The $\mathcal{B}$,$\mathcal{F}$,ϕ Model

John D. Ramsdell Joshua D. Guttman
Moses D. Liskov Paul D. Rowe

September 17, 2014

In this note, we describe a model patterned after the $\mathcal{B}$,$\overline{\mathcal{C}}$,ϕ model [2]. An execution of a protocol is described by a set strands. In this model, each strand is a sequence of events of five kinds, *transmissions*, *receptions*, *initializations*, *transitions*, and *observations*. Each event has a message with the exception of a transition, which has two. A transition event provides synchronization between protocol activity and state change, and an observation provides a view into the current state. These changes lead naturally to a $\mathcal{B}$,$\mathcal{F}$,ϕ model, which is motivated by the recent implementation of state semantics in CPSA 3, and is a synthesis of the various models proposed by the authors.

Much more should be here someday.

In the text that follows, an exclamation point in the margin marks important new material.

1 Strand Spaces With State

The parameters to the strand space theory with state are a set of messages ($\mathcal{M}$), and a carried by relation ($\sqsubseteq \subseteq \mathcal{M} \times \mathcal{M}$).

The set of messages $\mathcal{M}$ is often the carrier set of a message algebra. Intuitively, a message m_0 is carried by m_1 ($m_0 \sqsubseteq m_1$) if it is possible to extract m_0 from m_1.

In strand space theory, the *trace* of a strand is a linearly ordered sequence of events $e_0 \Rightarrow \cdots \Rightarrow e_{n-1}$, and an *event* is a message transmission $+m$, a reception $-m$, a state initialization $*m$, or a state observation $?m$, where $m \in \mathcal{M}$, or a state transition $!t$, where $t \in \mathcal{M} \times \mathcal{M}$. A *strand space* $\mathcal{S}$ is a map !

from a set of strands to a set of traces. We choose the set of strands to be a prefix of the natural numbers, so a strand space is finite sequence of traces. The set of strands of strand space $\mathcal{S}$ is $\mathcal{Z}(\mathcal{S}) = Dom(\mathcal{S})$.

A node names an event in a strand space. The set of *nodes* of strand space $\mathcal{S}$ is $\{(z, i) \mid z \in \mathcal{Z}(\mathcal{S}), 0 \leq i < |\mathcal{S}(z)|\}$, and the event at a node is $evt_{\mathcal{S}}(z, i) = \mathcal{S}(z)(i)$. A node is a *path node* in $\mathcal{S}$ iff the event at the node is a state initialization or a transition. The set of nodes of $\mathcal{S}$ is $\mathcal{N}(\mathcal{S})$ and the set of transition nodes of $\mathcal{S}$ is $\mathcal{N}^!(\mathcal{S})$.

A message *originates* in trace c at index i iff

1. $c(i)$ is the transmission $+m$, it is carried by m, and it is not carried by any event earlier in the trace, or

2. $c(i)$ is the initialization $*m$, it is carried by m, and it is not carried by any event earlier in the trace, or !

3. $c(i)$ is the transition event $!(m_0, m_1)$, the message is carried by m_1, and it is not carried by m_0 or any event earlier in the trace.

A message m is *non-originating* in strand space $\mathcal{S}$, written $non(\mathcal{S}, m)$, if it originates at no node. A message m *uniquely originates* in strand space $\mathcal{S}$ at node n, written $uniq(\mathcal{S}, m, n)$, if it originates at n and nowhere else.

The model of execution is a bundle. The triple $\mathcal{B} = (\mathcal{S}, \rightarrow, \rightsquigarrow)$ is a *bundle* iff it defines a finite directed acyclic graph, where the vertices are the nodes of $\mathcal{S}$, and an edge represents communication ($\rightarrow$), state passing ($\rightsquigarrow$), or strand succession ($\Rightarrow$) in $\mathcal{S}$.

For communication, if $n_0 \rightarrow n_1$, then there is a message t such that $evt_{\mathcal{S}}(n_0) = +m$ and $evt_{\mathcal{S}}(n_1) = -m$. For each reception node n_1, there is a unique transmission node n_0 with $n_0 \rightarrow n_1$.

