Appendix: Supplementary Materials Algorithm, Implementation and Alist Examples

A Algorithm for converting an alist query to a SPARQL query.

Algorithm 1: Alist to SPARQL

```
Input  : A simple Alist, $\mathcal{A}$
Output: A SPARQL query
Let $\mathcal{A}$ be a simple alist, $U$ be a set of ⟨concept, URI⟩ pairs,
GetIdentifier($x$) is a helper function that returns the URI
  of item $x$ in a given knowledge base,
Attributes($\mathcal{A}$) returns the list of attributes of $\mathcal{A}$, and
$\mathcal{V}$ is the set of variables in $\mathcal{A}$.
 // The notation $S[y]$ is used to access the
    value of attribute $y$ in an alist or a
    dictionary $S$.
for $a \in [s, p, o]$ do
    if $\mathcal{A}[a] \in \mathcal{V}$ then
        $U[a] \leftarrow$ GetIdentifier($\mathcal{A}[a]$)
    else
        $U[a] \leftarrow \mathcal{A}[a]$
$query \leftarrow$ "SELECT ⟨$\mathcal{A}[v]$⟩ WHERE {"
 // $v$ is the operation variable of $\mathcal{A}$
$query \leftarrow query +$ "⟨$U[s]$⟩ ⟨$U[p]$⟩ ⟨$U[o]$⟩. "
 // Terms in ⟨·⟩ are replaced by their values
 // Add time attribute as to the query
if $t \in$ Attributes($\mathcal{A}$) then
    $t_{property} \leftarrow type(\mathcal{A}[t])$
     // Default time property type is 'year'
    $t_{id} \leftarrow$ GetIdentifier($t_{property}$)
    $query \leftarrow query +$ "⟨$U[s]$⟩ ⟨$t_{id}$⟩ ⟨$\mathcal{A}[t]$⟩."
 // Add location attribute as to the query
if $l \in$ Attributes($\mathcal{A}$) then
    $l_{id} \leftarrow$ GetIdentifier("location")
    $query \leftarrow query +$ "⟨$U[s]$⟩ ⟨$l_{id}$⟩ ⟨$\mathcal{A}[l]$⟩."
$query \leftarrow query$"}"
Return query
```

Alist are first normalised to simple alists (without nesting) before conversion to other query languages. In the algorithm below, we make the following assumptions:

1. We assume that the alist contains at least one projection variable and has the default VALUE operation (i.e. an identity function on its operation variable).
2. We assume that we have a helper function `GetIdentifier` that retrieves the unique identifier (e.g. the unique resource identifier(URI)) of a concept in an given knowledge graph.
3. We assume that the variable in the alist is assigned to the subject or object property.

The alist attribute symbols used in the algorithm are the same as those shown in Table 1 of the paper and repeated below in Table 1 for convenience.

Python code files for an implementation of this algorithm to access the Wikidata knowledge graph, and variations of the algorithm to that work the World Bank and MusicBrainz datasets have also been added as supplementary materials.

Category	Name	Symbol
Object-Level	subject	s
	property	p
	object	o
	time	t
	location	l
Meta-Level	explanation	x
	context	c
	uncertainty	u
	source	d
Functional	operation	h
	operation_variable	v

Table 1. List of attributes, their categories and shorthand symbols

B Alist Examples

We now demonstrate how the different types of questions in the LC-QuAD dataset[1] can be represented in an alist. We use double quotes around multi-word values of alists for clarity, and omit them otherwise to avoid clutter.

1. **Single Fact**
   Who is the screenwriter of Mr. Bean?

```
{h:value, v:?x, s:"Mr. Bean", p:screenwriter, o:
    ?x}
```

2. **Single Fact with Type** Billie Jean was on the track list of which album?

```
{h:value, v:?x, s:"Billie Jean", p:partOf, o:?x,
  ?x: {h:value, v:?y, s:?y, p:type, o:album}}
```

3. **Multi-Fact** What is the name of the sister city tied to Kansas City, which is located in the county of Seville Province?

```
{h:value, v:?x, s:?x, p:"sister city", o: "
    Kansas City",
  ?x: {s:?y, p:location, o:"Seville Province"}}
```

   Or

```
{h:value, v:?x, s:"Kansas City", p:"sister city"
    , o:?x,
  ?x: {s:?y, p:location, o:"Seville Province"}}
```

4. **Facts with Qualifiers**
   What is the venue of Barak Obama's marriage?

```
{h:value, v:?y, s:$x, p:venue, o:?y,
  $x: {s:"Barak Obama", p:marriage, o:?z}}
```

5. **Two Intentions** Who is the wife of Barak Obama and where did he get married?

```
{h:list, v:[$x,$y],
  $x: {s: "Barak Obama", p:wife, o:?x},
  $y: {s: ?y, p:location, o:$z,
    $z: {s: "Barak Obama", p:marriage, o:$z}} }
```

   Note: When an alist has no projection variable in its scope, the result of applying h to its operation variable is returned.

6. **Boolean** Did Breaking Bad have 5 seasons?

```
{h:equal, v:[$x,$y],
  $x: {h:count, v:?x, s:"Breaking Bad", p:
      seasons, o:?x},
  $y: 5}
```

7. **Count** What is the number of siblings of Edward III of England?

```
{h:count, v:?x, s:"Edward III of England", p:
    siblings, o:?x}
```

8. **Ranking** What is the binary star which has the highest colour index?

```
{h:rank, v:[$x,1], s:?y, p:"color index", o:$x,
  ?y: {h:value, v:?z, s:?z, p:type, o:"binary
      star"}}
```

   Note: *rank* is a generalization of the *max* and *min* functions such that it sorts the list of values (with respect to the variable provided) and returns the indexed value: $max(x) = rank(x, 1)$ and $min(x) = rank(x, -1)$. This generalisation allows for other ranks to be returned; e.g. *second highest* $= rank(x, 2)$, *third lowest* $= rank(x, -3)$, etc.

9. **String Operation**
   Give me all the rock bands that start with letter R.

```
{h:startswith, v:[?x,"v"], s:?x, p:type, o:"rock
    band"}
```

10. **Temporal Aspect**
    With whom did Barak Obama get married in 1992?

```
{h:value, v:?x, s:"Barak Obama", p:married,
 o:?x, t:1992}
```

[1] Dubey, M., Banerjee, D., Abdelkawi, A., & Lehmann, J. (2019). Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II 18 (pp. 69-78). Springer International Publishing.