Dash to TLA+ Translator

Introduction:

- This is a high-level description of the Dash to TLA+ translator
- This serves as a feature checklist for the developer of the translator
- This documents design decisions, notes and potential roadblocks involved

Feature list

Alloy

The Alloy features require translation beginning with a core group of most-used language constructs, with progressive development of an expanding fringe of lesser-used constructs.

- signatures
- sub-signatures
- multiplicities - set, one, lone, some
- fields
- arithmetic operations
- set operations
- quantification
- relations
- signature hierarchies
- functions
- predicates
- sum and cardinality

Signatures and Sub-signatures are to be implemented as sets, using set operators to describe the relations between the sub-signatures and their parent signatures. Thus, set operations (which have direct analogues in TLA+) and signatures are part of the core group.

Since guards and actions are boolean expressions, the next translation target involve everything necessary to generate a simple valid guard.

In the example of the musical chairs model:

```
default state Start {
        trans Walk {
            on MusicStarts
            when #activePlayers > 1
            goto Walking
            do occupiedChairs' = none -> none
        }
```

Constructing simple expressions requires the use of arithmetic operators and the cardinality operator. This, along with the prime notation, have direct analogues in TLA+, making it part of the core group.

Translating quantification is closely tied to multiplicities, which are both bundled together into the next group.

- run commands
- check commands

Implementing these require careful consideration of the config file and TLC's mode of operation, and the translation requires that an API combining the config and the module is implemented first (i.e. ensuring the config is always valid for a given module and vice versa)

Dash

- states - basic, AND, OR
- transitions
- events - environmental and internal (modelled using big-step/small-step semantics)

The above are to be ported directly from the earlier codebase. The below depend on having a working translation of Alloy first.

- guards
- actions

Dash+

- parameterized states

LTL properties

- temporal operators

Roadmap:

- Implement TLA+ AST elements
- Implement TLA+ AST module API functions
- Port earlier translator from the CUP-grammar variant
- Extend translator to read Alloy expressions for guards and conditions and place them in the appropriate location in the model
- Translate Alloy (note: this description lacks sufficient granularity)

Notes:

- The implementation of the TLA+ AST and Module is expected to ensure that Modules with obvious semantic errors are impossible to build, by implementing a layer of abstraction to manipulate the AST.
- Full feature coverage is not expected, features are implemented as needed.
- The `sum` function in Alloy has no clear translation in TLA+. TLA+ has ways to filter and map sets and records, but no way to reduce them. `sum` is a reduction on a list.

- Symmetry-breaking is carried out by the Alloy Analyzer. Since everything is a set in Alloy, these could be represented as lists in the translation without any loss of detail. However, it is likely that there are optimizations involved in handling sets that are implemented behind the scenes by TLC, which makes it reasonable to translate sets to sets and not lists.