Algebraic State Machines: Concepts and Applications to Security (Extended Abstract)

Jan Jürjens*

Software & Systems Engineering, Informatics, TU Munich, Germany

Abstract. The concept of *algebraic state machine* has been introduced in [BW00] as a state transition system the states of which are each defined as an algebra, and that communicate through channels.
To make efficient use of this concept, one needs a formal semantics, as well as notions of composition and refinement, which are provided in the present work. To demonstrate their usefulness for an application area of major interest, we show how to extend algebraic state machines with data types modelling cryptographic operations and with an adversary model to reason about security-critical systems. As an example we consider a cryptographic protocol proposed in the literature.

1 Introduction

Algebraic state machines (AlgSMs) [BW00] (based on Abstract State Machines [Gur95]) are state transition systems, the states of which are represented by algebras and which can communicate through channels. It is motivated by the need to abstract away from low-level states when describing large systems by state machines. In this work, we extend this work in several aspects:

- We give a formal semantics for AlgSMs as stream-processing functions.
- We define the composition at the level of algebraic state machines. We prove that this composition satisfies useful structural properties (associativity) and that it is preserved by the interpretation as stream-processing functions (in the sense that the composition of the interpretations of two AlgSMs gives the interpretation of their composition).
- A well-known paradigm of system development is that of *stepwise refinement*: One starts with an abstract specification and refines it in several steps to a concrete specification, which is finally implemented. We define refinement of AlgSMs wrt. the interfaces defined by their communication mechanisms.

* `http://www.jurjens.de/jan` – `juerjens@in.tum.de` .

– To demonstrate their usefulness for an application area of major interest, we show how to extend AlgSMs with data types modelling cryptographic operations and with an adversary model to reason about security-critical systems. We define a notion of *secrecy*. To demonstrate usefulness of our approach, we formally analyse a security protocol proposed in the literature (a variant of the Internet protocol TLS, which is a successor of SSL), exhibit a flaw, propose a correction, and prove it secure.

For space reasons, proofs of the results as well as a discussion of related work has to be omitted from this extended abstract but can be found in the long version.[1]

2 Algebraic State Machines

The idea of connecting abstract state machines using the communication concept of [BS01] has also been pursued under the name of "interactive ASM" in [Jür02b, Jür02a, Jür03].

We fix a set $\mathcal{D}$ of data values that can be exchanged between AlgSMs. For a set C of channel names we write $C^{[*]}$ for the set of functions $v : C \to \mathcal{D}^*$ where $\mathcal{D}^*$ is the set of finite sequences in $\mathcal{D}$. Thus v assigns a finite communication history to each of the channels in C.

An *algebraic state machine (AlgSM)* is a state machine where the states are algebras and which is connected through channels with its environment through which it exchanges inputs and outputs: An AlgSM $\mathcal{A}$ is given by

– a set Alg of Σ-algebras (for a signature Σ), which is the set of states of $\mathcal{A}$, also written as State.
– sets I resp. O of input resp. output channel names,
– a state transition function $\Delta : \mathsf{Alg} \times I^{[*]} \to \mathcal{P}(\mathsf{Alg} \times O^{[*]})$,
– and a set of initial states $\mathsf{Alg}^0 \subseteq \mathsf{Alg}$.

An AlgSM is executed as follows:

– one of the initial states is chosen non-deterministically,
– the following step is iterated: the state transition function is applied to the current state and the current channel valuation; from the resulting set, a pair $(S, v) \in \mathsf{Alg} \times O^{[*]}$ is chosen non-deterministically.

[1] Available at http://www4.in.tum.de/~juerjens .

As a new contribution, we give an interpretation of AlgSMs as stream-processing functions.

For a set C of channel names we write $C^{\rightarrow}$ for the set of functions $v : C \rightarrow \mathcal{D}^*)^{\omega}$ (the valuations of the channels in C by infinite communication histories; X^{ω} is the set of infinite sequences $\boldsymbol{c} : \mathbb{N} \rightarrow X$). Given $c_0 \in C^{[*]}$ and $\boldsymbol{c} \in C^{\rightarrow}$, we write $c_0.\boldsymbol{c} \in C^{\rightarrow}$ for the channel valuation such that for each $x \in C$, $c_0.\boldsymbol{c}(x)$ is the sequence the head of which is $c_0(x)$ and the tail of which is $\boldsymbol{c}(x)$. Also, given a set C of channel names, a subset $D \subseteq C$, and a (finite or infinite) communication history $v \in C^{[*]} \cup C^{\rightarrow}$, the communication history $v\downharpoonleft_D \in D^{[*]} \cup D^{\rightarrow}$ is defined by $v\downharpoonleft_D(d) = v(d)$ for each channel $d \in D$ (the restriction of v to the channels in D).

