The Sigali Tool Box Environment

Loic Besnard Hervé Marchand
IRISA-CNRS IRISA-INRIA
Campus Univ. de Beaulieu, 35042 Rennes,France
{Loic.Besnard, Herve.Marchand}@irisa.fr

Eric Rutten
INRIA Futurs, LIFL
59655 Villeneuve d'Ascq Cedex France
Eric.Rutten@inria.fr

I. Overview of the tool

SIGALI is a tool that offers functionalities for verification of reactive systems and discrete controller synthesis. It manipulates *ILTS: Implicit Labeled Transition Systems*, an equational ans symbolic representation of automata. The techniques used consist in manipulating the system of equations modeling the system instead of the sets of solutions, thus avoiding the enumeration of the state space. Each set of states is uniquely characterized by a predicate and the operations on sets can be equivalently performed on the associated predicates. A wide variety of properties, such as invariance, reachability and attractivity can be checked or ensured. Many algorithms for computing state predicates are also available.

II. The Controller Synthesis methodology

Control theory of discrete event systems allows to use constructive methods, that ensure, a priori, required properties on the system behavior. Starting from a representation of the possible behaviors of the system (e.g. in the form of a Implicit Labeled Transition Systems) and the properties that have to be satisfied by the controlled system, the synthesis produces directly the constrained automaton, i.e., the one that presents only those behaviors that satisfy the required properties. In our framework, the system is represented by an ILTS composed of state and event variables, a transition function that updates the values of the states variables according to the values of the event variables that have been triggered. The choice of the admissible events is restricted to the ones that satisfy a constraint predicate depending on the current values of the state variables.

The control of the system is performed by restricting the controllable event values to values suitable for the control goal. This restriction is obtained by incorporating new algebraic equations into the initial system.

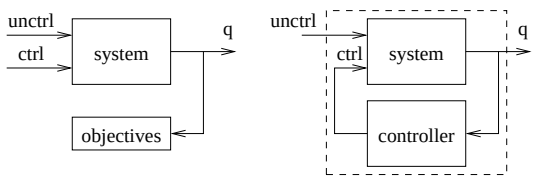

Fig. 1. Discrete control synthesis: from uncontrolled system (left) to closed-loop (right).

Using symbolic methods [6] (based on BDD techniques, avoiding state space enumeration), the various control objectives for which we are able to synthesize a controller are the following:

- "Traditional" control objectives such that: Given E and F to sets of states, one can compute controllers ensuring the *invariance* of E [1], the *reachability* of E from the initial states of the system, the *attractivity* of E from F, the *persistence*, or the *recurrence* of E.
- Control objectives expressed as partial order relations over the states of the system such that:
  - the *minimally restrictive control* (choice of the values for the controllable variables such that the system evolves, at the next instant, into a state where the maximum number of uncontrollable events is admissible)
  - the *stabilization of a system* (choice of the values for the controllable variables such that the system evolves, at the next instant, into a state with minimal change for the state variable values)
  - the *optimal control* (minimization of the cost of the trajectories between a set of initial states and a set of final states)

III. The Sigali Tool Box

Sigali is a tool box that manipulates (sets of) variables and predicates by means of (predefined) functions using a MATHEMATICA-based language. Using SIGALI, one can create predicates over a set of variables. There exists functions to manipulate them (intersection, union, complementary, test of inclusion, etc). Some functions are used to replace some variables in a predicate by other predicates. All these operations are extended to the manipulation of lists of predicates. SIGALI also manipulates cost functions. These are functions that associate to each solution of a predicate a given integer value. These cost functions can be build by associating to each variable a given cost according to the value of the variable.

Starting from the existing functions, it is also possible to define new functions, allowing, from example, to have

[1]Note that if the property is expressed by means of an observer ω, with a sink state $Error$, then it is sufficient to perform the synchronous product between the ILTS modeling the plant and the observer and to compute a controller that avoids states of the form $(q, Error)$ to be reachable in this new system.

libraries of functions dedicated to controller synthesis, optimal control or verification. Of course, one can also manipulate ILTS (e.g. to compose them according to different composition operators). Functions allowing to compute the successors (resp. predecessors, etc) belongs to the function kernel of Sigali. They can be further used to implement more intricate functions as the ones that compute the set of reachable states, the set of states that can reach a set of states by triggering uncontrollable events, etc.
The SIGALI language is described in a user manual at [12].

IV. MODELS AND SYNTHESIS TOOLS

The current implementation of the method relies on the chain of Figure 2. Centrally, we use SIGALI [5], which is a tool that performs model-checking, controller synthesis for logical goals, and optimal controller synthesis.

The model of the system can be described using a synchronous formalism. The equational language SIGNAL is the synchronous language originally connected with the synthesis tool SIGALI [5]. Another such formalism is Mode Automata [4] (associated with the MATOU tool). Once specified in either SIGNAL or MATOU, systems are automatically compiled in an ILTS than can be further analyzed by SIGALI. Finally note that the set of properties can be as well specified in one of these two languages.

