# XVERSA: An Integrated Graphical and Textual Toolset for the Specification and Analysis of Resource-Bound Real-Time Systems *

Duncan Clarke, Hanêne Ben-Abdallah, Insup Lee and Hong-liang Xie
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389
{dclarke,hanene,lee,hxie}@saul.cis.upenn.edu

Oleg Sokolsky
Computer Command and Control Company
Philadelphia, PA 19104
sokolsky@cccc.com

**Abstract.** We present XVERSA, a set of tools for the specification and analysis of resource-bound real-time systems. XVERSA facilitates the use of the Algebra of Communicating Shared Resources (ACSR), a real-time process algebra with explicit notions of resources and priority. A text based user interface supports syntax checking, analysis based on equivalence checking, state space exploration, and algebraic rewriting. A graphical user interface allows systems to be described and analyzed using intuitive pictorial representations of ACSR language elements.

## 1 Introduction

There has been significant progress in the development of formal methods for the design of real-time systems in an effort to increase safety and reliability. Formal approaches to the specification and analysis of real-time systems have taken many forms, including state machines, logics, and process algebras. We focus here on tools to support the algebraic paradigm. The algebra we use is the Algebra of Communicating Shared Resources (ACSR)[LBGG94].

ACSR is a timed process algebra that facilitates the description of concurrent real-time systems with serially reusable resources. Most concurrent real-time process algebras adequately capture delays due to process synchronization, e.g., timed extensions of the classic untimed process algebras CSP and CCS[BB91, MT90, NS94]. However, these algebras abstract out resource-specific delays and priority arbitration mechanisms. In contrast, the computation model of ACSR is based on the view that the notions of resource and priority are central to real-time systems. The use of shared resources is modeled by timed actions whose executions are subject to the availability of resources. Contention for synchronization and resources is arbitrated according to the priorities of the competing actions.

To facilitate the use of ACSR in the design and analysis of real-time systems we have created GCSR [BALC95], a graphical language that captures the semantics of ACSR in an intuitive pictorial representation, and XVERSA, a toolset that automates the analysis of ACSR and GCSR system models.

The remainder of this paper is organized as follows. Section 2 introduces the ACSR and GCSR languages. Section 3 describes the XVERSA toolset. Section 4 presents some concluding remarks and information on how to obtain further information and an executable copy of the toolset.

## 2 The ACSR and GCSR Formalisms

ACSR is a timed process algebra based on the synchronization model of CCS that includes features for representing synchronization, time, temporal scopes[LG85], resource requirements, and priorities. The semantics of ACSR has been developed for both dense time and discrete time models. However, the tools presented in this paper use the discrete time model exclusively.

The semantics of an ACSR process is defined in terms of a prioritized labeled transition system. Edges are labeled with prioritized *events* of the form $(e, p)$, or *actions* that represent sets of prioritized resources to be

* This research was supported in part by NSF CCR-9415346, AFOSR F49620-95-1-0508, and ARO DAAH04-95-1-0092.

consumed for one time unit, e.g., $\{(r_1, p_1), \cdots, (r_n, p_n)\}$. Events model synchronization between concurrent process terms in a manner similar to that used in CCS. Actions represent resource allocation during the synchronous passage of one unit of discrete time. When several events and/or actions are offered simultaneously, a preemption relation determines which events or actions are allowed by pruning low priority edges.

ACSR offers two basic notions of behavior equivalence that are defined over the prioritized labeled transition system. The first equivalence relation is based on strong bisimulation, $\sim_\pi$, which insures that equivalent processes match one another's labeled transitions; it is a congruence relation. The second is based on weak bisimulation, $\approx_\pi$, which insures that equivalent processes match one another's non-$\tau$ events but allows one process to make transitions on $\tau$ that an equivalent process does not match.

A sound and complete set of approximately 30 $\sim_\pi$-preserving algebraic laws has been developed for ACSR. These laws can be used to derive proofs of properties of ACSR process expressions.