For state passing, if $n_0 \rightsquigarrow n_1$, then !

1. n_0 and n_1 are transition nodes, and there is a message m such that $evt_{\mathcal{S}}(n_0) = !(m_0, m)$ and $evt_{\mathcal{S}}(n_1) = !(m, m_1)$, or

2. n_0 is an initialization node and n_1 is a transition node, and there is a message m such that $evt_{\mathcal{S}}(n_0) = *m$ and $evt_{\mathcal{S}}(n_1) = !(m, m_1)$, or

3. n_0 is a transition node and n_1 is an observation node, and there is a message m such that $evt_{\mathcal{S}}(n_0) = !(m_0, m)$ and $evt_{\mathcal{S}}(n_1) = ?m$, or

4. n_0 is an initialization node and n_1 is an observation node, and there is a message m such that $evt_{\mathcal{S}}(n_0) = *m$ and $evt_{\mathcal{S}}(n_1) = ?m$, or

5. n_0 is an observation node and n_1 is a transition node, and there is a message m such that $evt_{\mathcal{S}}(n_0) = ?m$ and $evt_{\mathcal{S}}(n_1) = !(m, m_1)$.

Additionally,

1. for all path or observation nodes n_0, and transition nodes n_1 and n_2, $n_0 \rightsquigarrow n_1$ and $n_0 \rightsquigarrow n_2$ implies $n_1 = n_2$, and

2. for each transition or observation node n_1, there exists a path node n_0 such that $n_0 \rightsquigarrow n_1$, and

3. for all path nodes n_0, transition nodes n_1, and observation nodes n_2, $n_0 \rightsquigarrow n_1$ and $n_0 \rightsquigarrow n_2$ implies $n_2 \rightsquigarrow n_1$.

Each acyclic graph has a transitive irreflexive relation $\prec$ on its vertices. The relation specifies the causal ordering of nodes in a bundle. A transitive irreflexive binary relation is also called a strict order.

For a bundle $\mathcal{B}$, its associated strand space will be denoted $\mathcal{S}_{\mathcal{B}}$ unless the association is clear from the context.

With the definitions of origination and bundles given here, strand spaces with state retains a key property of the original version of strand spaces.

Lemma 1. *If message m is carried by $evt_{\mathcal{S}_{\mathcal{B}}}(n)$, then m originates in $\mathcal{B}$.*

Proof. By induction on the graph of $\mathcal{B}$ and a case analysis of the events at n that carry m.

1. If n is a transmission, then either m originates at n or m is carried earlier in the strand by the definition of origination.

2. If n is a reception, then there is an earlier transmission node that carries m.

3. If n is an observation, then there is an earlier path node that carries m.

4. If n is a initialization, then either m originates at n or m is carried earlier in the strand.

5. If $evt_{\mathcal{S}_{\mathcal{B}}}(n) = !(m_0, m_1)$ and $m \sqsubseteq m_0$, then there is an earlier path node that carries m.

6. If $evt_{\mathcal{S}_\mathcal{B}}(n) = !(m_0, m_1)$ and $m \sqsubseteq m_1$, then $m \sqsubseteq m_0$, or m originates at n, or m is carried earlier in the strand. □

In the remainder of this section, the theory of strand spaces used in the proofs has been simplified. In the full theory, origination assumptions can be inherited from roles. See [3] for all the gory details.

When a bundle is a run of a protocol, the behavior of each strand is constrained by a role. Adversarial strands are constrained by roles as are non-adversarial strands. A *protocol* is a set of roles, and a *role* is a set of traces. A trace c is an *instance* of role r iff c is a prefix of some member of r. For protocol P, bundle $\mathcal{B} = (\mathcal{S}, \rightarrow, \rightsquigarrow)$ is a *run of protocol* P iff there exists a role assignment $ra \in \mathcal{Z}(\mathcal{S}) \rightarrow P$ such that for all $z \in \mathcal{Z}(\mathcal{S})$, $\mathcal{S}(z)$ is an instance of $ra(z)$.

