# 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]
- ...

## 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."*

**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*

$$f(t_1, \ldots, t_n) := t_0\,, \quad n \geq 0 \quad \overset{Update}{\rightarrow} \quad \langle\ \underbrace{f^S(t_1^S, \ldots, t_n^S)}_{location}\ ,\ \underbrace{t_0^S}_{value}\ \rangle$$

### Update Sets

$\Delta_S(P)$ *program P*

### Computations

$$S_0 \overset{\Delta_{S_0}(P)}{\longrightarrow} S_1 \overset{\Delta_{S_1}(P)}{\longrightarrow} S_2 \overset{\Delta_{S_2}(P)}{\longrightarrow} \cdots$$

*"pure runs"*

# General Abstraction Principles

**Clear & Concise Specifications:** "*ground models*"

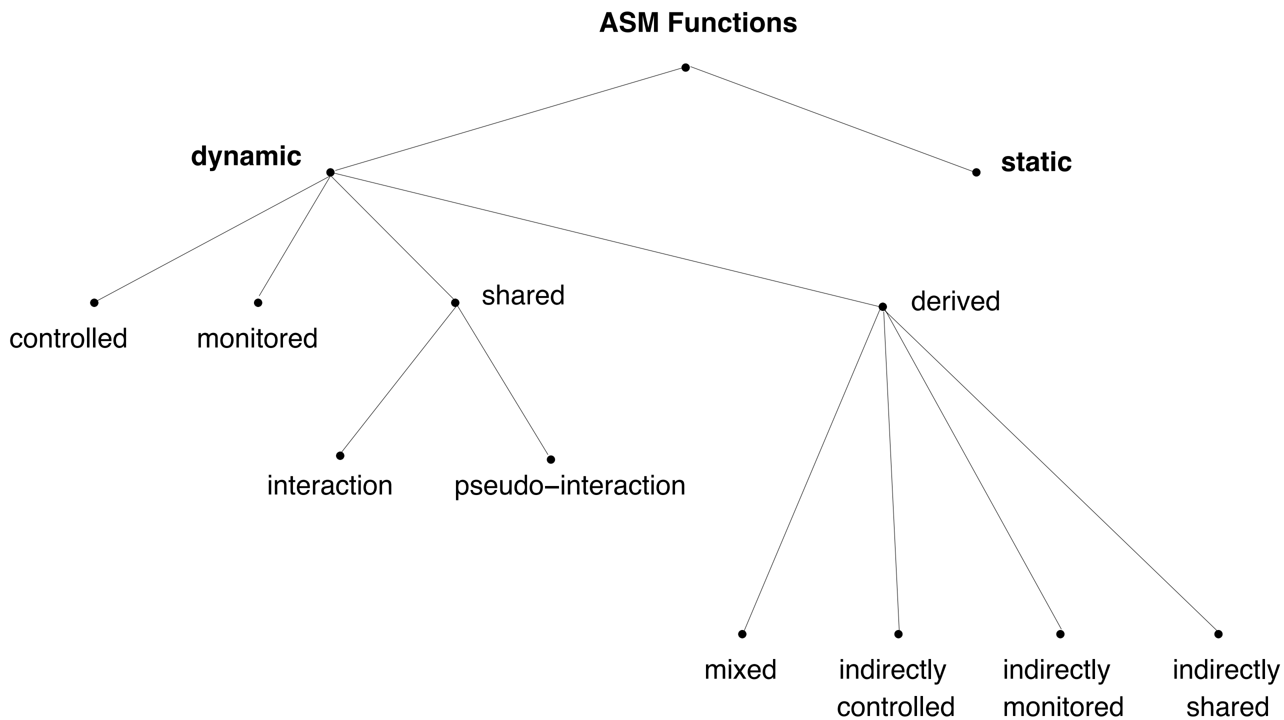
- **abstract operational views**
  - ▷ hierachical descriptions
  - ▷ incremental refinements
  - ▷ *behaviour* separated from *context*

- **information hiding & interface mechanisms**

  classification of ASM functions:

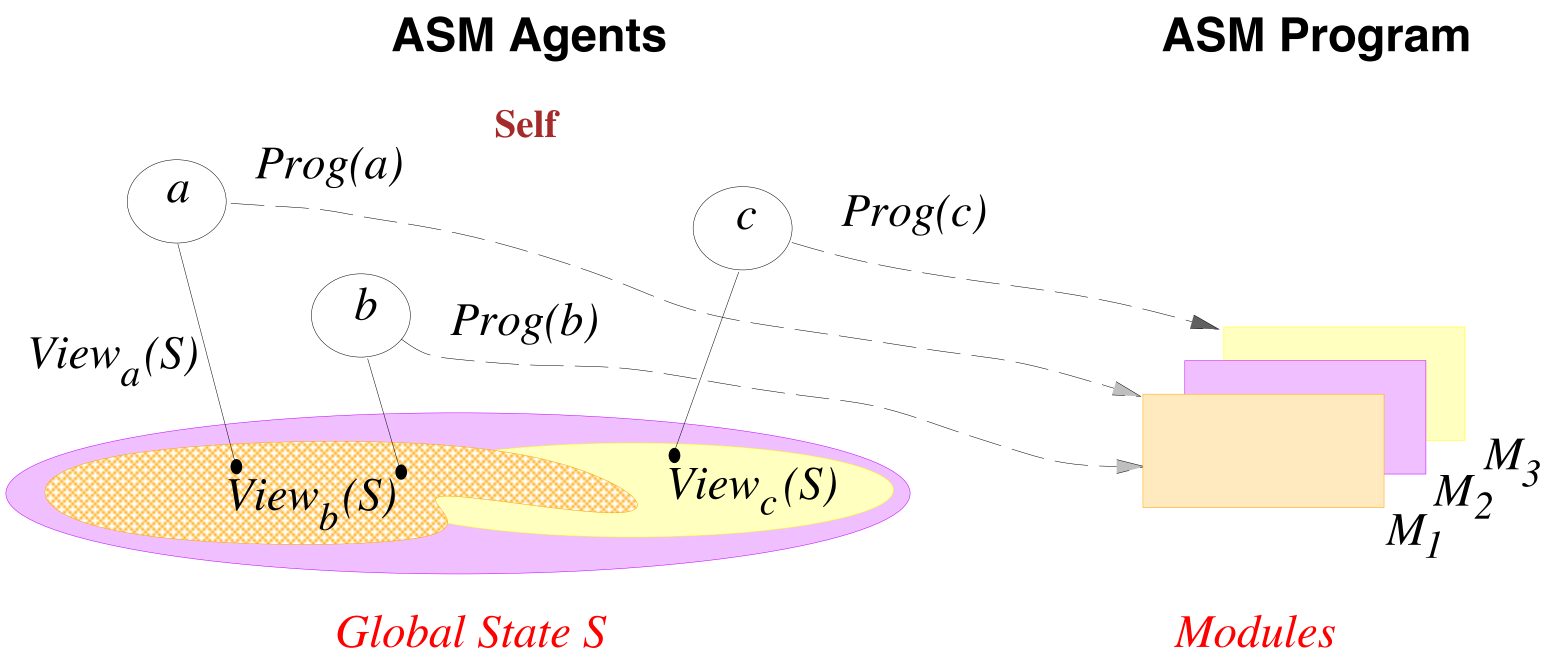
  - static
  - dynamic (*controlled, monitored, shared ... global, private*)

## Classification of ASM Functions



# Multi-Agent Real-Time ASM

CONCURRENCY AND NONDETERMINISM




"*partially ordered runs*"