The Graphical Communicating Shared Resources (GCSR) language addresses a shortcoming of ACSR (and process algebras in general): textual, mathematical notations often produce obtuse descriptions. GCSR was developed to support the modular, hierarchical, and thus scalable, specification of real-time systems. In GCSR, the visibility scope of communication events, which reflect potential dependencies between system components, can be limited. Furthermore, GCSR's notion of hierarchy is *structured* in the sense that no edge can cross node boundaries and there is a graphical distinction between control transfer due to an interrupt versus an exception, i.e., involuntary versus voluntary release of control. These two syntactic features, in addition to the explicit representation of resources and priorities, distinguishes GCSR from other graphical languages for real-time systems, e.g., Statecharts[Har87], Modechart[JM94] and Communicating Real-time State Machines[Sha92].

The GCSR and ACSR languages are well integrated as there is a sound translation both from GCSR descriptions to ACSR processes and vice versa[BA96]. Thus, the theory of ACSR (including its semantic model, notions of equivalence, and set of algebraic laws) is directly applicable to GCSR. For instance, the algebraic laws of ACSR can be used to restructure a GCSR description to a graphically more succinct, e.g., fewer edges and nodes, yet, equivalent GCSR description. In addition, the sound integration between GCSR and ACSR makes it is possible to mix the graphical and textual notations; for example, to specify the high-level view of a system graphically and then fill the details of components textually.

## 3 The XVERSA Toolset

We have implemented a toolset with a graphical user interface to facilitate the use of ACSR and GCSR for modeling and analysis of real-time systems. Figure 1 shows the overall structure of the XVERSA system. The user's view of the tool is provided by the GCSR GUI and an X-Windows interface. The analysis of ACSR specifications is carried out by the VERSA system[CLX95b] that is accessed through these interfaces.

The user interfaces are responsible for management of input/output streams. They allow processes to be entered as graphical GCSR process descriptions or as ACSR processes using a text-based notation. Graphical input of GCSR specifications is managed with drawing support functions, syntax-checking functions, and automated translation from GCSR to ACSR. The text-based notation accepted by the X-Windows interface enhances the ACSR process algebra with the facility to define macros (e.g., to define manifest constants) and indexing which can be used to emulate value-passing.

Within VERSA there are four major functional areas for analyzing processes: term rewriting, state space exploration, equivalence testing, and interactive execution.

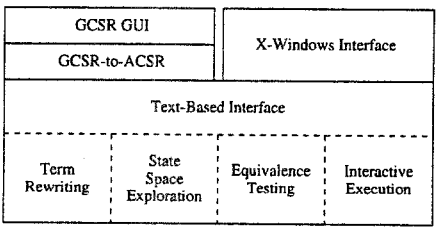


**Fig. 1.** The XVERSA Toolset

The rewrite system facilitates the rewriting of ACSR process expressions according to sound algebraic laws that preserve prioritized strong equivalence, a bisimulation relation that respects priority. At the direction of the user, the rewrite system applies pre-defined algebraic laws to one or more processes, producing a new process that may be bound to a new, or pre-existing process variable. In this way, algebraic proofs of the equivalence of process expressions may be developed. The tool aids this process by automatically determining and applying laws applicable to a highlighted ACSR term.

State space exploration, equivalence testing and interactive execution operate on a labeled transition system (LTS) representation of the system being analyzed. The LTS for one or more processes is produced by an algorithm that expands the process to produce a labeled transition system representing all possible executions. The LTS construction algorithm also prunes edges made unreachable by the semantics of the prioritized transition system, in most cases reducing the size of the resulting LTS.

State space exploration analysis can be used to determine key properties of a system's LTS. These include (1) number of states and transitions; (2) presence of deadlocked states; (3) states capable of *Zeno* behaviors (*i.e.*, infinite sequences of instantaneous events); (4) states that require synchronization to take place before time can progress; and (5) reachability of specific externally observable events.

Process equivalence can be tested using a number of different notions of equivalence including syntactic equivalence, a weaker syntactic equivalence which allows renaming of process variables and simple changes in structure, prioritized strong equivalence, and prioritized weak equivalence. In the order listed, these notions of equivalence increase in computational complexity and decrease in "strength" (i.e., equate more terms).

The interactive execution feature allows user-directed execution of process specifications. The user may interactively step through the LTS one action at a time, produce traces from random executions of the LTS, save process configurations to a stack for later analysis while an alternate path is explored, and analyze the size and deadlock characteristics of the LTS resulting from their process.

