Systematic Design of Real-Time Systems Based on CSP+T Process Algebra

Manuel I. Capel\textsuperscript{1}, José R. Balsas\textsuperscript{2} and Juan A. Holgado\textsuperscript{1}

\textsuperscript{1} Depto. Lenguajes y Sistemas, Universidad de Granada, 18071 Granada, Spain
\textsuperscript{2} Dpto. Informática, Universidad de Jaén, 23071 Jaén, Spain

Abstract. In this paper, a bottom-up formal technique to obtain a correct system specification from the RT/SA requirements specification of real-time systems is proposed. As an application, a design of the production cell is introduced.

1 Introduction

We present a complete bottom-up systematic specification technique in order to derive a correct system specification from a semi-formal user requirements specification in RT/SA \cite{2, 3} by applying a set of transformation rules. The proposed technique integrates two complementary approaches to describe a real-time system: (1) RT/SA based notations, and (2) CSP+T \cite{4} process terms to model real-time processes including the specification of their timing requirements.

A semantic interpretation of RT/SA entities must be flexible enough to accept alternative interpretations of some analysis entities in order to maintain the adaptability of RT/SA to different modeling cases \cite{1}. Thus, our method does not fix a particular semantics, when there are different possibilities to solve a given ambiguity of RT/SA and it is up to the analyst to select the most convenient notation semantics depending on the system to be specified. This feature of the method is a result of the flexibility provided by CSP+T design notation.

2 Real-Time Systems Specification with CSP+T

The use of extensions of algebraic process description languages such as Timed CSP or CSP+T gives a precise and flexible semantics to RT/SA entities. CSP+T description language is a powerful notation to specify deterministic processes with time constrained behavior. There are only a few new operators, which are mainly related to the specification of time, in CSP+T with respect to those offered by CSP.

A new operator \(\star\) (star) is introduced to denote process instantiation. This event is unique in the system since it represents the origin of a global time at which the processes start their execution.

The event operator \(\triangleright\triangleleft\) is used jointly with a variable to record the time instant at which the event occurs. The variables associated with this operator are called marker variables and their scope is strictly limited to one sequential process. Each event is associated with the event enabling interval during which the event is available for
communication between the process and its environment, and is relative to a preceding event. The following term is an example of a process written in CSP+T.

\[ P = 0. \star \rightarrow [1, 2] a \triangleright \langle v \rightarrow \text{TOP} \]

After the execution of the above process \( P \), the value of the marker variable satisfies the inequality \( 1 \leq v \leq 2 \).

**Transformation rules for RT/SA entities**

1. **Modeling input interface.** This consists of an input communication symbol for every origin entity \( O \) communicating data or control signals towards \( P \), and, vice-versa, of an output communication for every destination entity \( D \). \( O \) and \( D \) are RT/SA entities with the only limitation being that neither of them are data stores (DS).

2. **Modeling continuous data flows.** These cannot be directly modeled by means of communication events. It is therefore necessary to write an extra CSP+T process for each continuous data flow.

3. **Modeling state-transition diagrams.** Every control transformation scheme (CTP) is represented by a unique state-transition diagram (STD) from the point of view of control specification. An STD can be formally defined as a tuple \((Q, C, A, T, q)\) in which \( Q \) is a set of states; \( C \) is a set of conditions, each one corresponding to an input flow of control in the CTP; \( A \) is a set of actions, each one causing the execution of an activity in the process; \( T \) is a set of transitions, where each one is a tuple defined as \((q_1, c, a, q_2)\) in which \( q_1, q_2 \in Q \), \( c \in C \) or is null, \( a \in A \) or is null.

4. **Modeling timed control transformation processes.** A timed STD is an initial STD plus the timing constraints imposed on the system. The transition concept can be extended to specify timing constraints in the system by describing enabling intervals and marker events. These constraints are described as a set \( R \) of tuples \((e_1, I, e_2)\) in which \( e_1 \in C \) or \( e_1 \in A \), and \( e_1 \) is called the marker event, \( I \) is a real number interval of the form \([\alpha, \beta]\), where \( \alpha, \beta \in \mathbb{R}^+ \), and \( \alpha \leq \beta \) or \( I \) is an interval relative to the preceding event or to event \( e_1 \). The event \( e_2 \in C \) or \( e_2 \in A \) is called a restricted event. The interpretation of a timing constraint \( R \) is as follows: event \( e_2 \) can only occur within the interval of time \( I \) from the occurrence of \( e_1 \).