An AlgSM $\mathcal{A}$ with sets I resp. O of input resp. output channels and state transition function $\Delta : \mathsf{Alg} \times I^{[*]} \rightarrow \mathcal{P}(\mathsf{Alg} \times O^{[*]})$ is interpreted as a function $[\![\mathcal{A}]\!] : I^{\rightarrow} \rightarrow \mathcal{P}(O^{\rightarrow})$, a *stream-processing function* from the input channels in I to the output channels in O, as follows.

We define $f_1 \leq f_2$ for $f_1, f_2 : \mathsf{Alg} \rightarrow (I^{\rightarrow} \rightarrow \mathcal{P}(O^{\rightarrow}))$ if and only if $f_1(s)(\boldsymbol{i}) \subseteq f_2(s)(\boldsymbol{i})$ for all $s \in \mathsf{Alg}$ and $\boldsymbol{i} \in I^{\rightarrow}$. Then we define $[\![\_]\!] : \mathsf{Alg} \rightarrow (I^{\rightarrow} \rightarrow \mathcal{P}(O^{\rightarrow})), s \mapsto [\![s]\!]$ to be the largest function that satisfies

$$[\![s]\!](i_0.\boldsymbol{i}) \stackrel{\text{def}}{=} \Big\{o_0.\boldsymbol{o} : \exists s' \in \mathsf{Alg}.\Big((s', o_0) \in \Delta(s, i_0) \wedge \boldsymbol{o} \in [\![s']\!](\boldsymbol{i})\Big)\Big\}$$

for all $s \in \mathsf{Alg}$ and $i_0.\boldsymbol{i} \in I^{\rightarrow}$, with respect to the following relation: Here $i_0.\boldsymbol{i}$ denotes the sequence with head i_0 and tail $\boldsymbol{i}$. Then for $\boldsymbol{i} \in I^{\rightarrow}$, we define $[\![\mathcal{A}]\!](\boldsymbol{i}) \stackrel{\text{def}}{=} \bigcup_{s \in \mathsf{Alg}^0} [\![s]\!](\boldsymbol{i})$.

3 Composition

We define (delayed) composition for two AlgSMs $\mathcal{A}_1$ and $\mathcal{A}_2$. It assumes that the communication between components incurs a time delay. The definition is new; it has as counterpart the canonical definition of composition of stream-processing functions, in a way that ensures that the mapping from AlgSMs to stream-processing functions preserves the respective compositions (in the sense that the composition of the interpretations of two AlgSMs gives the interpretation of their composition; see below). Suppose we are given two AlgSMs $\mathcal{A}_i = (\Sigma_i, \mathsf{Alg}_i, I_i, O_i, \Delta_i, {\mathsf{Alg}^0}_i)$ (for $i = 1, 2$) where $\Sigma_i = (S_i, F_i, \mathbf{fct}_i)$, and with $O_1 \cap O_2 = I_1 \cap I_2 = \emptyset$, and $l^{\mathcal{A}_1} = l^{\mathcal{A}_2}$ for the carrier sets of each channel $l \in L \stackrel{\text{def}}{=} (O_1 \cup O_2) \cap (I_1 \cup I_2)$. We define their *delayed composition* $\mathcal{A}_1 \odot \mathcal{A}_2 \stackrel{\text{def}}{=} (\Sigma, \mathsf{Alg}, I, O, \Delta, \mathsf{Alg}^0)$ as follows:

– $\Sigma \stackrel{\text{def}}{=} (S_1 \uplus S_2 \uplus S_3, F_1 \uplus F_2 \uplus F_3, \mathbf{fct}_1 \uplus \mathbf{fct}_2 \uplus \mathbf{fct}_3)$ where

- $S_3 \stackrel{\text{def}}{=} \{\mathbf{loc}_l^{\mathcal{A}_1 \odot \mathcal{A}_2} : l \in L\}$ where the sort $\mathbf{loc}_l^{\mathcal{A}_1 \odot \mathcal{A}_2}$ has the same carrier set as $l^{\mathcal{A}_1} = l^{\mathcal{A}_2}$,
- $F_3 \stackrel{\text{def}}{=} \{\mathbf{feed}_l : l \in L\}$ and
- $\mathbf{fct}_3(\mathbf{feed}_l) \stackrel{\text{def}}{=} (\mathbf{loc}_l^{\mathcal{A}_1 \odot \mathcal{A}_2})^*$ for $l \in L$.

