Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 242 (2009) 139–159

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Adding Branching to the Strand Space Model

Sibylle Fröschle[1]

Department für Informatik
Universität Oldenburg
Oldenburg, Germany

Abstract

The strand space model is one of the most successful and widely used formalisms for analysing security protocols. This might seem surprising given that the model is not able to reflect choice points in a protocol execution: the key concept in the strand space model is that of a bundle, which models exactly one possible execution of a security protocol. Inspired by the branching processes of Petri nets, we show that branching can be introduced into the strand space model in a very natural way: bundles can be generalized to branching bundles, which are able to capture several conflicting protocol executions. Our investigations of the theory of branching bundles will motivate the concept of symbolic branching bundles, and culminate in the result that every protocol has a strand space semantics in terms of a largest symbolic branching bundle. We hope our results provide a strong theoretical basis for comparing models and providing process calculi semantics in security protocol analysis. Altogether our work is related but different to a series of works by Crazzolara and Winskel. Throughout we will profit from a close relationship of the strand space model to event structures, which has already been pointed out by these authors.

Keywords: Models for security protocol analysis, strand spaces, event structures, branching processes

1 Introduction

The strand space model [6] is one of the most successful and widely used formalisms for analysing security protocols. For example, it has been employed to verify security properties by hand, to give formal semantics to protocol logics, and as the underlying model of model-checking tools (c.f. [8]). In spite of this success two points of criticism have been brought against it: one is that, in contrast to models based on multiset rewriting, it is an ad hoc model rather than rooted in a rich theory. The second is that it is not able to reflect aspects of branching such as choice points in a protocol execution. To explain the latter we recall that the central concept of the strand space model is that of the *bundle*. A bundle models exactly one snapshot of a protocol execution: a set of strands represents the sessions that have occurred so far while a relation between the send and receive events of the strands describes how messages have flowed between them.

[1] Email: froeschle@informatik.uni-oldenburg.de

doi:10.1016/j.entcs.2009.06.017

Both points of criticism have been countered by results of Crazzolara and Winksel [2,1,3]. On the one hand, they have shown that the strand space model is closely related to event-based models for concurrency such as event structures. On the other hand, aiming to obtain compositional event-based semantics for protocol languages, they have shown how the strand space model can be extended by a notion of conflict [3]. Their notion of conflict is introduced at the level of strand spaces, which are conceptually a level lower than bundles: a strand space fixes all the sessions that are to be considered in the protocol analysis; it is the space from which bundles are 'carved out'.

In this paper our thesis is that branching can be introduced into the strand space model very directly at the conceptual level of bundles. The idea is to generalize bundles to *branching bundles* in the same way as in Petri net theory branching processes generalize Petri net processes. Petri net branching processes [5] are introduced as a formalization of an initial part of a run of a Petri net, which can include conflicting choices. They come with a very satisfying theory: the branching processes of a Petri net form a complete lattice (modulo isomorphism) with respect to a natural notion of approximation. The largest element of this lattice captures all possible behaviour of the Petri net, and is called its unfolding.

In this paper we wish to investigate whether protocols have as satisfying a theory of branching bundles. If every protocol P had indeed a largest branching bundle, this branching bundle would capture all possible behaviour of P, and would thus provide a natural strand space semantics. This could provide a strong theoretical basis for comparing models and providing semantics for protocol languages. Our contributions are as follows:

(1) We show that bundles can indeed be generalized to *branching bundles* in a very natural way. Branching bundles are able to capture several conflicting protocol executions.

(2) We investigate the theory of branching bundles. We find that every branching bundle can be viewed as a labelled prime event structure. This will motivate a notion of computation state for branching bundles in terms of sub-bundles, and a transition relation between them. Following the approach of [5] we will investigate whether the branching bundles of a protocol form a complete lattice with respect to a natural notion of approximation. We will however obtain a negative result: the branching bundles of a protocol do not even form a complete partial order.

(3) By analysing this negative result we will, however, be led to a notion of *symbolic* branching bundle. We will obtain that the symbolic branching bundles of a protocol indeed form a complete lattice (modulo isomorphism). Thus, every protocol can be given a strand space semantics in terms of a largest symbolic branching bundle. We will motivate that this semantics is suitable for most situations in security protocol analysis.

In the following section we will introduce the strand space model using variations on the original definition of [2] and [7]. The remainder of the paper is structured according to the above contributions. Proofs that are straightforward have been moved to Appendix A.

