SDL for Real-Time: What is Missing?

Iulian Ober
Alain Kerbrat

TELELOGIC, Toulouse

Marius Bozga
Susanne Graf
Laurent Mounier

VERIMAG, Grenoble

Daniel Vincent

France Telecom

Overview

- SDL and real-time: status, pros, cons
- high-level real-time modeling issues
  - problems
  - proposal: timed simulation semantic framework
- open issues
  - real-time programming
  - semantic profiles
- related work, conclusions

SDL & Real-Time: Facts

- “language for specification and description of telecommunications systems” (Z.100)
- increasingly used in other RT application areas: process control, automotive...
- builds on the same concepts as other RT modeling languages (ROOM, RT-UML?)

SDL & Real-Time: Pros

- formal semantics
- powerful development environments

support for
intensive validation,
essential for
critical systems

- may be used for both
  - high-level modeling
  - programming
- supports a large part of a project's lifecycle

SDL & Real-Time: Cons

(high-level modeling)

- formal semantics must be adapted for simulation and verification
- cannot capture execution and communication times
- does not account for scheduling
- cannot model temporal non-determinism

SDL & Real-Time: Cons
(programming)

- cannot specify interruptive emergency procedures
- there are no native constructs for mutual exclusion and synchronization

High-level RT Modeling in SDL, Issue #1

Control over Time

- time progress (highlights from Z.100):
  - action execution time is unspecified
  - agents may stay ready for an unspecified amount of time

  ⇒ can lead to unrealistic simulation scenarios
- time progress must be controlled by the simulator according to the system specification

  ⇒ tools make simplifying assumptions which can exclude realistic simulation scenarios

High-level RT Modeling in SDL, Issue #2

Execution Times

- no assumptions on execution times
- necessary in simulation, verification, performance analysis, test generation, scheduling analysis...
- execution times must be emulated using timers

High-level RT Modeling in SDL, Issue #3
Flexible Channel Specification

- SDL channels model perfect links:
  - never lose messages
  - transmission is either instantaneous or with unspecified delays
- more complex channel characteristics must be modeled (e.g. for flow control protocols)
  - loss probability (laws)
  - lower and upper bounds for delays, probability laws

High-level RT Modeling in SDL, Issue #4
Scheduling

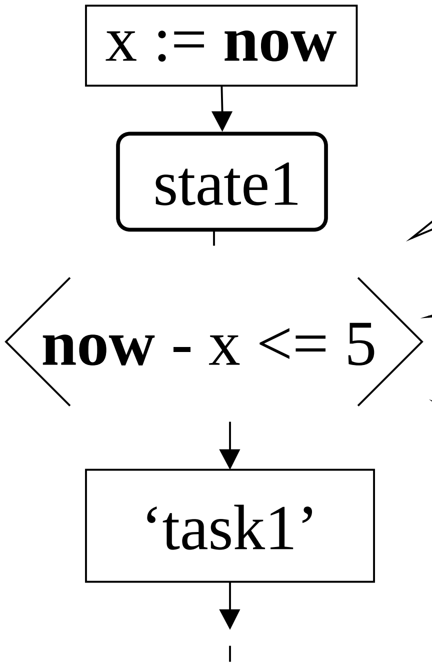
- important RT design tasks:
  - defining the scheduling policy and
  - determining scheduling parameters (e.g. priorities)
- (?) SDL must support scheduling analysis
- (!) SDL must support validation over different scheduling policies

Timed SDL Semantics
Concepts

- simulation time is guided by the SDL model
- time passes in (simulation) states, depending on the enabled transitions
- transitions are assigned urgencies (urgency = priority over time passage)
  - eager: absolute priority over time
  - lazy: same priority as time (=> non-determinism)
  - delayable: marginal priority over time (=> non-determinism)

Timed SDL Semantics

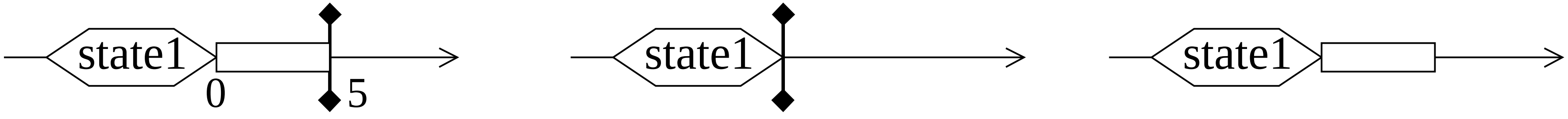
Example



delayable : transition is taken 0 to 5 time units after entering **state1.** Time may not pass beyond 5 t.u.

eager : transition is taken immediately after entering **state1**

lazy : transition is either taken 0 to 5 time units after entering **state1**, or it is not taken at all



Timed SDL Semantics

Usage

- urgencies are just underlying semantics
- default urgency examples:
  - internal INPUT, PRIORITY INPUT à eager
  - feeds from the environment à lazy
- user level extensions:
  - action duration intervals
  - communication time intervals

  à one implicit simulation state + one delayable transition

Timed SDL Semantics

Analysis Methods & Results

- symbolic methods for handling time
  (transferred from timed automata)
- simulation is closer to the real-time world
  - covers all realistic behaviors wrt time
  - avoid most of the unrealistic scenarios
- interesting properties may be derived/verified:
  - minimal/maximal time between arbitrary events
  - invariants on timers and timer relationships

RT Programming in SDL, Issue #1
Timeout Emergency Actions

- no real timeout emergency actions:
  - timer messages are received asynchronously
  - if the receiver is not ready, the timeout action is not executed immediately
- SDL'2000 has emergency actions: exceptions
- we need a connection between the system time and the exception mechanism

RT Programming in SDL, Issue #2
Synchronization Constructs

- synchronization (mutual exclusion, or general-form synchronization) is important in most concurrent systems
- no native synchronization mechanisms in SDL
- external code may be inserted in the SDL code
  - hampers simulation, verification and portability

SDL Semantics: Profiles

- diverging uses of the semantics => different semantic profiles
    - e.g. profile for code generation, profile for simulation and verification
- lightweight variations of the semantics => parametric profiles
    - e.g. parameter for the atomicity of transitions
- inter-profile compliance should be studied and formalized

Related Work

ObjectGEODE Performance Evaluation Extensions

- all-eager semantics + time consuming actions
- no general solution for time progress control, atomicity, channel specification
- partial handling of temporal non-determinism
- discrete time, limited analysis possibilities

⇒ can be modeled with urgencies

Related Work

Queuing SDL - QUEST

- all-eager semantics + time consuming actions + facilities for performance measurement
- action duration is computed dynamically
  => can model overloading
- discrete time, limited verification possibilities

Conclusions and Future Work

- identified RT-related weaknesses of SDL
  - programming
  - high-level modeling
- proposed a new semantic framework
  & analysis methods
- prototyped the new semantics with encouraging results
- ongoing work on the open issues