## Global System Time

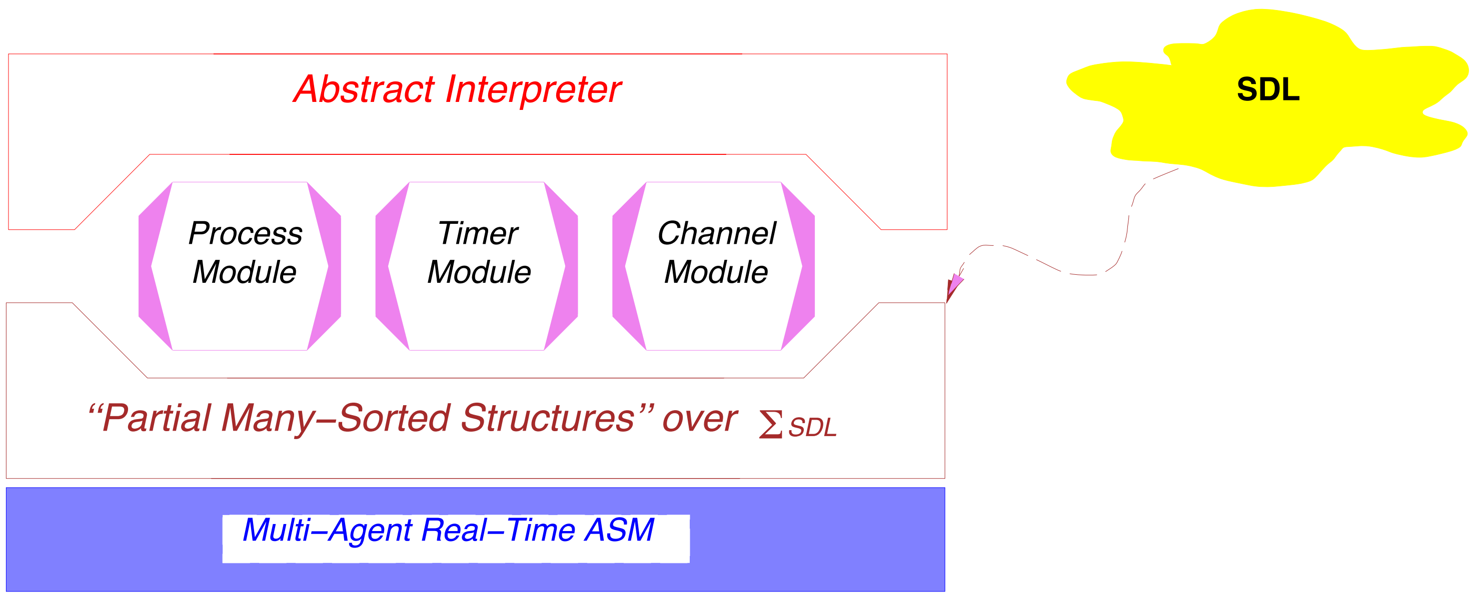
***external (monitored) function***:

$$now : \quad TIME, \qquad TIME \subseteq \mathbb{R}$$

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 $\epsilon$).

## Interpretation Model (Basic SDL)



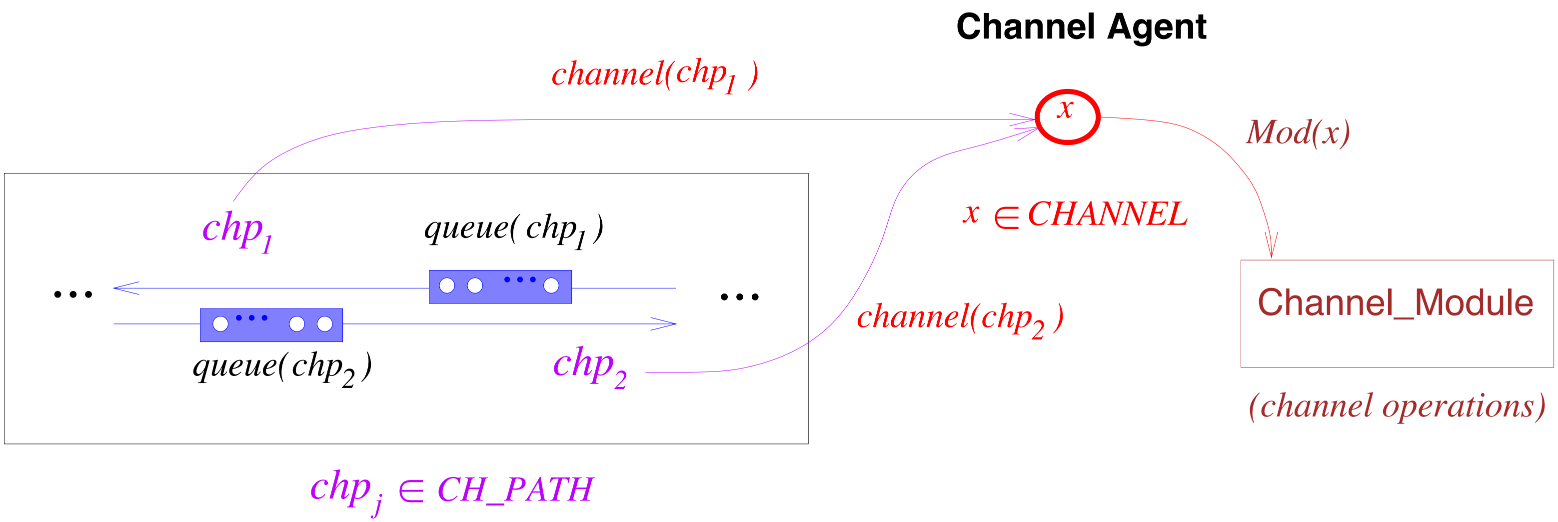
## Static Reachability Constraints

### Reachability Sets

Choice of *receiver* and *path*:

$$choose\_reachability : \quad PROCESS \times SIGNAL \times \textbf{to-Arg} \times \textbf{via-Arg} \\ \rightarrow PATH \times RECEIVER$$