– $\mathsf{Alg} \stackrel{\text{def}}{=} \{A_1 \uplus A_2 \uplus A_3 : A_1 \in \mathsf{Alg}_1 \wedge A_2 \in \mathsf{Alg}_2 \wedge A_3 \in \mathsf{Alg}(\Sigma_3)\}$ where $\Sigma_3 = (S_3, F_3, \mathbf{fct}_3)$.

– $I \stackrel{\text{def}}{=} (I_1 \cup I_2) \backslash (O_1 \cup O_2)$ and $O \stackrel{\text{def}}{=} (O_1 \cup O_2) \backslash (I_1 \cup I_2)$ with $c^{\mathcal{A}_1 \odot \mathcal{A}_2} = c^{\mathcal{A}_i}$ for $c \in I_i \cup O_i$ (for $i \in \{1, 2\}$).

– For $\boldsymbol{i} \in I^{[*]}$, the state transition function is defined as

$$\begin{aligned}\Delta(A_1 \uplus A_2 \uplus A_3, \boldsymbol{i}) \stackrel{\text{def}}{=} \Big\{ & (B_1 \uplus B_2 \uplus B_3, \tilde{o}|_O) : \tilde{o} \in (O_1 \cup O_2)^{[*]} \\ & \wedge \exists \tilde{i} \in (I_1 \cup I_2)^{[*]}. \Big(\boldsymbol{i} = \tilde{i}|_I \wedge (B_1, \boldsymbol{o}|_{O_1}) \in \Delta_1(A_1, \tilde{i}|_{I_1}) \\ & \wedge (B_2, \boldsymbol{o}|_{O_2}) \in \Delta_2(A_2, \tilde{i}|_{I_2}) \wedge \forall l \in L. (\tilde{i}(l) = \mathbf{feed}_l{}^{A_3} \wedge \mathbf{feed}_l{}^{B_3} = \boldsymbol{o}(l)) \Big) \Big\}.\end{aligned}$$

– $\mathsf{Alg}^0 \stackrel{\text{def}}{=} \{A_1 \uplus A_2 \uplus A_3 : A_1 \in \mathsf{Alg}^0{}_1 \wedge A_2 \in \mathsf{Alg}^0{}_2 \wedge \forall l \in L. \mathbf{feed}_l{}^{A_3} = \varepsilon\}$ where ε is the empty sequence.

Here the set of states of $\mathcal{A}_1 \odot \mathcal{A}_2$ is the cartesian product $\mathsf{State}_1 \times \mathsf{State}_2 \times \mathsf{Alg}(\Sigma_3)$ of the sets of states State_1 of $\mathcal{A}_1$ and State_2 of $\mathcal{A}_2$, and the set of Σ_3-algebras where Σ_3 consists of constants $\mathbf{feed}_l$ storing the contents of each of the local channels $l \in L$. The transitions are constructed as in the case of instantaneous composition, except that the inputs from the local channels are taken from the $\mathbf{feed}_l$ before the transition is fired and the outputs to the local channels are written to $\mathbf{feed}_l$ after the transition is fired. This introduces a delay of one time step into the composition.

Composition is associative if there is no confusion between the sorts and channels:

Theorem 1. *Suppose we are given AlgSMs $\mathcal{A}_i = (\Sigma_i, \mathsf{Alg}_i, I_i, O_i, \Delta_i, \mathsf{Alg}^0{}_i)$ (for $i = 1, 2, 3$) where $\Sigma_i = (S_i, F_i, \mathbf{fct}_i)$. Suppose that for $i, j \in \{1, 2, 3\}$ we have $O_i \cap O_j = I_i \cap I_j = \emptyset$ and $l^{\mathcal{A}_i} = l^{\mathcal{A}_j}$ for the carrier sets of each channel $l \in (O_i \cup O_j) \cap (I_i \cup I_j)$. Then $(\mathcal{A}_1 \odot \mathcal{A}_2) \odot \mathcal{A}_3 = \mathcal{A}_1 \odot (\mathcal{A}_2 \odot \mathcal{A}_3)$.*