2 The Strand Space Model

Graph Terminology
A *labelled (directed) graph* is a tuple (E, K, L, l) where E is the set of nodes, which in our context will depict *events*, $K \subseteq E \times E$ is the set of edges, L is the set of labels, and $l : E \to L$ is a labelling function that assigns a label to every node. When L is clear from the context we will often keep it implicit for notational simplicity.

A labelled graph (E, K, L, l) is *totally ordered* if there is a total ordering $e_1 e_2 \dots$ of the elements of E such that $(e_i, e_j) \in K$ if and only if $j = i + 1$. A labelled graph (E, K, L, l) is a *labelled tree* if K is acyclic and there is no backwards branching, i.e., if $(e', e) \in K$ and $(e'', e) \in K$ then $e' = e''$. A *branch* of a tree is a possibly empty, finite or infinite sequence $e_1 e_2 \dots$ of elements of E such that $(e_i, e_{i+1}) \in K$ for all indices i. A *labelled bi-graph* is a tuple $(E, \Rightarrow, \rightarrow, L, l)$ such that both $(E, \Rightarrow, L, l)$ and $(E, \rightarrow, L, l)$ are labelled graphs.

Message Algebra
In the following we assume that messages are modelled by a message algebra. The results we present here are independent of the actual structure of this algebra. We only assume a set of messages *Mesg*, a set of atomic messages *AMesg*, from which *Mesg* is built up from, and a binary relation $\sqsubseteq$ on *Mesg* that says when one message is contained in another. Messages and atomic messages may contain variables. A message is ground if it does not contain any variables. We denote the set of ground messages by *GMesg*.

Actions, Roles, and Protocols
In a protocol execution, principals can either send or receive messages. If a message is sent then it can contain data that have just been freshly generated such as nonces. This gives rise to the following set of *actions*:

$$\begin{aligned} Act = \{ + \, \mathit{fresh}\ \mathcal{N}\ \mathit{in}\ M \mid M \in \mathit{Mesg}\ \&\ \mathcal{N} \subseteq \mathit{AMesg}\ \&\ \forall N \in \mathcal{N}, N \sqsubseteq M \} \\ \cup \{ - M \mid M \in \mathit{Mesg} \}. \end{aligned}$$

In an action of the form '$+$ *fresh* $\mathcal{N}$ *in* M', '+' indicates that message M is thought to be sent while $\mathcal{N}$ specifies which elements of M are thought to be freshly generated. We assume that only atomic messages can be freshly generated. In an action of the form '$-\ M$', '$-$' indicates that message M is thought to be received. Given an action $A \in Act$ of either of the two forms we use $\mathit{mesg}(A)$ to depict M, $\mathit{sign}(A)$ to depict '+', or '$-$' respectively. If $\mathit{sign}(A) = +$ we will further use $\mathit{fresh}(A)$ to depict $\mathcal{N}$. A *ground action* is an action that does not contain any variables. We denote the set of ground actions by *GAct*. In the context of a labelled graph with label set *GAct* we will carry over the previous concepts for actions to the events of the graph in the obvious way. A *trace* is a finite sequence of ground actions.

A *role* defines the actions a principal can perform in a protocol session. Formally, a role is a finite sequence of actions $R = A_1 \dots A_n$ such that

R1 for all $i \in [1, n]$, for all $N \in \mathit{fresh}(A_i)$
  (a) N is a variable, and
  (b) A_i is the first action that contains N: $\forall j < i$, $N \not\sqsubseteq \mathit{mesg}(A_j)$.

Axiom (R1) makes sure that we cannot specify a constant to be freshly generated, and that variables that represent data to be freshly generated at some action cannot occur in previous actions.

A *protocol* is a finite set of *roles* $P = \{R_i\}_{i \in r}$ where $r \in \mathbb{N}$.

Intruder Model

The power of the Intruder is typically modelled by two ingredients: the set of messages initially known to the Intruder such as all public keys and his own private key; and a set of *Intruder roles*, which specify the Intruder's basic elements of attack such as decrypting a message with a key that he has already obtained. (Intruder roles are originally called *parametric Intruder traces* [6].) Similarly to protocol roles, Intruder roles are essentially sequences of signed messages, where '+' denotes output and '−' denotes input.

