

Algebraic State Machines: Concepts and Applications to Security

Jan Jürjens

Software & Systems Engineering

Informatics, TU Munich

Germany



juerjens@in.tum.de

<http://www.jurjens.de/jan>



A Foundation for Security Engineering

Motivation to consider AlgSMs: UMLsec (UML extension for **Secure Software Engineering**).

Security requirements are **subtle**. Need formal foundation for mechanical **verification**.

Needed: formalism that supports

- **abstraction** (to scale to realistic systems)
- **asynchronous communication** (as in UML)
- **algebraic reasoning** (for cryptography)

Algebraic State Machines (AlgSMs)

Proposed by Broy, Wirsing based on
Abstract State Machines (Gurevich).

Each **state** is an **algebra**.

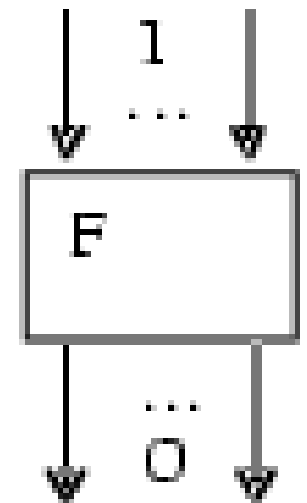
Asynchronous communication through
channels.

Here: **formal semantics** of AlgSMs as
stream-processing functions, **composition**,
refinement, applications to **security**.

AlgSM Definition

AlgSM \mathcal{A} given by:

- set **State** of Σ -algebras
- sets **I**, **O** (input, output channel names)
- state transition function
 $\Delta: \text{State} \times I^{[*]} \rightarrow \mathcal{P}(\text{State} \times O^{[*]})$,
- initial states $\text{State}^0 \subseteq \text{State}$



Interpretation: stream-processing fn's

AlgSM \mathcal{A} interpreted as $\llbracket \mathcal{A} \rrbracket: I^\rightarrow \rightarrow \mathcal{P}(O^\rightarrow)$:

For $f_1, f_2: \text{State} \rightarrow (I^\rightarrow \rightarrow \mathcal{P}(O^\rightarrow))$ define

$f_1 \leq f_2$ iff $f_1(s)(i) \subseteq f_2(s)(i)$ each s, i .

Then $\llbracket _ \rrbracket: \text{State} \rightarrow (I^\rightarrow \rightarrow \mathcal{P}(O^\rightarrow))$ largest s.th.

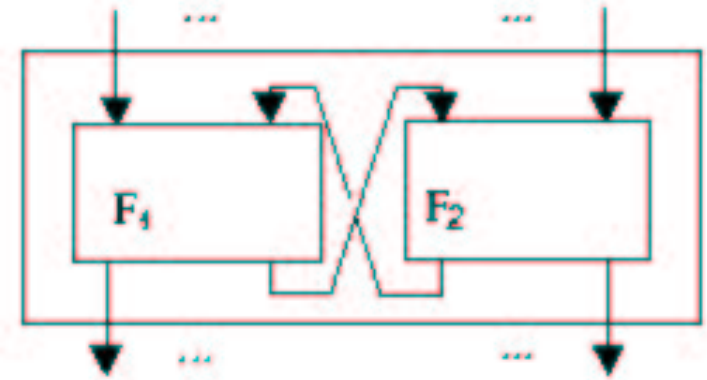
$\llbracket s \rrbracket(i_0.i) =$

$\{o_0.o: \exists s' \in \text{State}. ((s', o_0) \in \Delta(s, i_0) \wedge o \in s'(i))\}$

(all s, i_0, i).

Then $\llbracket \mathcal{A} \rrbracket(i) = \bigcup_{s \in \text{State}_0} \llbracket s \rrbracket(i)$ each i .

Composition



State = $\{A_1 \cup A_2 \cup A_3 :$

$$A_1 \in \text{State}_1 \wedge A_2 \in \text{State}_2 \wedge A_3 \in \text{Alg}(F_3)\}$$

where $F_3 = \{\text{feed}_l : l \in L\}$, $L = (O_1 \cup O_2) \cap (I_1 \cup I_2)$

$\Delta(A_1 \cup A_2 \cup A_3, i) =$

$$\{(B_1 \cup B_2 \cup B_3, o' \upharpoonright_o) : o' \in (O_1 \cup O_2)^{[*]}\}$$

$$\wedge \exists i' \in (I_1 \cup I_2)^{[*]}. (i = i' \upharpoonright_{I_1} \wedge (B_1, o \upharpoonright_{O_1}) \in \Delta_1(A_1, i' \upharpoonright_{I_1}))$$

$$\wedge (B_2, o \upharpoonright_{O_2}) \in \Delta_2(A_2, i' \upharpoonright_{I_2})$$

$$\wedge \forall l \in L. (i'(l) = \text{feed}_l^{A_3} \wedge \text{feed}_l^{B_3} = o(l))\}$$

(denotational definition).

Results

Theorem: Composition associative
(provided no confusion between
channels).

Theorem: Mapping to stream-processing
fn's preserves composition.

Proof: See extended version of paper.

Refinement

Reduce non-determinism by refinement.

Definition: AlgSM \mathcal{A} refines \mathcal{A}' if $[[\mathcal{A}']](i) \subseteq [[\mathcal{A}]](i)$ (each i).

Theorem: **Composition** and interpretation as **stream-processing fn's** are **preserved** by refinement.

Similarly: **timeless** refinement (abstract from time steps before applying refinement).
Preserves interpretation as stream-processing fn's (but not in general composition).

Applications to Security

Extend formalism with abstract **cryptographic operations** to reason about security.

Consider **variant** of **TLS** (SSL successor) proposed at IEEE Infocom'99:

$C \rightarrow S : N_j, K_C, \text{Sign}_{K_C^{-1}}(C::K_C)$

$S \rightarrow C : \{\text{Sign}_{K_S^{-1}}(k_j::K_C)\}_{K\{C\}}, \text{Sign}_{K_{CA}^{-1}}(S::K_S)$

$C \rightarrow S : \{s_i\}_{k_j}$

Showed: contains a **flaw**. **Proved** corrected version correct (see paper).

Future Work

With T. Kuhn (Siemens): Applications to **mobile security**.

Use to address issues in verification of cryptoprotocols vs. low-level properties of cryptoalgorithms ?

Some resources

Book: Jan Jürjens, Secure Systems
Development with UML, Springer-Verlag,
due 2003

Tutorials: Sept: FME (Pisa), FDL (Frankfurt),
SAFECOMP (Edinburgh), FORTE (Berlin);
Oct: GI (Frankfurt), ASE (Montreal), ...

Special SoSyM issue on Critical Systems
Development with UML (deadl. 15 Sep.)

CSDUML'03 @ UML'03 conference (Oct. in
SFO, deadl. 26 July)

More information (incl. slides + audio):

<http://www4.in.tum.de/~juerjens/>

