Question Name

intro text for the question

1 Ideal Statements

paragraph or list describing the ideal input statements

1.1 statement parameters to utilize

- first param
- second param
- third param

2 TLA Statement problems

paragraph talking about known data issues within current TLA implementation

3 Algorithm

3.1 Summary

1. step 1
2. step 2
3. step 3

3.2 Query an LRS via REST

How to query an LRS via a GET request to the Statements Resource [1]

```
Agent = "agent={"account":
                {"homePage": "https://example.homepage",
                "name": 123456}}"

Since = "since=2018-07-20T12:08:47Z"

Until = "until=2018-07-21T12:08:47Z"

Base = "https://example.endpoint/statements?"
```

---

[1] S is the set of all statements parsed from the statements array within the HTTP response to the Curl request. It may be possible that multiple Curl requests are needed to retrieve all query results. If multiple requests are necessary, S is the result of concatenating the result of each request into a single set

```
endpoint = Base + Agent + "&" + Since + "&" + Until

Auth = Hash generated from basic auth

S = curl -X GET -H "Authorization: Auth"
          -H "Content-Type: application/json"
          -H "X-Experience-API-Version: 1.0.3"
          Endpoint
```

- Update as needed to reflect the needs of an LRS query

3.3   Z Specifications

Outline of Z, includes templates and an example of a system used to check staff members in and out of a building

3.3.1   xAPI Statement(s) Schema

[Statement] [Actor] [Verb] [Object] [Result] [Context] [Timestamp]

$$
\begin{array}{|l}
\hline
\quad Statement \\
\hline
s : Statement \\
\hline
s = \{Actor, Verb, Object, Timestamp\}\,| \\
\quad \{Actor, Verb, Object, Timestamp, Context\}\,| \\
\quad \{Actor, Verb, Object, Timestamp, Result\}\,| \\
\quad \{Actor, Verb, Object, Timestamp, Result, Context\} \\
\hline
\end{array}
$$

- The variable s is of type Statement and consists of an Actor, Verb, Object, Timestamp and optionally Context and Result

$$
\begin{array}{|l}
\hline
\quad Statements \\
\hline
S : Statements \\
\hline
S = \{\, s : Statement \,|\, S \neg \emptyset \} \\
\hline
\end{array}
$$

- The variable S is of type Statements and is a set of objects s, each of type Statement
- The variable S is a non empty set

3.3.2   Introduce Basic Types

Template   [Name of variable(s) of type set]

Example   [X]

3.3.3 Example Schema

Basic unit of specification, defines state variables, system state, operations, etc.

Template

$$\begin{array}{|l}
\hline
\quad SchemaName \underline{} \\
\quad VariableDeclarations \\
\hline
\quad Predicate/Invariants \\
\hline
\end{array}$$

Example

$$\begin{array}{|l}
\hline
\quad Counter \underline{} \\
\quad ctx : \mathbb{N} \\
\hline
\quad 0 \le ctr \le max \\
\hline
\end{array}$$

Variables

$$\begin{array}{|l}
\hline
\quad Counter \underline{} \\
\quad ctx : \mathbb{N} \\
\hline
\\
\hline
\end{array}$$

- the variable ctx is a natural number

Predicates

$$\begin{array}{|l}
\hline
\quad Counter \underline{} \\
\hline
\quad 0 \le ctr \le max \\
\hline
\end{array}$$

- ctr is greater than or equal to 0

- ctr is less than or equal to max

3.3.4 Initialisation

The starting conditions

Template

$$\begin{array}{|l}
\hline
\quad Init[VarName] \underline{} \\
\quad NameOfExistingSchema \\
\hline
\quad InitStateOfVarsWithinRefSchema \\
\hline
\end{array}$$

Example

```
┌─ InitCounter ─────────────────────────────────
│ Counter
│ ────────────────
│ ctr = 0
└───────────────────────────────────────────────
```

- the value of the counter starts at 0

3.3.5   Operations

an operation is specified in Z with a predicate relating the state before and after the invocation of that operation

Template

```
┌─ OperationName ───────────────────────────────
│ ΔSchemaName
│ inputParam? : SomeType
│ outputParam! : SomeType
│ ──────────────────────────────
│ InvariantPredicate
│ NewValForVar′ = OperationOnInput/OutputParams
└───────────────────────────────────────────────
```

Example

```
┌─ Increment ───────────────────────────────────
│ ΔCounter
│ ────────────────
│ ctr < max
│ ctr′ = crt + 1
└───────────────────────────────────────────────
```

- There is an implicit conjunction (logical-and) between successive lines of the predicate

```
┌─ Decrement ───────────────────────────────────
│ ΔCounter
│ d? : ℕ
│ ────────────────
│ ctr ≥ d?
│ ctr′ = ctr − d?
└───────────────────────────────────────────────
```

- input params suffixed with ?

$$
\begin{array}{|l}
\underline{\quad Display \quad} \\
\Xi Counter \\
c! : \mathbb{N} \\
\hline
c! = ctr
\end{array}
$$

- output params suffixed with !

- the greek symbol means that the operation cannot change the state of Counter

3.4 Pseudocode

Algorithm 1: How to write algorithms

Input: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
> read current;
> **if** *understand* **then**
>> go to next section;
>> current section becomes this one;
>
> **else**
>> go back to the beginning of current section;
>
> **end**

end

3.5 Result JSON Schema

3.6 Visualization Description

description of the associated visualization in english

3.7 VEGA example

This section will be updated to include a VEGA JSON blob for prototype viz