The results here are independent of the actual format of the Intruder roles. We only assume that an *Intruder theory* is given as a pair $I = (\mathcal{K}_I, \mathcal{R}_I)$ where $\mathcal{K}_I \subseteq \mathit{GMesg}$ is the set of *initial Intruder knowledge* and $\mathcal{R}$ is the set of *Intruder roles*, and that each Intruder role is a finite sequence of actions of the following form:

$$\mathit{IAct} = \{\, +\, M \mid M \in \mathit{Mesg}\} \,\cup\, \{\, -\, M \mid M \in \mathit{Mesg}\}$$
$$\cup\, \{\, +\, M \ \mathit{of}\ \mathit{I\text{-}Knowledge} \mid M \in \mathit{Mesg}\}.$$

We redefine the set of actions Act defined in the previous paragraph to include actions of this form: $\mathit{Act} := \mathit{Act} \cup \mathit{IAct}$.

Strands, Strand Spaces, and Bundles

We now come to the core notions of the strand space model: *strands* and *bundles*. We define these concepts relative to a fixed protocol P.

A *strand* represents an instantiation of a protocol or Intruder role or of a prefix thereof. (We admit prefixes to be able to model incomplete protocol or Intruder sessions, a situation that naturally arises in a snaphot of a protocol execution.) Formally, a *strand* of P is a *totally ordered* labelled graph $s = (E, \Rightarrow, \mathit{GAct}, l)$ such that there is a prefix R of a role of P or $\mathcal{R}_I$ and a ground substitution σ so that, assuming

- $E = \{e_1, \dots, e_n\}$ with $e_1 \Rightarrow \dots \Rightarrow e_n$, and
- $R = A_1 \dots A_m$,

we have

S1 $l(e_1) \dots l(e_n) = A_1\sigma \dots A_m\sigma$, S2 $\forall i \in [1, n]$, if $\mathit{sign}(e_i) = +$ and $n \in \mathit{fresh}(e_i)$ then for all $j < i$, $n \not\sqsubseteq \mathit{mesg}(e_j)$,

S3 $\forall e \in E$, if $l(e)$ is of the form '$+\ m\ \mathit{of}\ \ \mathit{I\text{-}Knowledge}$' then $m \in \mathcal{K}_I$.

Observe how the axioms ensure that s can indeed be understood as an instantiation of R via σ. We call E the set of *events of* s, denoted by $\mathit{events}(s)$. If an event e has sign '+', we call it a *send event*, and if it has sign '−', a *receive event* respectively. We say message m *originates on event* e_i if e_i is a send event, $m \sqsubseteq \mathit{mesg}(e_i)$, and for all $j < i$, $m \not\sqsubseteq \mathit{mesg}(e_j)$. Note that Axiom (S2) ensures that when an atomic message is freshly generated at an event then it originates on that event. We call $l(e_1) \ldots l(e_n)$ the *trace* of strand s. We say two strands are *disjoint* if their sets of events are disjoint.

A snapshot of a protocol execution consists of the set of (complete and incomplete) protocol and Intruder sessions that have been executed so far plus information on how the messages flow between the sessions. This leads us to the concept of *strand space.*[2] A *strand space* of P is a pair $B = (S, \rightarrow)$ where S is a set of pairwise disjoint strands of P, and $\rightarrow \subseteq E \times E$ is a relation on the *events* of S, $E = \bigcup_{s \in S} \mathit{events}(s)$. The single-arrow relation is thought to represent the flow of messages. It is clear that we can equivalently regard B as a labelled bi-graph $(E, \Rightarrow, \rightarrow, \mathit{GAct}, l)$, a view we will often adopt. We call E the set of *events of* B, denoted by $\mathit{events}(B)$.

A strand space can contain situations that are counter-intuitive such as a receive event leading to a send event. A snapshot of a protocol execution is modelled by a *bundle.* Formally, a *bundle* of P is a strand space $B = (E, \Rightarrow, \rightarrow, \mathit{GAct}, l)$ of P such that the following axioms are satisfied:

B1 if $e_1 \rightarrow e_2$ then $\mathit{sign}(e_1) = +$, $\mathit{sign}(e_2) = -$, and $\mathit{mesg}(e_1) = \mathit{mesg}(e_2)$,

B2 if $e_1 \rightarrow e_2$ then there is no other e'_1 such that $e'_1 \rightarrow e_2$,

B3 $\forall e \in E$, if $\mathit{sign}(e) = -$ then there is $e' \in E$ such that $e' \rightarrow e$,

B4 the relation $(\rightarrow \cup \Rightarrow)$ is acyclic,

B5 $\forall e \in E$, $\{e' \mid e' \ (\rightarrow \cup \Rightarrow)^* \ e\}$ is finite,

B6 $\forall e \in E$, if $\mathit{sign}(e) = +$ and $n \in \mathit{fresh}(e)$ we have: n is *uniquely originating* on e: there is no event e' with $e' \neq e$ such that n originates on e'.