5. **Modeling data and control storages.** A DS or a control store (CS) with input flows \( \{f_{i1}, ..., f_{in}\} \) and output flows \( \{f_{o1}, ..., f_{om}\} \) is modeled using the interface \( \{f_{i1}, ..., f_{im}, f_{o1}, ..., f_{om}\} \) that comprises all communication actions by which the process interacts with its environment.

6. **Modeling transformation specifications.** We suppose that the functionality of primitive data transformation schemes (DTPs) is simple enough to allow the analyst to obtain a CSP+T process for each DTP.

**The bottom-up design method proposed**

We follow a systematic design process that consists of the following steps:

1. Prepare the RT/SA schemes to perform the transformation into CSP+T processes. It may be necessary to rename analysis entities to avoid conflicts.
2. Transform the CTPs and DTPs of the lower level.
3. Select the other schemes in ascending order.
4. Once the CSP+T model has been obtained for all the entities in a scheme, a
CSP+T process is defined to model the complete scheme. If this scheme is already included in a CTP or DTP of a higher level, repeat step (3), thus progressively integrating the CSP+T model of the system in an ascending way. The process finishes when the model of the System Context Diagram is obtained.

3 Specification example: The Production Cell

As an application result of our method, a complete CSP+T model for the production cell problem can be found at http://lsi.ugr.es/~sc/vveis04.pdf. We now present part of the design of the Robot Control Process (RCP).

\[
\begin{align*}
\text{RCT} &= \text{start} \rightarrow \text{actions (start)} \rightarrow \text{RTurnCW} \\
\text{RTurnCW} &= \{ \text{robot_pos}_1 \rightarrow \text{action (RTurnCW)} \rightarrow \text{POS}_1 \\
& \quad \mid \text{robot_pos' } \rightarrow \text{robot_pos} \rightarrow \text{RTurnCW} \} \\
\text{POS}_1 &= \{ \text{I}_1, \text{a}_{_\text{finish}} \rightarrow \text{actions (POS}_{_1,\text{CCW}} \rightarrow \text{RturnCCW} \rightarrow \text{POS}_1 \\
& \quad \mid \text{I}_1, \text{a}_{_\text{finish}} \rightarrow \text{disable}_a \rightarrow \text{action (blank_timeout)} \rightarrow \text{RTurnCCW} \} \\
\end{align*}
\]

Its associated enabling intervals are defined as follows: \( I_1, (a_{_\text{finish}}) = [t, t+T] \), which indicates that the robot arm1 \( (a) \) picked up a blank from the rotary table; \( I_2, (a_{_\text{finish}}) = (t+T, \infty) \), in this case there is no blank to pick up within time \( T \). We have two possible cases to model: the first one represents the presence of a blank on the rotary table that the arm1 gripper will be able to pick up, the action \( a_{_\text{finish}} \) is therefore executed, then arm1 is positioned and is prepared to extend to pick up the blank; the second one occurs when no new blank prompts on time \( (i.e., \text{within the deadline } T) \) causing an exception to be raised to inform the system of a \( \text{blank_timeout} \) event. When it reaches the position \( \text{POS}_1 \), the control of the robot starts turning arm1 counterclockwise (CCW) until arm2 picks a forged plate from the press, or arm2 \( (a) \) points to the deposit belt to place a plate on it, or arm1 goes to the position in which it deposits a blank on the press.

Conclusions

Our methodological scheme uses CSP+T process algebra to provide the user with a set of patterns into which they can translate from RT/SA entities into CSP+T processes with different semantics.

We are currently working on the development of a formal software tool based on CTJ and Java, capable of automated specification, verification and code generation of real-time and embedded system software for several platforms.

References