2 State and Compatibility

The parameters to the state theory are

1. a set of states ($\mathcal{Q}$),
2. a set of labels ($\mathcal{L} \subseteq \mathcal{M}$),
3. a set of initial states ($\mathcal{I} \subseteq \mathcal{Q}$),
4. a labeled state transition relation ($\mathcal{T} \subseteq \mathcal{Q} \times \mathcal{L} \times \mathcal{Q}$), and
5. an injective state encoding function ($f \in \mathcal{Q} \rightarrow \mathcal{M}$).

Let π be a finite sequence of states $\mathcal{Q}$, and λ be a finite sequence of labels $\mathcal{L}$. The pair $\mathcal{C} = (\pi, \lambda)$ is a *computation* iff

1. $|\pi| = |\lambda| + 1$, and
2. $\pi(0) \in \mathcal{I}$, and
3. $\forall i < |\lambda|. (\pi(i), \lambda(i), \pi(i+1)) \in \mathcal{T}$.

A *computation family* is a finite sequence of computations. The set of positions in family $\mathcal{F}$ is $\mathcal{P}(\mathcal{F}) = \{(i, j) \mid i \in Dom(\mathcal{F}), (\pi, \lambda) = \mathcal{F}(i), 0 \leq j < |\lambda|\}$. For positions (i_0, j_0) and (i_1, j_1) in $\mathcal{P}(\mathcal{F})$, $(i_0, j_0) \hookrightarrow (i_1, j_1)$ iff $i_0 = i_1$ and $j_0 = j_1 + 1$.

In the $\mathcal{B}$,$\mathcal{F}$,ϕ model, each execution is a triple. The $\mathcal{B}$ refers to a bundle as described above, $\mathcal{F}$ refers to a computation family, and ϕ to a map from transition nodes to positions in the family.

We define $\mathcal{B}$,$\mathcal{F}$,ϕ to be a *compatible triple* iff

1. ϕ is a bijection between transition nodes and positions in $\mathcal{F}$,
2. ϕ preserves the strict order $\prec$, meaning that for all transition nodes n_0 and n_1, $n_0 \prec n_1$ implies $\phi(n_0) \hookrightarrow \phi(n_1)$, and
3. ϕ preserves transitions, meaning that for $(i, j) = \phi(n)$ and $(\pi, \lambda) = \mathcal{F}(i)$, $evt(n) = !(f(\pi(j), f(\pi(j+1))$. !
4. $\mathcal{T}$ has all observations, meaning for observation node n with $evt(n) = ?m$, there is some $q \in \mathcal{Q}$ and $\ell \in \mathcal{L}$ such that $(q, \ell, q) \in \mathcal{T}$ and $m = f(q)$.

One could add a requirement that the length of family $\mathcal{F}$ be the same as the number of initialization nodes in $\mathcal{B}$, but this is unnecessary. Note that the structure of bundles and the definition of a computation ensures that the state encoded by an initialization node is an initial state.

3 The $\mathcal{B}$,$\overline{\mathcal{C}}$,ϕ Model

Strand spaces with states is a natural way of adding state to strand spaces. To bundles that contain message-passing edges, it adds state-passing edges, and the rest follows. However, the state-passing model has a serious shortcoming. State and message-passing are intertwined in a way that makes it hard to reuse results on slightly different problems.

The $\mathcal{B}$,$\overline{\mathcal{C}}$,ϕ model [2] was designed to address this shortcoming. In this model, states are related using a labeled transition system as in the $\mathcal{B}$,$\mathcal{F}$,ϕ model. In fact, the $\overline{\mathcal{C}}$ in the $\mathcal{B}$,$\overline{\mathcal{C}}$,ϕ model is really a computation family $\mathcal{F}$. Variables related to ones defined by strand spaces with state will be barred with the exception of computation families. Thus, we disambiguate by writing $\overline{\mathcal{B}}$,$\overline{\mathcal{C}}$,$\overline{\phi}$ for the Guttman's model.