Axiom (B1) ensures that messages flow from send events to receive events. Axiom (B2) enforces that an event can receive its message from at most one event. Axiom (B3) guarantees that each receive event is matched up with a send event. Axiom (B4) ensures that the reflexive and transitive closure of $\rightarrow \cup \Rightarrow$ is a partial order, which, as we will explain below, captures *causality.* Axiom (B5) ensures that every event depends on only finitely many previous events. It is necessary in our setting since we allow bundles to contain infinitely many events. Axiom (B6) ensures that if an atomic message is specified to be freshly generated on some event then on any other strand it has to be received before it can be sent.

We denote the relation $\rightarrow \cup \Rightarrow$ by $\prec_1$. $\prec_1$ expresses *immediate causality*: If $e \rightarrow e'$ then e is an immediate cause of e' due to the message flow causality between

[2] This notion slightly varies from the standard notion of strand space related to in the introduction.

received messages and sent messages. If $e \Rightarrow e'$ then e is an immediate cause of e' due to the execution order causality within a protocol session. The reflexive and transitive closure of $\prec_1$, denoted by $\preceq$, is a partial order, which captures *causality.*

We define the *(causal) depth* of event e, written $depth(e)$, to be 1 if e is minimal with respect to $\preceq$, and $1 + \max\{depth(e') \mid e' \prec_1 e\}$ otherwise.

For every event e of a bundle there is at most one event e' such that $e' \Rightarrow e$, and at most one event e'' such that $e'' \rightarrow e$. If the first exists define $\Rightarrow$-$pred(e) = e'$ otherwise define $\Rightarrow$-$pred(e) = nil$. If the latter exists define $\rightarrow$-$pred(e) = e''$, and $\rightarrow$-$pred(e) = nil$ otherwise. Naturally we assume $nil \notin E$.

3 Branching Bundles

We now define our concept of *branching bundles.* As motivated in the introduction branching bundles should be capable of representing several conflicting protocol executions. To obtain them as a natural generalization of bundles, we will define them as bi-graphs of events labelled by actions of $GAct$. In contrast to bundles we will allow them to contain events that represent conflicting points in a protocol execution. We can distinguish between three situations when two events e_1 and e_2 of a protocol execution should naturally be considered to be in conflict with each other:

(i) e_1 and e_2 belong to different futures of the same session;

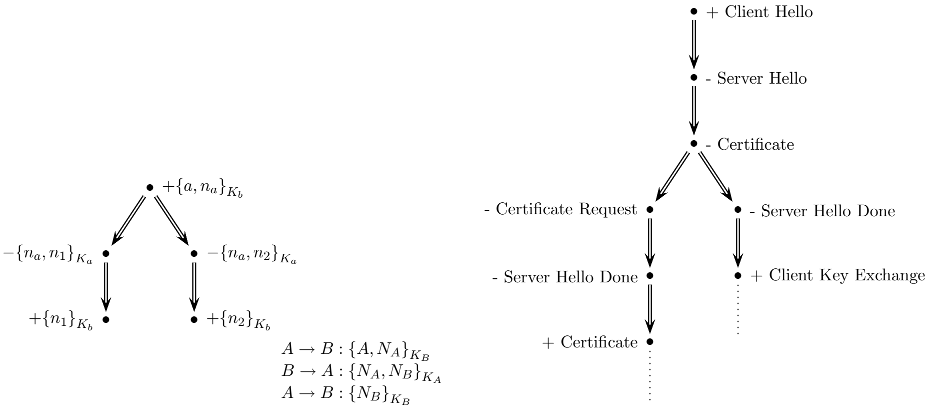
(ii) one of e_1 and e_2, say e_1, is a send event that sends as part of its message a freshly generated atomic message n, while the other event, e_2, contradicts unique origination of n: e_2 sends n as part of its message but n has never been received earlier in the session of e_2.

(iii) e_1 and e_2 are causally dependent on two events that are in conflict according to (i) or (ii).

