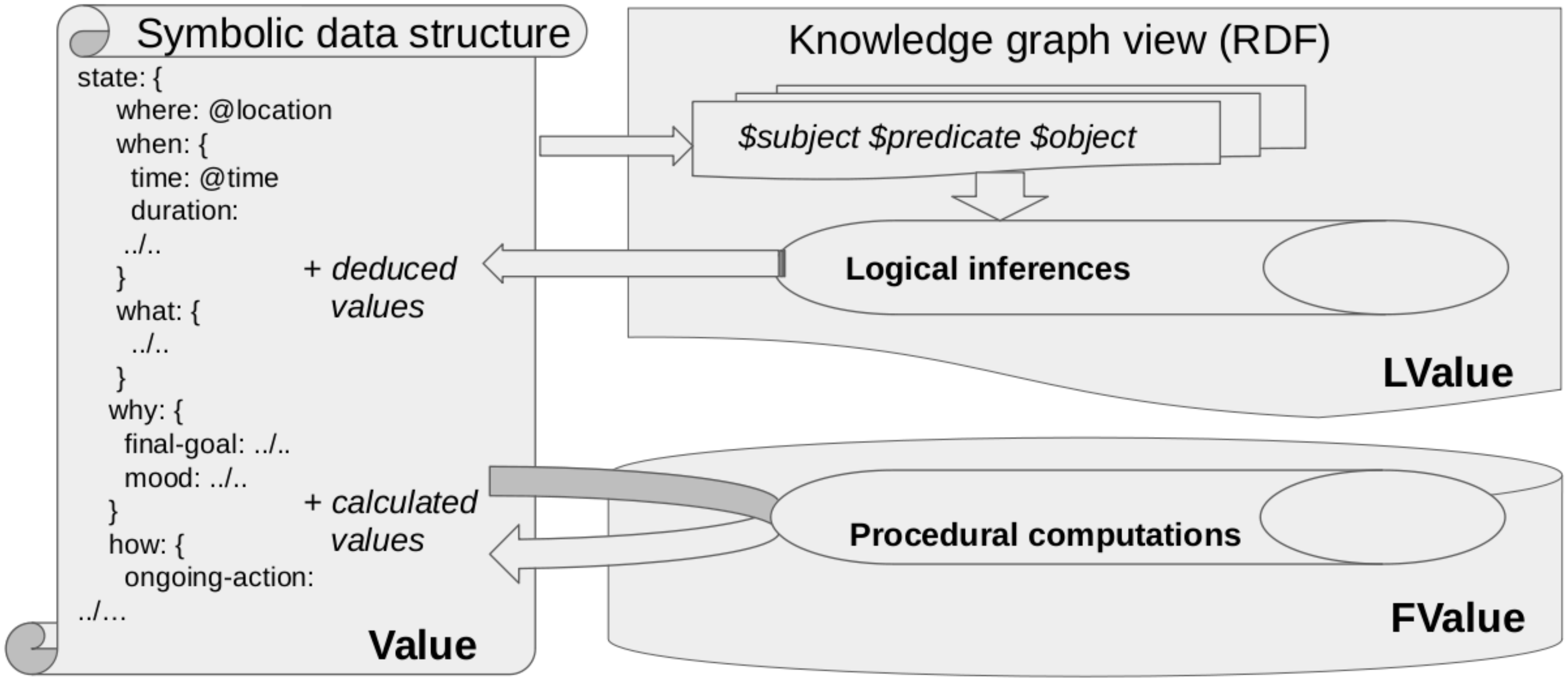
[3]`https://www.w3.org/Submission/SWRL/`

[4]https://en.wikipedia.org/wiki/Abox

[5]https://en.wikipedia.org/wiki/Tbox

Position of the problem: Calculated and deduced value

At the implementation level, More precisely a static data structure implemented as `Value` can also have dynamical features computed from other values, implemented as `FValue`, and features deduced using a reasoner, implemented as `LValue`, as represented here:



A one to one correspondence between both representations

Each hierarchical data structure is translated[6] in terms of RDF statements as follows: Each record item is a ”subject” and each named value corresponds to a ”property”, the value being the ”object” targeted by the relationship.

