Interfacing with Proof Assistants for Domain Specific Programming Using EventML

Vincent Rahli

Cornell University

July 11, 2012

Credits

PRL group and others:

- Mark Bickford
- Robert Constable
- David Guaspari
- Richard Eaton
- Vincent Rahli
- Robbert van Renesse
- Nicolas Schiper
- Jason Wu

Work done as part of the CRASH project (**Correct-by-Construction Attack-Tolerant Systems**) funded by DARPA.

Problem

Problem: unverified protocols are wrong.

Goal: automatic synthesis of verified diversifiable distributed systems.

Our solution: tools that cooperate with a Logical Programming Environment (e.g., a constructive proof assistant).

Solving the problem

Example of such a tool: a high-level formal method tool which **gives precedence to the programming task**, and also allows programmers to **cooperate with a proof assistant**.

▶ **Bridge the gap between the programming and verification tasks.**

EventML: specification and programming language

- Not as low level as a programming language, not as high level as a specification language (an in-between working balance).
- A ML-like **functional** language.
- Features programming/logical constructs from the **Logic of Events**.
- To **code/specify** distributed protocols (domain specific).
- **Translates** specifications into event classes (easy to reason about).
  - **Logical aspect**
- **Synthesizes** distributed programs (in the model underlying the Logic of Events) from specifications.
  - **Computational aspect**

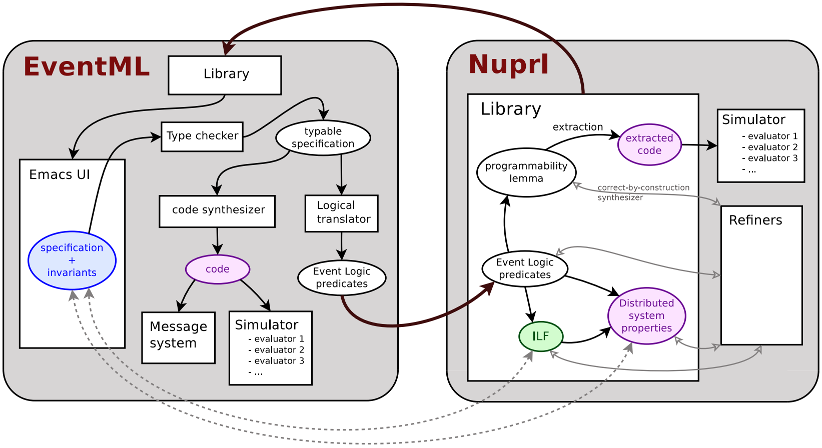
EventML cooperates with the Nuprl proof assistant

- A **constructive type theory** (CTT12 an evolution of CTT84 closely related to ITT82).
- Computational semantics: λ-terms provide evidence for the truth of propositions.
- Semantics provided by a computation system and a type system.
- A type is a collection of mathematical objects with equality defined on them.
- A hierarchy of universes to avoid Girard's paradox
- Extensional function equality.

EventML cooperates with the Nuprl proof assistant

- Extraction of computational content.
- A powerful tactic mechanism that uses ML.
- A myriad of powerful types: recursive type, dependent types, set type (refinement type), quotient type, intersection type (used to implement co-inductive types and dependent records), partial types, base type... (which go beyond CTT84, CTT04, CIC04).

Cooperation with a Logical Programming Environment



Inductive logical forms

Inductive logical forms are formulas that completely characterize the information flow of distributed systems.

They provide declarative reasons that events happen.

They are human readable.

They are useful for finding bugs.

They provide a powerful tool to prove properties of distributed systems.

Inductive logical forms

An example:

```
(∀[bnum:BNum]. ∀[accpts:bag(Id)]. ∀[Op,Cid:{T:Type| valueall-type(T)} ].
 ∀[eq_Cid:EqDecider(Cid)].∀[es:EO']. ∀[e:E]. ∀[i:Id]. ∀[m:Message].
   {<i, m> ∈ paxos_scout_output(Cid;Op;accpts) bnum@Loc o (Loc,
            paxos_p1b'base(Cid;Op), paxos_ScoutState(Cid;Op;accpts;eq_Cid) bnum)(e)
   ⟺ ↓(header(e) = ‘‘paxos p1b‘‘)
        ∧ (type(info(e)) = (Id × BNum × ((BNum × ℤ × Id × Cid × Op) List)))
        ∧ (i = loc(e))
        ∧ (((bnum = (fst(snd(body(info(e))))))
          ∧ (bag-size(fst(State of Scout bnum at e)) < paxos_threshold(accpts))
          ∧ (m = make-Msg(‘‘paxos adopted‘‘;
                          BNum × ((BNum × ℤ × Id × Cid × Op) List);
                          <bnum , snd(State of Scout (for bnum) at e)>)))
          ∨ ((¬(bnum = (fst(snd(body(info(e)))))))
            ∧ (m = make-Msg(‘‘paxos preempted‘‘;BNum;fst(snd(body(info(e)))))))}))
```

Accomplishments

We have **specified, synthesized, and proved the correctness** of several distributed protocols, such as:

- Leader election in a ring.
- 2/3 consensus protocol (safety properties–2 days worth of work).
- **Paxos** (safety properties–couple of weeks worth of work).

Why was that possible? Thanks to the powerful theory (a logic of events) developed by Mark Bickford and Robert Constable in Nuprl over the past few years, and thanks to a collaboration with the system group at Cornell (especially Robbert van Renesse).

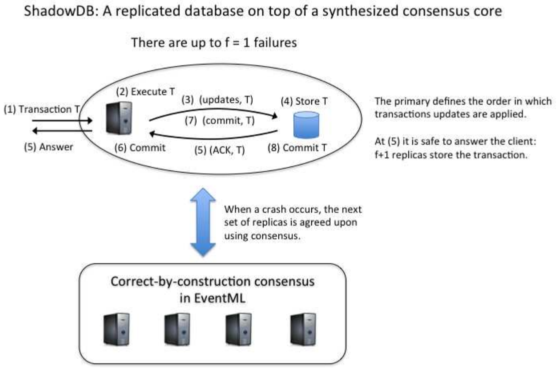
First verified version of a synthesized and deployed Paxos!?

The methodology works!

Accomplishments

The methodology works and **it is used!**

Nicolas Schiper (Cornell postdoc) has implemented ShadowDB, a replicated database on top of our synthesized 2/3 and Paxos consensus protocols.


ShadowDB: A replicated database on top of a synthesized consensus core
There are up to f = 1 failures
(1) Transaction T
(5) Answer
(2) Execute T
(6) Commit
(3) (updates, T)
(7) (commit, T)
(5) (ACK, T)
(4) Store T
(8) Commit T
The primary defines the order in which transactions updates are applied.
At (5) it is safe to answer the client: f+1 replicas store the transaction.
When a crash occurs, the next set of replicas is agreed upon using consensus.
Correct-by-construction consensus in EventML

Demo

Paxos!

What's next?

Dependent types would be useful to extend the expressivity of EventML's invariant language.

Nuprl's tactic language is ML. Could we import Nuprl tactics into EventML?