The first situation motivates that a concept of branching bundles must be based on a concept of *branching strands.* So let us analyse in turn what sources of branching there are within a session. When does a session split into different futures?

(i) The receive actions of a protocol specification typically contain variables to be bound to parts of the incoming message. A session with such a receive action will have different futures depending on the received message. (The different futures will, however, be equivalent modulo the value that is bound to the respective input variable.) This situation is depicted in Figure 1(a).

(ii) The protocol specification may contain choice points. For example, the course of the SSL/TLS handshake protocol depends on which method for establishing the pre-master-secret is negotiated at the start, and on whether client authentication is requested by the Server or not. Typically the choice between several options of a protocol will be resolved by received input. On the other hand, to abstract away from detail, we may allow protocols to contain nondeterministic choice. The first situation is depicted in Figure 1(b).



(a) (b)

Fig. 1. Sources of branching within a session

There is yet another source of branching if we take a purely observational view. Say Eavesdropper Eve observes the first two actions of a session, but she cannot tell to which role these actions belong to: according to their format the actions could form an initial part of an instance of role A or of role B. Then in one future of Eve the *observed session* may evolve into an instance of role A, whereas in another future of Eve it will evolve into an instance of role B.

(iii) An *observed* session may have different futures due to ambiguity in the protocol specification.

To include (iii) as a source of branching is a design decision and may seem counter-intuitive at first. To include it seems, however, in the spirit of the strand space model: it is consistent with the fact that if there is ambiguity in the protocol specification then a strand may be interpreted as an instance of several roles. To resolve this type of ambiguity one would need to keep a role identifier at each strand, which would make the model less abstract and technically cumbersome. As we will now see our decision to include (iii) leads to a very simple formalization of branching strand. In the following, let P be a protocol.

Definition 3.1 A *branching strand* of P, abbreviated *b-strand* of P, is a labelled tree $s = (E, \Rightarrow, GAct, l)$ such that all branches of s are strands of P. (Note that this implies that branching strands are trees of finite depth.)

By definition every strand is a b-strand, and every b-strand whose events are totally ordered is a strand. We carry over all the concepts defined for strands in Section 2 in the obvious way. The notation $\Rightarrow$-$pred(e)$ naturally also carries over.

Having defined a notion of b-strands we obtain b-strand spaces in the obvious way. A *b-strand space* of P is a pair $B = (S, \rightarrow)$ where S is a set of disjoint b-strands of P, and $\rightarrow \subseteq E \times E$ is a relation on the events of S, $E = \bigcup_{s \in S} events(s)$. Analogously to strand spaces, we will often view B as a labelled bi-graph $(E, \Rightarrow, \rightarrow,$

$GAct, l)$. The three situations of conflict pinpointed in the beginning of the section give rise to a binary conflict relation on the events of a b-strand space.

Definition 3.2 Let $B = (E, \Rightarrow, \rightarrow, GAct, l)$ be a b-strand space of P. Two *distinct* events e_1, $e_2 \in E$ are *in immediate conflict*, written $e_1 \#_1 e_2$, if

(i) $\Rightarrow\text{-}pred(e_1) = \Rightarrow\text{-}pred(e_2)$, or

(ii) $sign(e_1) = +$, and there is $n \in fresh(e_1)$ such that n originates on e_2, or

(iii) the symmetric condition holds.

Two events e_1, $e_2 \in E$ are *in conflict*, written $e_1 \#_B e_2$, if there exist distinct events e'_1, $e'_2 \in E$ such that $e'_1 \#_1 e'_2$ and $e'_i \ (\Rightarrow \cup \rightarrow)^* \ e_i$ for $i = 1, 2$. For $e \in E$, we say e is *in self-conflict* if $e \#_B e$.

To be able to interpret a b-strand space as a branching protocol execution we need to impose axioms. Naturally we will adopt Axioms (B1) to (B5) of the definition of bundles. However, Axiom (B6) will be dropped: the axiom on unique origination is clearly not needed when events are allowed to be in conflict. On the other hand, in the presence of conflict a new axiom will be required: we need to make sure that events are never in conflict with any of the events they are causally dependent on. Formally, this gives rise to the following definition.

Definition 3.3 A *branching bundle* of P, abbreviated *b-bundle* of P, is a b-strand space B of P such that Axioms (B1) to (B5) as well as the following axiom hold:

BB No event of B is in self-conflict.

