Reactive Actors in Isolation for Efficient Analysis
Reactive Actors: Isolation for Efficient Analysis

Marjan Sirjani
IDT Department
MDH
Vasteras, Sweden
marjan.sirjani@mdh.se

Ehsan Khamespanah
School of Computer Sceince
Reykjavik University)
City, Country
email address

Fatemeh Ghassemi
School of ECE
University of Tehran)
City, Country
email address

***Abstract*—I will introduce timed actors for modeling distributed systems and will explain our theories, techniques and tools for model checking and performance evaluation of such models. Timed Rebeca can be used to model asynchronous event-based components in systems, and real time constraints can be captured in the language. I will explain how floating-time transition system can be used for model checking of such models when we are interested in event-based properties, and how it helps in state space reduction. I will show different applications of our approach including analysing a wireless sensor network application, mobile ad-hoc network protocols, network-on-chip designs, and a macroscopic agent-based simulation of urban planning.**

***Index Terms*—Actors, Real-time systems**

I. INTROCUTION

Message of the paper:

The actor-based language, Rebeca, provides a usable and analyzable model for distributed, concurrent, event-based asynchronous systems (Cyber-Physical systems).

Floating Time Transition System is a natural event-based semantics for timed actors, giving us a significant amount of reduction in the state space, using a non-trivial novel idea.

How models shape the thought and ease the analysis

a) Reactive Systems:

b) Reactive Actors:

c) Faithful Models for Reactive Systems: From Rocco paper: a major challenge in designing languages is to devise appropriate abstractions and linguistic primitives to deal with the specificities of the domain under investigation.

II. FLOATING TIME TRANSITION SYSTEM

III. DISTRIBUTED MODEL CHECKING

IV. VERIFICATION OF MOBILE AD-HOC PROTOCOLS

In decentralized wireless networks there is no pre-existing infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks, nodes continuously send messages to each other to self-configure the network on ad hoc. Ad-hoc wireless network consists of mobile nodes that freely move, so the underlying topology is dynamic. As wireless communication depends on the locality of nodes, i.e., the underlying topology, the behavior of nodes in mobile Ad-hoc networks depends on the topology. Therefore, the correctness properties of Mobile Ad-hoc Networks is weaker in comparison with wired networks. For instance, one of the main properties of routing protocols is *loop-freedom*, i.e., no established route stored in the routing tables visits the same node more than once. However, in Ad-hoc networks, this property should hold for any mobility scenario. Another property for routing protocols is *packet delivery*: always packets can be sent from a source to a connected destination. For mobile Ad-hoc networks, the packet delivery property is considered as if there is a path from a source to a destination for enough long period, any packet sent from a source can be received by the destination [**?**]. Rebeca was extended in [**?**] to verify the topology-dependent properties of Ad-hoc protocols. Rebeca was extended by unicast and broadcast operators.

In the semantics of wRebeca, each semantic behavior is restricted to the set of topologies for which that behavior is valid. Such restrictions, expressed in terms of *network constraints* [**?**], [**?**], are used to verify the topology-dependent properties [**?**], [**?**].To characterize the timing-dependent behavior of such protocols concerning mobility scenarios, Timed Rebeca was extended orthogonally with the topology concepts of wRebeca. For instance, the mobility scenario over which the maximal response time to find a routing path can be extracted via model checking technique. This was achieved by combining the floating-time idea of Timed Rebeca with network constraints exploited in wRebeca.