For our result that the mapping from AlgSMs to stream-processing functions preserves composition, we first define *delayed composition* of stream-processing functions. Given stream-processing functions $f_i : I_i^{\rightarrow} \rightarrow \mathcal{P}(O_i^{\rightarrow})$ (for $i = 1, 2$) with $O_1 \cap O_2 = I_1 \cap I_2 = \emptyset$, we define the delayed composition $f_1 \odot f_2 : I^{\rightarrow} \rightarrow O^{\rightarrow}$ where $I \stackrel{\text{def}}{=} (I_1 \cup I_2) \setminus (O_1 \cup O_2)$ and

$O \stackrel{\text{def}}{=} (O_1 \cup O_2) \setminus (I_1 \cup I_2)$ as follows. For $\tilde{i} \in I^{\rightarrow}$, we define

$$f_1 \odot f_2(\tilde{i}) \stackrel{\text{def}}{=} \Big\{ \tilde{o} : \exists \boldsymbol{i} \in (I_1 \cup I_2)^{\rightarrow}, \boldsymbol{o}_1 \in f_1(\boldsymbol{i}|_{I_1}), \boldsymbol{o}_2 \in f_2(\boldsymbol{i}|_{I_2}). \\ \Big(\tilde{i} = \boldsymbol{i}|_I \ \wedge \ \tilde{o} = (\boldsymbol{o}_1 \cup \boldsymbol{o}_2)|_O \wedge \forall l \in L. \boldsymbol{i}(l) = \varepsilon.(\boldsymbol{o}_1 \cup \boldsymbol{o}_2)(l) \Big) \Big\}$$

where ε is the valuation that maps each channel to the empty sequence. Again, the implicit assumption is that producing an output from an input induces a time delay.

Theorem 2. *For all AlgSMs $\mathcal{A}_1, \mathcal{A}_2$, we have $[\![\mathcal{A}_1 \odot \mathcal{A}_2]\!] = [\![\mathcal{A}_1]\!] \odot [\![\mathcal{A}_2]\!]$.*

4 Refinement

A useful paradigm of system development is that of *stepwise refinement*: One starts with an abstract specification and refines it in several steps to a concrete specification which is implemented.

We define a notion of *refinement* that allows to proceed from abstract to more concrete specifications in a well-defined way. We also define a notion of *timeless refinement*, which is a relaxation of refinement allowing delays to be inserted. It allows a more flexible treatment, while still offering convenient structural properties.

Definition 1 (Refinement). *Suppose we are given AlgSMs $\mathcal{A}$ and $\mathcal{A}'$. We say that $\mathcal{A}'$* refines *$\mathcal{A}$ if for each input valuation $\boldsymbol{i} \in I^{\rightarrow}$, we have $[\![A']\!](\boldsymbol{i}) \subseteq [\![A]\!](\boldsymbol{i})$.*

Theorem 3. *Composition of AlgSMs and interpretation of AlgSMs as stream-processing functions is preserved by refinement.*

For timeless refinement we define the time abstraction $|\boldsymbol{c}| \in C^{\omega}$ of a stream $\boldsymbol{c} \in C^{\rightarrow}$ by $|c_0.\boldsymbol{c}| \stackrel{\text{def}}{=} c_0 \hat{\ } |\boldsymbol{c}|$ (where $\hat{\ }$ denotes concatenation of sequences). Then for any stream-processing function $f : I^{\rightarrow} \to \mathcal{P}(O^{\rightarrow})$ we define the *time abstraction* $|f| : I^{\omega} \to \mathcal{P}(O^{\omega})$ by $|f|(\boldsymbol{i}) \stackrel{\text{def}}{=} \{|f(\tilde{i})| : \tilde{i} \in I^{\rightarrow} \wedge |\tilde{i}| = \boldsymbol{i}\}$.

Definition 2 (Timeless refinement). *Suppose we are given AlgSMs $\mathcal{A}$ and $\mathcal{A}'$. We say that $\mathcal{A}'$* refines *$\mathcal{A}$* timelessly *if for each $\boldsymbol{i} \in I^{\omega}$, we have $|[\![A']\!]|(\boldsymbol{i}) \subseteq |[\![A]\!]|(\boldsymbol{i})$.*

Theorem 4. *Interpretation of AlgSMs as stream-processing functions is preserved by timeless refinement.*

5 Security

In the long version of this paper, we demonstrate how one can apply AlgSMs to security analysis.