Analogously to bundles, due to Axiom (B4), we can associate a causality relation $\preceq_B$ with each b-bundle B; we carry over all concepts related to $\preceq_B$ from bundles to b-bundles. Due to Axiom (BB), for every b-bundle B, $\#_B$ is irreflexive as well as symmetric; this is what one would expect of a binary conflict relation.

Finally, we show that bundles are exactly those b-bundles where no events are in conflict. This illustrates that b-bundles are indeed the generalization of bundles we have been looking for.

Proposition 3.4 *For every b-strand space B of P, B is a bundle of P if and only if B is a b-bundle with $\#_B = \emptyset$.*

4 Towards a Theory of Branching Bundles

We now investigate whether b-bundles have as satisfying a theory as Petri net branching processes. In Section 4.2 we examine the relationship of b-bundles to event structures. In Section 4.3 we explore whether the b-bundles of a protocol approximate (modulo isomorphism) a largest b-bundle. In preparation, we introduce a notion of sub-b-bundle in Section 4.1, which is analogous to that of Petri net sub-b-processes [5].

In the following, we work as usual relative to a fixed protocol P. Given a b-bundle B of P we will implicitly assume $B = (E, \Rightarrow, \rightarrow, GAct, l)$. We carry this convention over to b-bundles B_1, B_2, and B' in the obvious way; e.g., we assume $B_1 = (E_1, \Rightarrow_1, \rightarrow_1, GAct, l_1)$.

B-bundles come with a notion of *isomorphism* induced by the standard notion for labelled bi-graphs. As usual the relation 'isomorphic' is an equivalence relation. Next to isomorphism a notion of *homomorphism* for b-bundles will be central. A homomorphism h from b-bundle B_1 to b-bundle B_2 formalizes the fact that B_1 can be folded onto a part of B_2. Given an event e of a b-bundle B, we define $\Downarrow e = \{e' \in E \mid e' \; (\Rightarrow)^* \; e\}$.

Definition 4.1 Let B_1 and B_2 be two b-bundles of P. A *homomorphism from B_1 to B_2* is a mapping h from E_1 to E_2 such that

(i) for every $e \in E_1$, $l_1(e) = l_2(h(e))$,

(ii) for every $e \in E_1$, the restriction of h to $\Downarrow_1 e$ is a bijection between $\Downarrow_1 e$ and $\Downarrow_2 h(e)$, and

(iii) for every $e, e' \in E_1$, if $e \rightarrow_1 e'$ then $h(e) \rightarrow_2 h(e')$.

It is easy to show that the composition of two homomorphisms is a homomorphism. If a homomorphism is bijective then the converse of (iii) is also true (using the fact that B_1 and B_2 are b-bundles). Thus, an isomorphism is a bijective homomorphism.

4.1 Sub-b-bundles

We introduce a natural notion of *sub-b-bundle*, which formalizes when a b-bundle is an initial part of another b-bundle.

Definition 4.2 Let B and B' be two b-bundles of P. B' is a *sub-b-bundle of B* if $E' \subseteq E$ and the identity on E' is a homomorphism from B' to B. If B' is a bundle we also say that B' is a *sub-bundle* of B.

In other words, B' is a sub-b-bundle of B if, $E' \subseteq E$, and for every $e \in E'$, (1) $l'(e) = l(e)$, (2) $\Rightarrow'\text{-}pred(e) = \Rightarrow\text{-}pred(e)$, and (3) $\rightarrow'\text{-}pred(e) = \rightarrow\text{-}pred(e)$ (using the fact that B and B' are b-bundles). This shows that B' really is an initial part of B.

We provide a characterization of the sub-b-bundles and sub-bundles of a b-bundle B in terms of *downwards-closed* subsets of E. This will further illustrate the concept of sub-b-bundle but will also be needed in the next section. A subset E' of E is *downwards-closed* if, for all $e_1, e_2 \in E$, if $e_1 \preceq_B e_2$ and $e_2 \in E'$ then $e_1 \in E'$. If B' is a sub-b-bundle of B then E' is clearly downwards-closed. This follows from the observation of the previous paragraph. On the other hand, every downwards-closed set of events determines a sub-b-bundle in a natural way.

Definition 4.3 Let B be a b-bundle, and let E' be a downwards-closed subset of E. The *sub-b-bundle* associated with E', denoted by $sbb(E')$ is defined as $(E', \Rightarrow', \rightarrow', GAct, l')$, where $\Rightarrow'$, $\rightarrow'$, and l' are obtained as the restriction of $\Rightarrow$, $\rightarrow$, and l respectively, to the events in E'.

