Towards a Denotational Semantics of Timed RSL Using Duration Calculus

LI Li (李黎)
Department of Computer Science and Technology
University of Science and Technology of China, Hefei 230027, P.R. China
E-mail: lli@venus.sui.ustc.edu.cn
Received May 26, 1999; revised May 8, 2000.

Abstract The Timed RAISE Specification Language (Timed RSL) is an extension of RAISE Specification Language by adding time constructors for specifying real-time applications. Duration Calculus (DC) is a real-time interval logic, which can be used to specify and reason about timing and logical constraints on duration properties of Boolean states in a dynamic system. This paper gives a denotational semantics to a subset of Timed RSL expressions, using Duration Calculus extended with super-dense chop modality and notations to capture time point properties of piecewise continuous states of arbitrary types. Using this semantics, the paper presents a proof rule for verifying Timed RSL iterative expressions and implements the rule to prove the satisfaction by a sample Timed RSL specification of its real-time requirements.

Keywords duration calculus, RAISE specification language, denotational semantics, real-time system

1 Introduction

The RAISE Specification Language (RSL)[1] is one of the most versatile and comprehensive languages for formal specification, design and development of software. However, it has no particular features for specifying real-time applications. To solve this problem, RSL is extended to Timed RAISE Specification Language (Timed RSL)[2] by introducing some time constructors. With the time constructors, Timed RSL can specify time-dependent operations, but cannot specify real-time requirements at an abstract level. Therefore, we need to give a denotational semantics of Timed RSL using a real-time logic, so that we can verify whether a Timed RSL specification satisfies its real-time requirements described in this real-time logic.

Duration Calculus (DC)[3] is a real-time interval logic which has been used to specify and verify designs for a number of real-time systems. In this paper, we give a denotational semantics to a subset of Timed RSL using an extension of Duration Calculus. The logic is called Extended Duration Calculus with Super-dense chop, abbreviated as SEDC, and it has incorporated features from several extensions of Duration Calculus[4,5].

The laws of SEDC can be used to derive properties of a Timed RSL expression as logical consequences of its DC semantics. For iterative expressions, which are defined as the weakest fixed point, we provide a special proof rule for establishing properties of interest. After several simple properties have been established, more complex properties and finally the satisfaction of user-defined real-time requirements can be proved. This process is demonstrated by an example.

This paper is organised as follows. Section 2 defines the logic SEDC. The denotational semantics of Timed RSL is presented in Section 3. Section 4 is devoted to a case study,

This work is partially supported by the National Natural Science Foundation of China (No. 69773025).
where the high-level proof rule of Timed RSL is used in verification of the design of an alarm system. The paper ends with a brief summary and discussion.

2 Extended Duration Calculus with Super-Dense Chop

The classical Duration Calculus\cite{3} was developed to specify and reason about timing and logical constraints on duration properties of Boolean states. This is inadequate for describing semantics of Timed RSL expressions, where states can be in arbitrary types and its time point properties are also important in some circumstances. In Extended Duration Calculus (EDC)\cite{4}, state variables can be arbitrary typed functions of time. Initial and final values are also introduced in EDC by right and left limits over time. To model state transitions, several of which may occur at one time point (this is so-called super-dense computation), DC is extended to SDC\cite{5} with neighbourhood properties and a super-dense chop modality.

Our logic SEDC is a combination of SDC and a modified version of EDC. Since neighbourhood properties are defined in SDC, we define initial and final values at points to capture point properties.

2.1 Syntax and Semantics

The three important syntactic categories are state expressions, duration terms, and duration formulae. The state expressions are constructed from sets of basic symbols: global variables, state variables, and function symbols. Their meanings are given by an interpretation $I$.

In the following, we use real numbers to represent time, with typical element $t$; and define $\text{Intv}$ to be the set of bounded and closed time intervals, with typical element $[b, e]$: \[
\text{Intv} \overset{\text{def}}{=} \{ [b, e] | b, e \in \text{Real}, b \leq e \}
\]

Global variables: $x_i$, $i = 0, \ldots$ with typical element $x$. Each global variable $x$ has an associated type denoted as $\text{Type}(x)$. For types, we generally use $T_i$, $i = 0, \ldots$, with typical element $T$. Timed RSL\cite{2} types like $\text{Real}$, $\text{Bool}$ and $\text{Time}$ are predefined types, where $\text{Time} = \{ r : \text{Real} \mid r \geq 0.0 \}$.

The meaning of a global variable $x$ is a value: $I(x) \in \text{Type}(x)$.

State variables: $v_i$, $i = 0, \ldots$, with typical element $v$. The type of a state variable $v$ is denoted as $\text{Type}(v)$.

Function symbols: $F_i$, $i = 0, \ldots$, with typical element $F$. Each $F$ has an associated arity denoted as $n(F)$, and a signature denoted by a sequence of argument types followed by a result type: $(T_1, \ldots, T_{n(F)} + 1)$. Constants are 0-ary function symbols, and operators on real numbers $+, -, \ldots$ are 2-ary function symbols.

A subset of function symbols is predicate symbols $R_i$, $i = 0, \ldots$, with typical element $R$, where the result type is $\text{Bool}$. The predicate symbols include: the Boolean constants (true, false), the Boolean operators ($\neg$, $\lor$, $\land$, $\Rightarrow$, $\Leftrightarrow$), and the relation operators ($<$, $>$, $=$, $\leq$, $\geq$, $\neq$).

The meaning of a function symbol $F$ having signature $(T_1, \ldots, T_{n(F)} + 1)$ is defined by: $I(F) \in T_1 \times \cdots \times T_{n(F)} \rightarrow T_{n(F)} + 1$. Operators on real numbers, Boolean operators, and relation operators have their usual meaning. The meanings of constants are the corresponding values of their types.

State Expressions: State expressions $E_i$, $i = 0, \ldots$, with typical element $E$, are built from global variables ($x$), state variables ($v$), and function symbols ($F$): \[
E ::= x \mid v \mid F(E_1, \ldots, E_{n(F)})
\]