When the purpose is to control the system, the result of the synthesis in SIGALI is a controller, in the form of a logical relation, which can be interpreted by a resolver module: for a given state and uncontrollable inputs, the constraints on controllable signals are solved, for example in an incremental, interactive way following the manual valuation of signals. The resolver can be coupled with the original specification of the uncontrolled system, using either the Polychrony environment, or the tool *SigalSimu* in the case of Mode Automata.

SIGALI is freely available for non-commercial use on the web page [10] (see also [12]). Note that you will need part of the Polychrony [10] or Matou [9] environment as front-end in order to specify your systems. A Signal/SIGALI manual is available at [12].

V. EXAMPLES

SIGALI has been successfully used to verify or control various academics or industrial systems (See the list below).

- Some academic examples (The cat and mouse example & the AGV example) are explained in [5].
- In [8], the synthesis methodology has been applied to the incremental design of a power transformer station controller.
- in [7], [11], we have been interested in the automatic control of systems with multiple tasks, each with multiple modes, implementing a functionality with different levels of quality (e.g., computation approximation), and cost (e.g., computation time, energy) and we made the use of SIGALI in order to control the switching of modes in order to insure properties like bounding cost while maximizing quality. The same approach was applied to the control of reconfigurations in fault-tolerant systems [3], [2]
- A domain-specific language for task managing systems has been designed in such a way that its compilation includes a pass of discrete controller synthesis. The language is partly declarative, with constraints giving synthesis objectives on the LTS, which corresponds to the more imperative part [1].

REFERENCES

[1] G. Delaval and E. Rutten. A domain-specific languagefor multi-task systems, applying discrete controller synthesis. In *21st ACM Symposium on Applied Computing, SAC 2006*, April 2006.
[2] E. Dumitrescu, A. Girault, and E. Rutten. Validating fault-tolerant behaviors of synchronous system specifications by discrete controller synthesis. In *7th Workshop on Discrete Event Systems, WODES'04*, pages 295–300, September 2004.
[3] A. Girault and E. Rutten. Discrete controller synthesis for fault-tolerant distributed systems. In *Ninth International Workshop on Formal Methods for Industrial Critical Systems, FMICS 04*, Linz, Austria, September 2004.
[4] F. Maraninchi and Y. Rémond. Mode-automata: a new domain-specific construct for the development of safe critical systems. *Science of Computer Programming*, 46(3):219–254, 2003.
[5] H. Marchand, P. Bournai, M. Le Borgne, and P. Le Guernic. Synthesis of discrete-event controllers based on the signal environment. *Discrete Event Dynamic System : Theory and Applications*, 10(4):347–368, October 2000.
[6] H. Marchand and M. Le Borgne. The supervisory control problem of discrete event systems using polynomial methods. Research Report 1271, Irisa, October 1999.
[7] H. Marchand and E. Rutten. Managing multi-mode tasks with time cost and quality levels using optimal discrete control synthesis. In *14th Euromicro Conference on Real-Time Systems*, 2002.
[8] H. Marchand and M. Samaan. Incremental design of a power transformer station controller using controller synthesis methodology. *IEEE Transaction on Software Engineering*, 26(8):729–741, 2000.
[9] Matou. www-verimag.imag.fr/people//florence.maraninchi/matou/.
[10] Polychrony. www.irisa.fr/espresso/polychrony/.
[11] E. Rutten and H. Marchand. Automatic generation of safe handlers for multi-task systems. Technical Report 5345, INRIA, October 2004.
[12] Sigali. *Tool box for controller synthesis of reactive systems*, available at http://www.irisa.fr/vertecs/softwares/sigali.html.

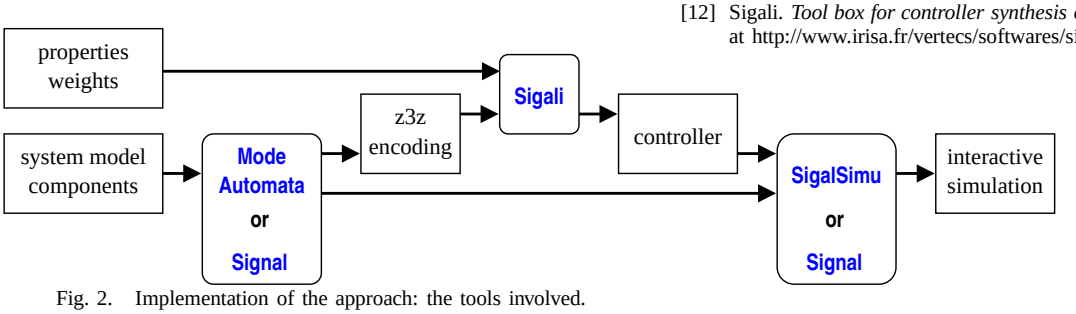

Fig. 2. Implementation of the approach: the tools involved.