V. VERIFICATION OF HYBRID REACTIVE SYSTEMS

Embedded systems consist of microprocessors which control physical behavior. In such *hybrid* systems, physical and cyber behaviors, characterized as continuous and discrete respectively, affect each other; the physical components may trigger the cyber components which change (by (de)activating) physical components in response. The new generation of embedded systems, cyber-physical systems (CPSs) composed of microprocessors controlling other software/physical systems via networks. For instance, in automotive systems, there are components like sensors, actuators, and controllers that communicate asynchronously with each other through a CAN network. The computational model of Rebeca provides a

The authors would like to acknowledge DPAC and SEADA.

```
 reactiveclass  Node(){
 statevars {
 int  sn, ip ;
 int []  dsn, rst ,hops,nhop;
 }
msgsrv  initial  ( int  i ,
boolean  starter ){
 ...  /* Initialization   code */
}
msgsrv rec\_newpkt(int  data , int  dip_)
{
if ( rst [dip_]==1)
{...  /*forward  packet */}
else  {
sn++;
rec_rreq (0, dip_,
dsn[dip_], self ,sn, self ,5) ;}
}
msgsrv rec_rreq  ( int  hops_,  int  dip_  ,  int  dsn_
      ,  int  oip_  ,  int  osn_ ,  int  sip_,  int
      maxHop)
{
boolean  gen_msg = false;
 ...  /*processing  code */
if  (gen_msg == true) {
if  (ip  == dip_) {
sn  = sn+1;
unicast (nhop[oip_],
rec_rrep (0  ,  dip_  ,  sn  ,  oip_  ,   self ))
succ:{
rst [oip_]  = 1;
}
unsucc:{
if ( rst [oip_]  == 1)
{...  /* error */}
rst [oip_]  = 2;}
}  else  {
hops_ = hops_ + 1;
if (hops_<maxHop) {
rec_rreq
(hops_,dip_,dsn_,oip_,
osn_, self ,maxHop);}
}}}
msgsrv  rec_rrep ( int  hops_  , int  dip_  , int  dsn_  ,
int  oip_  , int  sip_){
boolean  gen_msg = false;
 ...  /*processing  code */
if (gen_msg == true){
if (ip  == oip_  ){
 ...  /*forward  packet */  }
else  {
hops_= hops_+1;
unicast (nhop[oip_], rec_rrep
(hops_,dip_,dsn_,oip_, self ))
succ:{
rst [oip_]=1;
}
unsucc:{
if ( rst [oip_]  == 1)
{...}  /* error */
rst [oip_]  = 2;}
}}}
msgsrv  rec_rerr ( int  source_  ,
int  sip_ ,  int []  rip_rsn )
{...  /* error  recovery  code */}
}
main{
Node  n1(n2,n4) :(0, true ) ;
Node  n2(n1,n4) :(1, false ) ;
 ...
 constraints {
and(con(n1,n2),  con(n3,n4))
}
}
```

Fig. 1. The AODV protocol specified by wRebeca [?]

suitable level of abstraction to faithfully model such distributed asynchronously communicating systems in an intuitive way.

Timed Rebeca was extended by Hybrid Rebeca [] with physical behavior to support hybrid systems. Such an extension allows non-determinism inherent in concurrent and distributed systems, e. g., in the case of simultaneous arrival of messages (and no explicit priority-based policy to choose one over the other) to model check the possible implementations of systems. In Hybrid Rebeca, physical behaviors are encapsulated in so-called physical actors. Each physical actor, in addition to message handlers, is defined by a set of modes. Each mode defines the continuous behavior of the actor. A physical actor (which is instantiated from a physical class) must always have one active mode. By changing the active mode of a physical actor, it's possible to change the continuous behavior of the actor. The active mode can be changed upon receiving a message from either a software actor (controller) or a physical actor. The semantics of Hybrid Rebeca is defined as a hybrid automaton, for which many verification algorithms and tools are available.

We have shown that by using Hybrid Rebeca, the cost of improving and modifying models is vastly reduced compared to modeling in hybrid automata [] as the computational model of Hybrid Rebeca encapsulates many complexities. These complexities subsume high-level concepts like message passing and message buffering. Furthermore, modeling these complexities directly in hybrid automata can hugely decrease the analyzability of the models. Concluding that the abstraction resulted from choosing actors as the basic units of computation, offers more friendliness towards cyber-physical systems compared to the low-level languages like hybrid automata.

Acknowledgment

References