In the $\overline{\mathcal{B}}$,$\overline{\mathcal{C}}$,$\overline{\phi}$ model, a state synchronization event is a label. A node is a *neutral node* in $\overline{\mathcal{S}}$ iff the event at the node is a state synchronization. The set of neutral nodes of $\overline{\mathcal{S}}$ is $\bar{\mathcal{N}}^{\circ}(\overline{\mathcal{S}})$. The bundle $\overline{\mathcal{B}} = (\overline{\mathcal{S}}, \rightarrow)$ omits the state-passing edges $\rightsquigarrow$ from its associated graph along with the constraints associated with state-passing. In short, a bundle $\overline{\mathcal{B}}$ is just a strand space

bundle augmented with neutral nodes. Function $\overline{\phi}$ is a map from neutral nodes to positions in the family $\overline{\mathcal{C}}$.

We define $\overline{\mathcal{B}}$,$\overline{\mathcal{C}}$,$\overline{\phi}$ to be a *compatible triple* iff

1. $\overline{\phi}$ is a bijection between neutral nodes and positions in $\overline{\mathcal{C}}$,
2. $\overline{\phi}$ preserves the strict order $\prec$, meaning that for all neutral nodes n_0 and n_1, $n_0 \prec n_1$ implies $\overline{\phi}(n_0) \hookrightarrow \overline{\phi}(n_1)$, and
3. $\overline{\phi}$ preserves transitions, meaning that for $(i, j) = \overline{\phi}(n)$ and $(\pi, \lambda) = \overline{\mathcal{C}}(i)$, $evt(n) = \circ(\lambda(j))$.

4 Relating Models

This section needs help. Unresolved issues follow.

1. How does one reflect the definition of origination in the $\mathcal{B}$,$\mathcal{F}$,ϕ model into the $\overline{\mathcal{B}}$,$\overline{\mathcal{C}}$,$\overline{\phi}$ via constrains on labels imposed by $\mathcal{T}$?
2. What should be done about initialization nodes? $\mathcal{B}$,$\mathcal{F}$,ϕ has them, but $\overline{\mathcal{B}}$,$\overline{\mathcal{C}}$,$\overline{\phi}$ does not.
3. What should be done about observation nodes? They are not in the domain of ϕ in $\mathcal{B}$,$\mathcal{F}$,ϕ, but they are in the domain of $\overline{\phi}$ in $\overline{\mathcal{B}}$,$\overline{\mathcal{C}}$,$\overline{\phi}$.

5 Message Model

Typically, messages are modeled by elements of an order-sorted algebra [1]. An order-sorted algebra is a generalization of a many-sorted algebra in which sorts may be partially ordered. The carrier sets associated with ordered sorts are related by the subset relation.

Figure 1 shows the signature of the algebra used in examples in this paper. Sort M is the sort of all messages. Messages of sort A (asymmetric keys), sort S (symmetric keys), sort D (data), and sort E (text) are called *atoms*. Messages are atoms, tag constants, or constructed using encryption $\{\!|\cdot|\!\}_{(\cdot)}$, hashing $\#(\cdot)$, and pairing $(\cdot,\cdot)$, where the comma operation is right associative and parentheses are omitted when the context permits.

The algebra $\mathbb{A}$ is the initial quotient term algebra over the signature. The canonical representative for each element in the algebra is the term that

Sorts:	M, A, S, D, E		
Subsorts:	$\mathsf{A} < \mathsf{M}$, $\mathsf{S} < \mathsf{M}$, $\mathsf{D} < \mathsf{M}$, $\mathsf{E} < \mathsf{M}$		
Operations:	$(\cdot,\cdot)$	$: \mathsf{M} \times \mathsf{M} \to \mathsf{M}$	Pairing
	$\{\!\lvert \cdot \rvert\!\}_{(\cdot)}$	$: \mathsf{M} \times \mathsf{A} \to \mathsf{M}$	Asymmetric encryption
	$\{\!\lvert \cdot \rvert\!\}_{(\cdot)}$	$: \mathsf{M} \times \mathsf{S} \to \mathsf{M}$	Symmetric encryption
	$\#$	$: \mathsf{M} \to \mathsf{M}$	Hashing
	$(\cdot)^{-1}$	$: \mathsf{A} \to \mathsf{A}$	Asymmetric key inverse
	$(\cdot)^{-1}$	$: \mathsf{S} \to \mathsf{S}$	Symmetric key inverse
	$\mathsf{a}_i, \mathsf{b}_i$	$: \mathsf{A}$	Asymmetric key constants
	s_i	$: \mathsf{S}$	Symmetric key constants
	d_i	$: \mathsf{D}$	Data constants
	e_i	$: \mathsf{E}$	Text constants
	g_i	$: \mathsf{M}$	Tag constants
Equations:	$\mathsf{a}_i{}^{-1} = \mathsf{b}_i \quad \mathsf{b}_i{}^{-1} = \mathsf{a}_i$		$(i \in \mathbb{N})$
	$\forall k : \mathsf{A}.\, (k^{-1})^{-1} = k$		$\forall k : \mathsf{S}.\, k^{-1} = k$