For instance “someone who has the name Alice, knows someone else, who has the name Bob, who knows someone else who has the name Eve, while its email is bob@example.com” writes, using wJSON on one hand and Turtle syntax on the other hand:

Turtoise implicit node syntax

```
{
  @base: https://gitlab.inria.fr/line/aide-group/wjson/-/raw/master/src/test.nt
  @prefix: {
    foaf: <http://xmlns.com/foaf/0.1/>
  }
  foaf:name: Alice
  foaf:knows: {
    foaf:name: Bob
    foaf:knows: {
      foaf:name: Eve
    }
    foaf:mbox: bob@example.com
  }
}
```

Turtle blank node syntax

```
@base https://gitlab.inria.fr/line/aide-group/wjson/-/raw/mast
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
[ foaf:name "Alice" ]
  foaf:knows [
    foaf:name "Bob" ;
    foaf:knows [
        foaf:name "Eve" ] ;
    foaf:mbox "bob@example.com" ]
```

which (avoing taking the `@base` for the sake of clarity) expands in N-Triples syntax to

```
<local:@prefix>                <local:foaf>                       <http://xmlns.com/foaf/0.1> .
<local:>                       <http://xmlns.com/foaf/0.1/name>   "Alice" .
<local:foaf:knows>             <http://xmlns.com/foaf/0.1/name>   "Bob" .
<local:foaf:knows/foaf:knows>  <http://xmlns.com/foaf/0.1/name>   "Eve" .
<local:foaf:knows>             <http://xmlns.com/foaf/0.1/mbox>   "bob@example.com" .
```

The “Turtoise” syntax corresponds to defining a data structure using wJSON syntax with optional `@base` and `@prefix` directives.

[6]The position is very different from JavaScript Object Notation for Linked Data (JSON-LD) which is a method of encoding linked data specify to limit the work of transforming *any* existing JSON data structures to RDF linked data. Here only rather specific data structures are considered.

What corresponds to blank node[7] is serialized using `local: IRI` defining the relative location in the structure.

The `@prefix` directive is understood and implemented as specified[8] in the Turtle specification, e.g., `foaf:name` is expanded to the corresponding absolute `IRI`. The `@prefix` directive itself is also serialized in order to be able to reconstruct the original data structure. We thus have a one to one correspondence between the N-Triples serialization and the original data structure.

The `@base` directive is understood and implemented as specified in the Turtle specification. If the `@base` is defined, the `local:` prefix is expanded to the `@base IRI`. The `@base` directive is global to a whole data structure, must be unique and defined at the top level, in accordance with Turtle specification.

In compliance with the wJSON semantic, insertion order is made explicit (e.g., the node of name `Bob` is referenced as someone known by the node of name `Alice`). This will not changed the reasoning on the data structure contents, but allows to keep trace of both the structure and the insertion order, as in human spoken language with is intrinsically sequential.