We consider a variant of the TLS protocol proposed in [APS99]. We demonstrate that this protocol contains a flaw and propose a corrected version which can be informally specified as follows.

$$\begin{aligned}
&C \rightarrow S : \mathsf{N_i}, \mathsf{K_C}, \mathcal{S}\mathsf{ign}_{\mathsf{K_C^{-1}}}(\mathsf{C} :: \mathsf{K_C}) \\
&S \rightarrow C : \{\mathcal{S}\mathsf{ign}_{\mathsf{K_S^{-1}}}(\mathsf{k_j} :: \mathsf{N_i} :: \mathsf{K_C})\}_{\mathsf{K_C}}, \mathcal{S}\mathsf{ign}_{\mathsf{K_{CA}^{-1}}}(\mathsf{S} :: \mathsf{K_S}) \\
&C \rightarrow S : \{\mathsf{s_i}\}_{\mathsf{k_j}}
\end{aligned}$$

We use AlgSMs to prove the following theorem about the corrected version.

Theorem 5. *Suppose we are given a particular execution of the corrected protocol, a client C, and a number I with $S = S_I$ (where S_i is the server communicating with C in the ith execution round), and suppose that the server S is in its Jth execution round in the current execution when C in its Ith execution round initiates the protocol. Then this execution preserves the secrecy of $C.s_I$ against adversaries whose previous knowledge $\mathcal{K}_A^p$ fulfills the following conditions (where we use prefixing of values by C and S as in object-oriented notation to specify the instance of the client or server a value belongs to).*

- *we have*

$$\Big(\{C.s_I, K_C^{-1}, K_S^{-1}\} \cup \{S.k_j : j \geq J\} \\ \cup \{\{\mathcal{S}ign_{K_S^{-1}}(X :: C.N_I :: K_C)\}_{K_C} : X \in \mathbf{Keys}\}\Big) \cap \mathcal{K}_A^p = \emptyset,$$

- *for any $X \in \mathbf{Exp}$, $\mathcal{S}ign_{K_C^{-1}}(C :: X) \in \mathcal{K}_A^p$ implies $X = K_C$, and*
- *for any $X \in \mathbf{Exp}$, $\mathcal{S}ign_{K_{CA}^{-1}}(S :: X) \in \mathcal{K}_A^p$ implies $X = K_S$.*

More details can be found in the long version of this paper.

6 Conclusion

The general context of this work is the need to abstract away from low-level detail and to modularize system models when describing large systems. Towards this aim, [BW00] introduced the concept of *algebraic state* *machines.* To make efficient use of this concept, one needs a formal semantics, and composition and refinement, which are provided in the present work. As a major application area, we showed how to extend AlgSMs with data types modelling cryptographic operations and with an adversary to reason about security-critical systems. As an example we considered a cryptographic protocol proposed in the literature.

To conclude, the example from the application domain of security-critical systems indicates that AlgSMs are quite a flexible and expressible formal method, which allows to structure a specification in a convenient way using the concept of channels. The added concepts advance the development of the notion of AlgSMs in that they allow composition and refinement at the level of AlgSMs.

Due to the possible high degree of abstraction provided by algebraic state machines, this formalism seems to be suitable to consider larger parts of systems beyond security protocols (such as protocol contexts), as planned for future work.

Acknowledgements Helpful comments from Manfred Broy and Thomas Kuhn are gratefully acknowledged.

References

[APS99] V. Apostolopoulos, V. Peris, and D. Saha. Transport layer security: How much does it really cost ? In *Conference on Computer Communications (IEEE Infocom)*, New York, March 1999.

[BS01] M. Broy and K. Stølen. *Specification and Development of Interactive Systems.* Springer, 2001.

[BW00] M. Broy and M. Wirsing. Algebraic state machines. In T. Rus, editor, *8th International Conference on Algebraic Methodology and Software Technology (AMAST 2000)*, volume 1816 of *LNCS*. Springer, 2000.

[Gur95] Y. Gurevich. Evolving algebras 1993: Lipari guide. In E. Börger, editor, *Specification and Validation Methods*, pages 9–36. OUP, 1995.

[Jür02a] J. Jürjens. A UML statecharts semantics with message-passing. In *Symposium of Applied Computing 2002*, pages 1009–1013, Madrid, March 11–14 2002. ACM.

[Jür02b] J. Jürjens. Formal Semantics for Interacting UML subsystems. In *5th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS 2002)*, pages 29–44. IFIP, Kluwer, 2002.

[Jür03] J. Jürjens. *Secure Systems Development with UML.* Springer, 2003. In preparation.