Figure 1: Crypto Algebra Signature

contains no occurrences of the inverse operation $(\cdot)^{-1}$. At times, we conflate a message with its canonical representative. The carrier set $\mathbb{A}_{\mathsf{M}}$ for sort M is what is used to instantiate $\mathcal{M}$ in strand spaces with state. For sort S in the signature, we write $m : S$ for $m \in \mathbb{A}_S$.

A message m_0 is *carried by* m_1, written $m_0 \sqsubseteq m_1$ iff m_0 can be extracted from a reception of m_1, assuming plaintext is extractable from encryptions. In other words, $\sqsubseteq$ is the smallest reflexive, transitive relation such that $m_0 \sqsubseteq m_0$, $m_0 \sqsubseteq (m_0, m_1)$, $m_1 \sqsubseteq (m_0, m_1)$, and $m_0 \sqsubseteq \{\!\lvert m_0 \rvert\!\}_{m_1}$.

The roles that constrain adversarial behavior are defined by the functions

$$
\begin{aligned}
\mathit{create}(m : \mathsf{A}|\mathsf{S}|\mathsf{D}|\mathsf{E}) &= +m \qquad \mathit{tag}_i = +\mathsf{g}_i \\
\mathit{pair}(m_0 : \mathsf{M}, m_1 : \mathsf{M}) &= -m_0 \Rightarrow -m_1 \Rightarrow +(m_0, m_1) \\
\mathit{sep}(m_0 : \mathsf{M}, m_1 : \mathsf{M}) &= -(m_0, m_1) \Rightarrow +m_0 \Rightarrow +m_1 \\
\mathit{enc}(m : \mathsf{M}, k : \mathsf{A}|\mathsf{S}) &= -m \Rightarrow -k \Rightarrow +\{\!\lvert m \rvert\!\}_k \\
\mathit{dec}(m : \mathsf{M}, k : \mathsf{A}|\mathsf{S}) &= -\{\!\lvert m \rvert\!\}_k \Rightarrow -k^{-1} \Rightarrow +m \\
\mathit{hash}(m : \mathsf{M}) &= -m \Rightarrow +\#m
\end{aligned}
$$

Figure 2: Adversary Traces

in Figure 2. The role defined by the function is all the traces that it generates. For example, the role associated with the function *pair* is $\{pair(m_0, m_1) \mid m_0, m_1 : \mathsf{M}\}$. For the encryption related roles, $k : \mathsf{A}|\mathsf{S}$ asserts that k is either a symmetric or asymmetric key. For the create role, $m : \mathsf{A}|\mathsf{S}|\mathsf{D}|\mathsf{E}$ asserts that m is an atom.

An atom m is *penetrator non-originating* in bundle $\mathcal{B}$ if there is no strand in $\mathcal{B}$ with a trace that begins with $+m$. The assumption has the effect of prohibiting the use of the create role for the atom by the adversary.

6 Wrap-Decrypt Protocol

The state in the Wrap-Decrypt Protocol is a device that creates, stores, and shields symmetric keys. The device offers two operations using the keys it stores. It can encrypt a key using a key in the store, called wrapping, or it can decrypt a message using a key. A goal of this device is that all of its keys remain shielded within it. A key could be leaked if a key is used to wrap itself, and then the wrapped key is decrypted.

