EVBT — an Event-B tool for code generation and documentation

Fredrik Öhrström
May 13, 2021
`fredrik.ohrstrom@viklauverk.com`

Evbt is a command line tool for generating code and documentation from Event-B models created using the Rodin toolset. It contains its own independent implementation of an Event-B formula parser and a typing system. The tool expects already proven Event-B models as input. Even though the tool is work in progress, it can already generate documentation and code for several existing models, for example the model for traffic lights controlling access to a bridge.[1]

To generate documentation for a Rodin workspace, you run:
`evbt docgen tex workspace/BridgeTrafficLights` to create `BridgeTrafficLights.tex`.

To generate code, you run: `evbt codegen c++ workspace/BridgeTrafficLights/m3.bum` to create a `BridgeTrafficLights.h` and a `.cc` file. For this model you also need to add `#define d 17` to the cc file since that constant is undefined in the model. You also need to add suitable set/get events to read and modify the state from the outside.

The evbt code generation creates a state machine hidden behind an API in a header file. The API will consist of events which in the final refinement still have parameters. Such events are translated into API functions with parameters mirroring the event parameters.

`EVENT` addLoan
Loan a book to a borrower, the book must not be on loan already.

`ANY`

$borr$
$book$

`WHERE`

`grd1:`	$borr \in borrowers$	Valid borrower.
`grd2:`	$book \in books$	Valid book.
`grd3:`	$book \mapsto borr \notin loans$	Not a necessary test, but used for this example anyway.
`grd4:`	$book \notin \mathrm{dom}(loans)$	The book is not loaned out already.

`THEN`

`act1:`	$loans(book) := borr$	Add a new loan in the storage.

`END`

```
bool LibraryImplementation::addLoan(uint64_t borr,uint64_t book)
{
    // Valid borrower.
    bool grd1 = borrowers.count(borr);
    if (!grd1) return false;
    // Valid book.
    bool grd2 = books.count(book);
    if (!grd2) return false;
    // Not a necessary test, but used for this example anyway.
    bool grd3 = loans.count(book)==0 || loans[book] != borr;
    if (!grd3) return false;
    // The book is not loaned out already.
    bool grd4 = loans.count(book)==0;
    if (!grd4) return false;
    // Add a new loan in the storage.
    loans[book] = borr;
    traceEvent("addLoan");
    return true;
}
```

[1] Jean-Raymond Abrial (2010). *Modeling in Event-B: System and Software Engineering.* Cambridge University Press.

Values are returned from the API function through parameters prefixed with “out_”.

`EVENT` whoBorrowsBook
Return who is borrowing a book.

`ANY`

book
out_borrower

`WHERE`

grd1:	$book \in books$	Querying a valid book?
grd2:	$book \in \mathrm{dom}(loans)$	That is on loan?
grd3:	$out\_borrower = loans(book)$	Return the result through out.

`END`

```
bool LibraryImplementation::whoBorrowsBook(uint64_t book,uint64_t *out_borrower)
{
    // Querying a valid book?
    bool grd1 = books.count(book);
    if (!grd1) return false;
    // That is on loan?
    bool grd2 = loans.count(book);
    if (!grd2) return false;
    // Return the result through out.
    *out_borrower = loans[book];
    traceEvent("whoBorrowsBook");
    return true;
}
```

The evbt tool also provides a console for exploration of Event-B models and formulas. You start the console with: `evbt console` and you can now type for example: `add defaults` to fill the symbol table with a few default symbols for predicates and variables. You can now type `show formula "(P & x:BOOL) => Q"` and it will parse and print this formula: $(\mathtt{P} \wedge x \in \mathrm{BOOL}) \Rightarrow \mathtt{Q}$

If you start the console with a Rodin workspace as an argument:
`evbt console workspace/Library` then you can also print parts of the model with the command:
`show part "Library/events/whoBorrowsBook/guards"` or an invariant:
`show part "Library/invariants/inv3"`

You can embed evbt console commands inside a tex document:
`evbt docmod tex source.tex dest.tex workspace/Library`
You can now write `EVBT(...console command...)` in the `source.tex` file. These commands will be picked up by evbt, executed, and the result inserted into `dest.tex`. For example the event whoBorrowsBook described earlier was inserted into this document with the command `EVBT(show part "Library/events/whoBorrowsBook")`

The workshop covering the evbt tool will discuss the current features in more detail and how they are implemented in evbt as well as future directions for code generation. The evbt source can be downloaded here:
`https://github.com/viklauverk/EventBTool` and is available under the AGPLv3 license.