LARVA Converter User Manual

Christian Colombo

October 8, 2008

Contents

1 Introduction

Several notations can be used to represent properties of computer systems. We provide a conversion to LARVA for some of these notations. This document provides the information to use a suite of converters which act as a bridge from some notation to LARVA. The LARVA code generated is not complete for the simple reason that the user has to insert additional information about the relationship between the variables in the notation and the events of the system to be monitored. The application here described can also be used to check whether a counterexample formula is slowdown truth preserving and whether it is speedup truth preserving. Counterexample formulae are a subset of duration calculus for which we have proved a subset which is slowdown truth preserving and a subset which is speedup truth preserving.

2 Parameters

Running the LARVA Converter application requires the specification of the following two parameters:

Input file The first parameter is the input file name. This should be the absolute path of the file or relative to the working directory. In order to specify the type of notation contained in the file, the user should also use one of the following switches: *-LUSTRE* for LUSTRE code,*-QDDC* for QDDC,*-CE* for counterexample formulas, *-IMPL* for implementables. This switch should be placed exactly before the input file name with a space in between. A section for each of these notations will follow where we will explain in some detail what can be entered in the file.

Output file The user should usually also specify an output file where the output generated by the converter should be saved. The output is nonetheless output in the console. To specify the output file the user should use the switch *-o* and enter the path of the file with a space between them.

3 LUSTRE

The synchronous language LUSTRE can be used to specify security properties by specifying nodes which return true or false depending whether the state of the system is accepted or not. The input streams (variables) should be extractable from the target system through events. These will then be specified and added to the output of this application before it can be used as input for the LARVA compiler. A LARVA script will be created for each LUSTRE node which has not been used by other nodes. All these scripts will be output in the specified output file.

4 QDDC

A QDDC formula can be used to represent system properties by describing behaviours which are accepted. The accepted fragment is the following:

$$
\begin{array}{lll}
F & ::= & always(F) \mid eventually(F) \mid end(P) \mid repeat(F) \mid (G)then(F) \\
 & & \mid F\ and\ F \mid not\ F \mid F\ or\ F \mid G \\
G & ::= & age(P) \sim l \mid sigma(P) \sim l \mid count(P) \sim l \mid len \sim l \mid bounded(P, c, l) \\
 & & \mid P\ leadsto(l)\ P \mid P\ persist(l)\ P \mid stable(P, l) \mid begin(P) \mid [P]0 \mid [[P] \\
 & & \mid [[P]] \mid [[P]+ \mid [[P]]+ \mid G\ and\ G \mid G\ or\ G \\
P & ::= & P \mid P\ or\ P \mid P\ and\ P \mid not\ P
\end{array}
$$

where $\sim \in \{<, <=\}$; *[]* represents $\lceil\rceil$; *]0* represents $\rceil^0$; *]+* represents $\rceil^+$; $l, c \in \mathbb{N}$.

The fragment P represents a boolean predicate of the system while the fragment F is a QDDC formula.

5 Counterexample Formulas

Counterexample formulas [1] are a subset of duration calculus which is expressive enough to represent the set of implementables [2, 3]. The syntax accepted by the converter is as follows:

$$
\begin{array}{rcl}
ce\_formula & ::= & \neg(phase \frown (phase \mid events) \\
 & & \quad \frown \ldots \frown (phase \mid events) \frown true) \\
phase & ::= & (true \mid \lceil Predicate \rceil)[\wedge\ \ell \sim t] \\
 & & [\wedge \boxminus NAME \ldots \wedge \boxminus NAME] \\
\sim & ::= & \leq \mid < \mid > \mid \geq \\
events & ::= & \updownarrow NAME \mid \not\updownarrow NAME \\
 & & \quad events \vee events \mid events \wedge events
\end{array}
$$

for the console input use ; instead of $\frown$, [] instead of $\lceil\rceil$, <=, >= instead of $\leq$, $\geq$, *and*, *or* instead of $\wedge$, $\vee$, *change*$(NAME)$ instead of $\updownarrow NAME$, *invchange*$(NAME)$ instead of $\not\updownarrow NAME$, *nochange*$(NAME)$ instead of $\boxminus NAME$.

An a theoretical note, the difference between $\lceil true \rceil$ and *true* is that *true* is satisfied by an empty interval while $\lceil true \rceil$ is satisfied by any non-zero length interval.

The counterexample converter also returns information about slowdown truth preservation and speedup truth preservation (sdtp, sudp). This means that a formula which is sdtp and holds on an interpretation, will also hold on any stretched variation of that interpretation. Similarly, a formula which is sutp and holds on an interpretation, will also hold on any compressed variation of that interpretation.

6 Implementables

The following formulae are implementables [3]:

1. Initialisation: $init(P)$ demands that the system starts in a state satisfying P.
2. Sequencing: $seq(P, P1)$ demands demands that the system can evolve from state P only to a state satisfying P1.
3. Progress: $prog(P, t)$ demands that state P must be left after at most t seconds.
4. Stability: $stab(P, Q, P1, t)$ demands that if state P is entered while Q holds it is stable or only evolves to a state satisfying P1 during the next t seconds.
5. Synchronisation: $synch(P, Q, t)$ demands that state P must be left within t seconds if Q holds.

The implementables are converted to counterexample traces and then processed accordingly. The sdtp and sudp information is also issued for implementables.

References

[1] J. Hoenicke. *Combination of Processes, Data, and Time*. PhD thesis, University of Oldenburg, July 2006.

[2] A. P. Ravn. *Design of Embedded Real-Time Computing Systems*. PhD thesis, Technical University of Denmark, October 1995.

[3] M. Schenke and E.-R. Olderog. Transformational design of real-time systems part i: From requirements to program specifications. *Acta Inf.*, 36(1):1–65, 1999.