The device enforces its security policy by associating an attribute with each of its keys. A key has one of three attributes, init, wrap, and decrypt. A key is created with attribute init, wrapping is allowed when a key has attribute wrap, and decrypting is allowed when a key has attribute decrypt.

Attributes can be changed with the set wrap and set decrypt operation. The device policy is that set wrap succeeds as long as key's attribute is not decrypt and set decryt succeeds as long as key's attribute is not wrap. The remaining available operation of the device is key making.

Let $A = \{\mathsf{init}, \mathsf{wrap}, \mathsf{decrypt}\}$. The set of states $\mathcal{Q} = \mathbb{A}_\mathsf{S} \times A$. The state encoding function $f(k, a) = (\#k, f_A(a))$, where f_A maps each attribute to a distinct tag. We use the attribute symbol to name the tag, so $f(k, \mathsf{init}) = (\#k, \mathsf{init})$. The initial states $\mathcal{I} = \{(k, \mathsf{init}) \mid k \in \mathbb{A}_\mathsf{S}\}$.

Figure 3 displays the Wrap-Decrypt Protocol traces. In the full version of strand spaces [3], origination assumptions can be inherited from roles. This feature is used in the Wrap-Decrypt Protocol. Every instantiation of the *make* role adds the assumption that the hash of the key uniquely originates at the first node of a *make* strand. Additionally, the role adds the assumption that the key is penetrator non-originating.

An unlabeled transition system compatible with the Wrap-Decrypt Pro-

$$
\begin{aligned}
make(q : \mathsf{S}) &= *(\#k, \mathsf{init}) \Rightarrow \#k \\
setwrapi(k : \mathsf{S}) &= !((\#k, \mathsf{init}), (\#k, \mathsf{wrap})) \\
setwrapw(k : \mathsf{S}) &= !((\#k, \mathsf{wrap}), (\#k, \mathsf{wrap})) \\
setdecrypti(k : \mathsf{S}) &= !((\#k, \mathsf{init}), (\#k, \mathsf{decrypt})) \\
setdecryptw(k : \mathsf{S}) &= !((\#k, \mathsf{decrypt}), (\#k, \mathsf{decrypt})) \\
wrap(k_0, k_1 : \mathsf{S}) &= -\#k_0 \Rightarrow -\#k_1 \Rightarrow ?(\#k_1, \mathsf{wrap}) \Rightarrow +\{\!|k_0|\!\}_{k_1} \\
decrypt(m : \mathsf{M}, k : \mathsf{S}) &= -\{\!|m|\!\}_k \Rightarrow -\#k \Rightarrow ?(\#k, \mathsf{decrypt}) \Rightarrow +m
\end{aligned}
$$

Figure 3: Wrap-Decrypt Traces

tocol traces follows. For $\overline{\mathcal{T}} \subseteq \mathcal{Q} \times \mathcal{Q}$,

$$((k_0, a_0), (k_1, a_1)) \in \overline{\mathcal{T}} \text{ iff } k_0 = k_1 \wedge (a_0 = a_1 \vee a_0 = \mathsf{init}).$$

At this point, labels should be added, and then used to link this version of the protocol to one with neutral nodes.

7 Discussion

The definition of a bundle $\mathcal{B}$ is motivated by the recent implementation of state semantics in CPSA 3. In particular, the inclusion of state transition events and node orderings implied by initialization, transition, and observation nodes is new. In this model, the treatment of observations is greatly simplified, and is not part of the state component of the model. The state component need only focus on state transitions. The details of state are abstracted away by the encoding function f.

This model of strand spaces with state is much easier to specify in PVS. The contents of a transition event need not be in the transition relation, so one does not need to use subsets of the transition relation or the like.

References

[1] Joseph A. Goguen and José Meseguer. Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial

operations. *Theoretical Computer Science*, 105(2):217–273, 1992.

[2] Joshua D. Guttman. State and progress in strand spaces: Proving fair exchange. *Journal of Automated Reasoning*, 48(2):159–195, 2012.

[3] John D. Ramsdell. Proving security goals with shape analysis sentences. Technical Report MTR130488, The MITRE Corporation, September 2013. `http://arxiv.org/abs/1403.3563`.