ASM Semantics of SDL: Concepts, Methods, Tools

Uwe Glässer
HNI Paderborn, Germany
glaesser@uni-paderborn.de

Abstract State Machines
- Basic ASM Concepts
- Multi–Agent Real–Time ASM

Abstract SDL Machine
- Overall Organization
- Encoding of SDL Objects
- Operations on Signals and Timers

ASM Tool Support

Conclusions
Formal Semantics of SDL

ITU-T Recommendation Z.100  (≈ 200 pp. without Annexes)

Formal model of SDL (Meta-IV, CSP):

- Annex F.2:  Static properties of SDL  (437 pp.)
- Annex F.3:  Dynamic properties of SDL  (183 pp.)

Other Approaches

- \(\varphi_{SDL}: \) process algebra semantics  [Bergstra and Middleburg 96]
- Base SDL: Object-Z  [Lau and Prinz 95]
- SDL Time Nets: extended Petri Nets  [Fischer and Dimitrov 95]
- FOCUS: stream processing functions  [Broy 91], [Holz and Stølen 94]
- ...
Formal Semantics of SDL

Disclaimer

Z.100, Annex F.1, p. 1:

“This annex constitutes a formal definition of SDL. If any properties of an SDL concept defined in this document, contradicts the properties defined in Z.100 and the concept is consistently defined in Z.100, then the definition in Z.100 takes precedence and this formal definition requires correction.”
ASM Semantics of SDL

Our Approach: separate specification from verification

- operational semantics
  - embed a formal documentation into Z.100
  - reflect the common understanding of SDL
  - support alternative levels of abstraction

- approved meta-modelling concept

  previous work:
  ASM semantic of VHDL [Börger, Glässer, Müller 95]
Abstract State Machines

BASIC ASM CONCEPTS

States

Structures with domains and functions

\[ f(t_1, \ldots, t_n) := t_0, \quad n \geq 0 \quad \xrightarrow{\text{Update}} \quad \langle f^S(t_1^S, \ldots, t_n^S), \quad t_0^S \rangle \]

Update Sets

\[ \Delta_S(P) \quad \text{program } P \]

Computations

\[ S_0 \xrightarrow{\Delta_{S_0}(P)} S_1 \xrightarrow{\Delta_{S_1}(P)} S_2 \xrightarrow{\Delta_{S_2}(P)} \ldots \quad \text{“pure runs”} \]
General Abstraction Principles

Clear & Concise Specifications: “ground models”

- abstract operational views
  - hierarchical descriptions
  - incremental refinements
  - behaviour separated from context

- information hiding & interface mechanisms

  classification of ASM functions:
  - static
  - dynamic (controlled, monitored, shared ... global, private)
General Abstraction Principles

Classification of ASM Functions

ASM Functions

- dynamic
  - controlled
  - monitored
  - interaction
  - pseudo-interaction
    - mixed
    - indirectly controlled
    - indirectly monitored
    - indirectly shared

- static
  - derived
Multi-Agent Real-Time ASM

CONCURRENCY AND NONDETERMINISM

ASM Agents

ASM Program

Global State S

Modules

"partially ordered runs"
Global System Time

external (monitored) function:

\[
\text{now: } \text{TIME, } \text{TIME} \subseteq \mathbb{R}
\]

Agents perform instantaneous actions in continuous time:

An agent which is enabled at time \( t \) to fire a rule \( R \) actually fires \( R \) not later than \( t + \epsilon \) (for some infinitely small \( \epsilon \)).
Abstract SDL Machine

Interpretation Model (Basic SDL)

Abstract Interpreter

Process Module
Timer Module
Channel Module

“Partial Many-Sorted Structures” over $\Sigma_{SDL}$

Multi-Agent Real-Time ASM

SDL
Encoding of SDL Objects

Static Reachability Constraints

Reachability Sets

Choice of receiver and path:

\[
\text{choose_reachability} : \quad \text{PROCESS} \times \text{SIGNAL} \times \text{to-Arg} \times \text{via-Arg} \\
\rightarrow \text{PATH} \times \text{RECEIVER}
\]
Encoding of SDL Objects

Bidirectional Delaying Channel

Channel Module

\[ \forall j \in CH\_PATH \]

Channel Agent

\[ \forall j \in CH\_PATH \]