## Bidirectional Delaying Channel



## Behaviour of Channels

$$
\begin{array}{l}
\textsc{DeliverSignals} \\
\quad \equiv \texttt{do forall } chp: \; \mathit{CH\_PATH}(chp) \; and \; channel(chp) = \mathit{Self} \\
\qquad \texttt{if } \mathit{ReadyToDeliver}(chp) \; \texttt{then} \\
\qquad\quad queue(chp) := tail(queue(chp)) \\
\qquad\quad \texttt{let } si = head(queue(chp)), \; r = receivername(si) \; \texttt{in} \\
\qquad\qquad \texttt{if } r = env \; \texttt{then} \\
\qquad\qquad\quad \textsc{DeliverToEnv}(si) \\
\qquad\qquad \texttt{else} \\
\qquad\qquad\quad \textsc{DeliverToProcess}(si, r) \\
\quad \texttt{where} \\
\qquad \mathit{ReadyToDeliver}(chp) \\
\qquad\quad \equiv \; \exists si: \; \mathit{SIGINST}(si) \wedge si = head(queue(chp)) \wedge \neg \mathit{InTransit}(si, chp)
\end{array}
$$

**Behaviour of Channels** (continued)

$$
\begin{array}{l}
\textsc{DeliverToProcess}(SInst, PName) \\
\quad \equiv \texttt{let}\ \ PId = receiverid(SInst)\ \ \texttt{in} \\
\qquad\qquad \texttt{if}\ \ PId = undef\ \ \texttt{then} \\
\qquad\qquad\quad \texttt{choose}\ p:\ \ PID(p)\ and\ procname(p) = receivername(SInst) \\
\qquad\qquad\qquad buffer(p) := buffer(p)^\frown\langle SInst\rangle \\
\qquad\qquad \texttt{else} \\
\qquad\qquad\quad \texttt{if}\ \ PID_{Sys}(PId)\ \ \texttt{then} \\
\qquad\qquad\qquad buffer(PId) := buffer(PId)^\frown\langle SInst\rangle
\end{array}
$$

## Expiration Time

$$expire : TIMERINST \rightarrow TIME$$

## Behaviour of Timers

$$Active(t) \equiv$$

$$ActiveTime(t) \vee ActiveSignal(t)$$

$$ActiveTime(t) \equiv$$

$$TIMERINST(t) \wedge expire(t) \neq undef$$

$$ActiveSignal(t) \equiv$$

$$TIMERINST(t) \wedge \exists\ s \in SIGINST : \ t = timer(s) \wedge s\ \ in\ \ buffer(owner(t))$$

# Operations on Timers

# Operations on Timers

$\text{TIMEROPERATION}$

$\equiv$ `if` $MyAction(Self)$ `then`

  `if` $Action = set$ `then`

    `let` $time = fst(Arg)$ `in`

      $\text{SETEXPIRATIONTIME}(time)$

      $\text{DISCARDTIMERSIGNAL}$

  `else`

    `if` $Active(Self)$ `then`

      $expire(Self) := undef$

      $\text{DISCARDTIMERSIGNAL}$

`else`

  `if` $ActiveTime(Self) \wedge now \geq expire(Self)$ `then`

    $expire(Self) := undef$

    $\text{CREATETIMERSIGNAL}$

**Behaviour of Timers** (continued)

$$
\begin{aligned}
&\textsc{SetExpirationTime}(\mathit{Time})\\
&\quad\equiv \texttt{if}\ \ \mathit{Time} = \mathit{undef}\ \texttt{then}\\
&\qquad\quad \mathit{expire}(\mathit{Self}) := \mathit{now} + \mathit{duration}(\mathit{timername}(\mathit{Self}))\\
&\qquad \texttt{elif}\ \ \mathit{Time} \leq \mathit{now}\ \ \texttt{then}\\
&\qquad\quad \mathit{expire}(\mathit{Self}) := \mathit{undef}\\
&\qquad\quad \textsc{CreateTimerSignal}\\
&\qquad \texttt{else}\\
&\qquad\quad \mathit{expire}(\mathit{Self}) := \mathit{Time}
\end{aligned}
$$

# 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/*