The XVERSA toolset has been used successfully to model and analyze railroad crossing systems[LBAC96], airport taxiways[BALC95], real-time schedulability analysis problems[CLX95a], the Philips audio control protocol[BPV94], the production cell case study[LL95, BA96], and to verify the correctness of a Sunshine ATM switching network[CL95].

## 4 Summary

We have presented XVERSA, a toolset that supports the formal analysis of resource-bound real-time systems. XVERSA offers a graphical process description language and X-Windows based tools that automate time consuming and error-prone analysis tasks. Our research into the theory of ACSR/GCSR and supporting tools is ongoing. Current goals include (1) the enhancement of ACSR/GCSR with value-passing; (2) the development of a refinement theory for ACSR/GCSR processes; and (3) implementation of alternative semantic representations.

Further information on ACSR and GCSR is available on the World Wide Web at

`http://www.cis.upenn.edu/~rtg/home.html`.

The XVERSA tools and descriptions of several case studies using XVERSA are available from

`http://www.cis.upenn.edu/~lee/duncan/versa.html`.

Questions about downloading and installing the tools should be addressed to `versa@saul.cis.upenn.edu`.

## References


[BA96] Hanêne Ben-Abdallah. *Graphical Communicating Shared Resources: A Language for the Specification, Refinement, and Analysis of Real-Time Systems.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1996.

[BALC95] Hanêne Ben-Abdallah, Insup Lee, and Jin-Youg Choi. A graphical language with formal semantics for the specification and analysis of real-time systems. In *Proc. of IEEE Real-Time Systems Symposium*, Pisa, Italy, December 1995.

[BB91] J.C.M. Baeten and J.A. Bergstra. Real Time Process Algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.

[BPV94] D. Bosscher, I. Polak, and F. Vaandrager. Verification of an Audio Control Protocol. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *FTRTFT '94: Formal Techniques in Real-time and Fault-tolerant Systems*, pages 170–192. LNCS 863, Springer-Verlag, 1994.

[CL95] Duncan Clarke and Insup Lee. A hybrid approach to formal verification applied to an ATM switching system. Technical report, Dept. of CIS, Univ. of Pennsylvania, Dec 1995.

[CLX95a] J-Y. Choi, I. Lee, and H-L Xie. The Specification and Schedulability Analysis of Real-Time Systems using ACSR. In *Proc. of IEEE Real-Time Systems Symposium*, December 1995.

[CLX95b] D. Clarke, I. Lee, and H. Xie. VERSA: A tool for the specification and analysis of resource-bound real-time systems. *Journal of Computer and Software Engineering*, 3(2), April 1995.

[Har87] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[JM94] F. Jahanian and A.K. Mok. Modechart: A specification language for real-time systems. *IEEE Transactions on Software Engineering*, 20(12):933–947, December 1994.

[LBAC96] Insup Lee, Hanêne Ben-Abdallah, and Jin-Young Choi. A process algebraic method for the specification and analysis of real-time systems. In Constance Heitmeyer and Dino Mandrioli, editors, *Formal Methods for Real-Time Computing*, chapter 7. John Wiley & Sons, Chichester, January 1996.

[LBGG94] I. Lee, P. Brémond-Grégoire, and R. Gerber. A Process Algebraic Approach to the Specification and Analysis of Resource-Bound Real-Time Systems. *Proceedings of the IEEE*, 82(1):158–171, January 1994.

[LG85] I. Lee and V. Gehlot. Language Constructs for Distributed Real-Time Programming. In *Proc. IEEE Real-Time Systems Symposium*, 1985.

[LL95] Claus Lewerentz and Thomas Linder, editors. *Formal Development of Reactive Systems: Case Study Production Cell*, volume 891 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[MT90] F. Moller and C. Tofts. A Temporal Calculus of Communicating Systems. In *Proc. of CONCUR '90*, pages 401–415. LNCS 458, Springer Verlag, August 1990.

[NS94] X. Nicollin and J. Sifakis. The Algebra of Timed Processes ATP: Theory and Application. *Information and Computation*, 114(1):131–178, October 1994.

[Sha92] A. Shaw. Communicating Real-time State Machines. *IEEE Transactions on Software Engineering*, September 1992.