(channel operations)
Operations on Signals

Behaviour of Channels

\texttt{DeliverSignals}

\[ \equiv \text{do} \ \forall \ chp: \ CH\_PATH(chp) \ \text{and} \ \text{channel}(chp) = \text{Self} \]

\[ \ \text{if} \ \text{ReadyToDeliver}(chp) \ \text{then} \]

\[ \quad \text{queue}(chp) := \text{tail}(\text{queue}(chp)) \]

\[ \quad \text{let} \ si = \text{head}(\text{queue}(chp)), \ r = \text{receivername}(si) \ \text{in} \]

\[ \quad \text{if} \ r = \text{env} \ \text{then} \]

\[ \quad \text{DeliverToEnv}(si) \]

\[ \quad \text{else} \]

\[ \quad \text{DeliverToProcess}(si, r) \]

where

\[ \text{ReadyToDeliver}(chp) \]

\[ \equiv \ \exists si: \ SIGINST(si) \land si = \text{head}(\text{queue}(chp)) \land \neg \text{InTransit}(si, chp) \]
Operations on Signals

Behaviour of Channels (continued)

\[
\text{DELIVERToPROCESS}(S\text{Inst}, P\text{Name}) \equiv \begin{align*}
&\text{let } P\text{Id} = \text{receiverid}(S\text{Inst}) \text{ in} \\
&\text{if } P\text{Id} = \text{undef} \text{ then} \\
&\hspace{1em}\text{choose } p: \text{PID}(p) \text{ and } \text{procname}(p) = \text{receivername}(S\text{Inst}) \\
&\hspace{2em}\text{buffer}(p) := \text{buffer}(p)^\langle S\text{Inst}\rangle \\
&\text{else} \\
&\hspace{1em}\text{if } \text{PIDSys}(P\text{Id}) \text{ then} \\
&\hspace{2em}\text{buffer}(P\text{Id}) := \text{buffer}(P\text{Id})^\langle S\text{Inst}\rangle
\end{align*}
\]
Operations on Timers

Expiration Time

\[ \text{expire} : \text{TIMERINST} \rightarrow \text{TIME} \]

Behaviour of Timers

\[ \text{Active}(t) \equiv \]
\[ \text{ActiveTime}(t) \lor \text{ActiveSignal}(t) \]
\[ \text{ActiveTime}(t) \equiv \]
\[ \text{TIMERINST}(t) \land \text{expire}(t) \neq \text{undef} \]
\[ \text{ActiveSignal}(t) \equiv \]
\[ \text{TIMERINST}(t) \land \exists s \in \text{SIGINST} : t = \text{timer}(s) \land s \text{ in buffer(owner}(t)) \]
Operations on Timers
 Operations on Timers

\text{TimerOperation}
\equiv \text{if } MyAction(Self) \text{ then}
\hspace{1cm} \text{if } Action = set \text{ then}
\hspace{1cm} \text{let } time = fst(Arg) \text{ in}
\hspace{1cm} \text{SETEXPIRATIONTIME}(time)
\hspace{1cm} \text{DISCARDTIMERSIGNAL}
\hspace{1cm} \text{else}
\hspace{2cm} \text{if } Active(Self) \text{ then}
\hspace{3cm} \text{expire}(Self) := \text{undef}
\hspace{3cm} \text{DISCARDTIMERSIGNAL}
\hspace{1cm} \text{else}
\hspace{2cm} \text{if } ActiveTime(Self) \land now \geq expire(Self) \text{ then}
\hspace{3cm} \text{expire}(Self) := \text{undef}
\hspace{3cm} \text{CREATETIMERSIGNAL}
Operations On Timers

 Behaviour of Timers (continued)

\textbf{SetExpirationTime}(Time)
\begin{align*}
\equiv & \quad \text{if } Time = \text{undef} \text{ then} \\
& \quad \text{expire}(Self) := \text{now} + \text{duration}(\text{timername}(Self)) \\
& \quad \text{elif } Time \leq \text{now} \text{ then} \\
& \quad \text{expire}(Self) := \text{undef} \\
& \quad \text{CreateTimerSignal} \\
& \quad \text{else} \\
& \quad \text{expire}(Self) := Time
\end{align*}
Conclusions

Observations

- **SDL view** and **ASM view** of distributed “real-time” systems **coincide**:
  - notions of **concurrency**, **reactivity** and **time** are tightly related;
  - common understanding of SDL can **directly** be formalized.

- ASM semantics of Basic SDL
  - is particularly **concise**, readable and understandable;
  - can easily be **extended** and **modified**;
  - **bridging semantics** for combining SDL with other languages.

Further References

http://www.uni-paderborn.de/cs/asm/