Comparison with Turtle specification

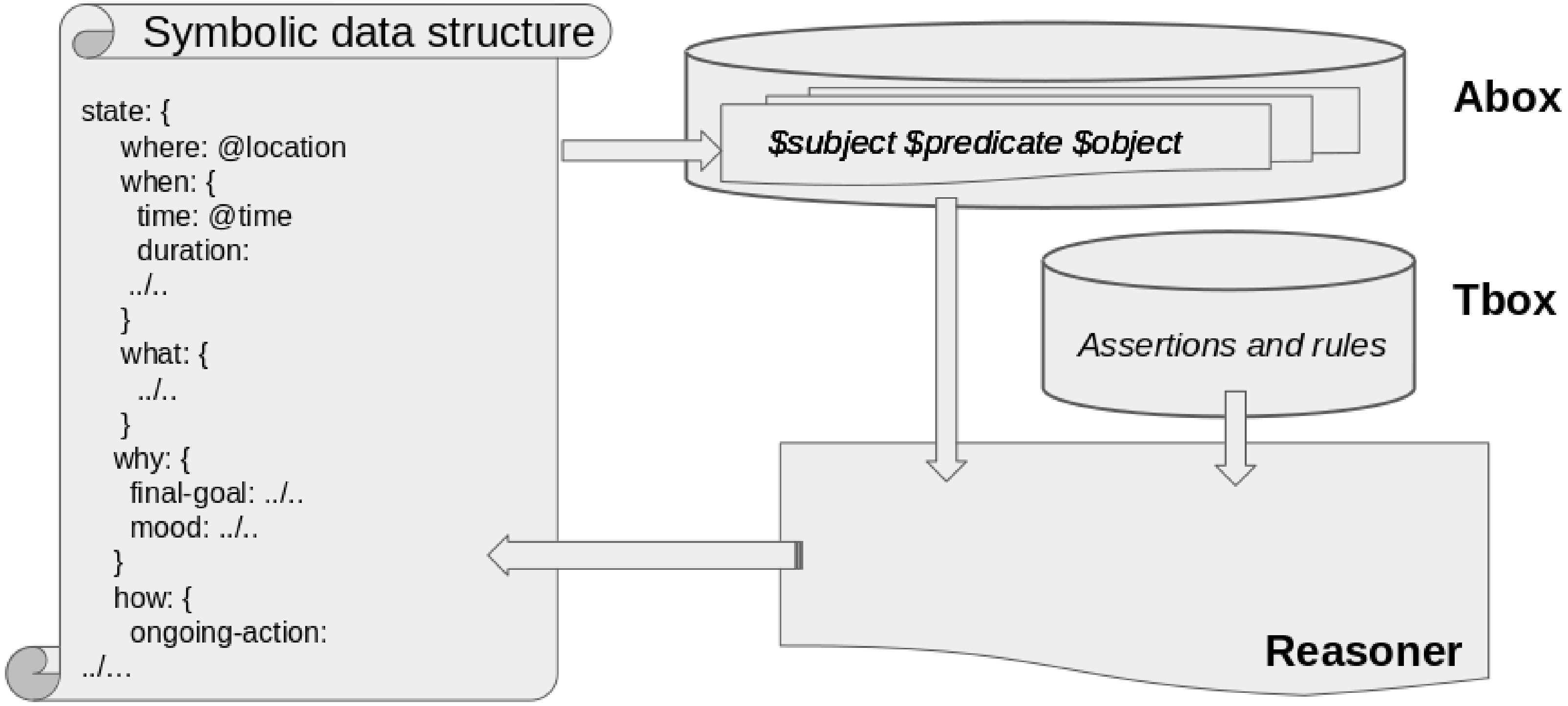
The idea is that "Turtoise" is a dialect of the Turtle semantic using the wJSON syntax with the following characteristics:
- The `IRI` are defined as for Turtle IRI using `@base` and `@prefix` directives.
- For this preliminary version, literal are only string without the capability to define Turtle literals: Boolean and numeric are parsed from their string representation in compliance with Value semantic, while language tag and region sub-tag can not be used. There is no obstacle to extend the specification including these features, through not very useful in the present application context.
- The predicate lists, object lists and blank nodes correspond to the structure of the JSON language using the wJSON syntax: A record corresponds to predicate list, an object list can be defined using a JSON array, while the hierarchical structure induces blank nodes and, up to our best understanding all common usage of blank nodes corresponds to such hierarchical definition.
- The Turtle collections correspond to JSON array but with a semantic difference: In wJSON, the underlying structure corresponds to RDF containers (represented as a numerical indexed vector of value) and not as in Turtle to a RDF collection (represented as a chained list of values). This seems preferable at the application level, through it would be straightforward to serialized a JSON array as a chained list if required.

[7]We do not use blank node labeling of the form `_:`*`URIref`* because it is interesting to encode the relative location in the hierarchical data structure in the blank node identifier which is not compatible with the `URIref` lexical constraints.

[8]In fact, in Turtoise, the `@prefix` directive visibility, i.e., scope includes all record items and sub-items following its declaration, but not parent data structure; several `@prefix` directives can thus be defined, with different scopes. This is not compliant with the Turtle specification and such usage is not recommended.

Interface with an external reasoner

At the implementation level, the `LValue` interface is performed using an external software as schematized here:



The T-box inference rules are not defined using the Turtoise syntax but using the RDFs, OWL2 languages or using SWRL rules since there is no need to represent such information as a hierarchical data structure.

References

[Hoekstra, 2009] Hoekstra, R. (2009). Ontology Representation - Design Patterns and Ontologies that Make Sense. *Frontiers in Artificial Intelligence and Applications*. IOS Press is an international science, technical and medical publisher of high-quality books for academics, scientists, and professionals in all fields. Some of the areas we publish in: -Biomedicine -Oncology -Artificial intelligence -Databases and information systems -Maritime engineering -Nanotechnology -Geoengineering -All aspects of physics -E-governance -E-commerce -The knowledge economy -Urban studies -Arms control -Understanding and responding to terrorism -Medical informatics -Computer Sciences.

[Mercier et al., 2021] Mercier, C., Roux, L., Romero, M., Alexandre, F., and Viéville, T. (2021). Formalizing Problem Solving in Computational Thinking : an Ontology approach. In *2021 IEEE International Conference on Development and Learning (ICDL)*, pages 1–8, Beijing, China.