It is immediate that $sbb(E')$ is indeed a b-bundle. It is clearly a sub-b-bundle by definition. We are now ready to state the characterization.

Proposition 4.4 *Let B be a b-bundle.*

(i) *A b-bundle B' is a sub-b-bundle of B if and only if $B' = sbb(E')$ for some downwards-closed subset E' of E.*

(ii) *A bundle B' is a sub-bundle of B if and only if $B' = sbb(E')$ for some downwards-closed* and conflict-free *subset E' of E.*

4.2 Branching Bundles and Event Structures

A *(labelled prime) event structure* is a tuple $(E, \leq, \#, L, l)$ consisting of a set E of *events*, which are partially ordered by $\leq$, the *causal dependency relation*, a binary, symmetric and irreflexive relation $\# \subseteq E \times E$, the *conflict relation*, a set L of *labels*, and a *labelling function* $l : E \to L$, which assigns a label to each event. Further, the following conditions must be satisfied for all $e, e', e'' \in E$:

E1 $e\!\downarrow = \{e' \mid e' \leq e\}$ is finite,

E2 if $e \# e'$ and $e' \leq e''$ then $e \# e''$.

Axiom (E1) means we only consider discrete processes where an event occurrence depends on finitely many previous events. Axiom (E2) makes sure that each event inherits conflict from the events it is causally dependent on.

Event structures come equipped with a notion of computation state, called *configuration*, and a transition relation between configurations. A *configuration* of an event structure $(E, \leq, \#, L, l)$ is a set $X \subseteq E$, which is

(i) downwards-closed: $\forall e, e' \in E : e' \leq e \;\&\; e \in X \Rightarrow e' \in X$, and

(ii) conflict-free: $\forall e, e' \in X : \neg(e \# e')$.

For two configurations X, X' and an event e we write $X \xrightarrow{l(e)} X'$ when $e \notin X$ and $X' = X \cup \{e\}$. In this way every event structure gives rise to a labelled transition system.

We shall now see that b-bundles are closely related to event structures. The following is straightforward:

(i) Every b-bundle B of P can be viewed as an event structure. This event structure gives a more abstract representation of B in that it abstracts away from the distribution of events over b-strands.

Proposition 4.5 *Let $B = (E, \Rightarrow, \to, GAct, l)$ be a b-bundle of P. Then $bb2ev(B) := (E, \preceq_B, \#_B, GAct, l)$ is an event structure.*

Just as the configurations of an event structure define its computation states, the sub-bundles of a b-bundle can be considered to define the reachable states of that part of the protocol execution described by the b-bundle. From Section 4.1 we know that the sub-bundles of a b-bundle can be captured in terms of conflict-free and downwards-closed subsets of events. Hence, we obtain:

(ii) There is a one-to-one correspondence between the sub-bundles of B and the configurations of $bb2ev(B)$, given by:

Proposition 4.6 *Let B be a b-bundle of P.*

(i) *If B' is a sub-bundle of B then E' (the set of events of B') is a configuration of $bb2ev(B)$.*

(ii) *If E' is a configuration of $bb2ev(B)$ then $sbb(E')$ is a sub-bundle of B'.*

We can define a transition relation between the sub-bundles of a b-bundle analogously to how this is done for event structures: given a b-bundle B, for two sub-bundles B', B'' of B, and an event $e \in E$, we write $B' \xrightarrow{l(e)} B''$ when $e \notin E'$ and $E'' = E' \cup \{e\}$. Altogether, we have:

(iii) Every b-bundle B induces a labelled transition system, where the states are given by the sub-bundles of B and the transition relation describes how a sub-bundle can evolve into a new one by executing an action. The induced labelled transition system is isomorphic to that induced by $bb2ev(B)$.

4.3 Approximation

Every b-bundle of a protocol P captures an initial part of the behaviour of P. We now wish to investigate whether the b-bundles of P consistently approximate, modulo isomorphism, a largest b-bundle. If every protocol P had indeed a largest b-bundle, this b-bundle would capture all possible behaviour of P, and would thus provide a natural strand space semantics for protocols. Furthermore, in view of the results of the previous section this strand space semantics would come with a notion of computation state in terms of bundles, and a transition relation between them. The induced labelled transition system would give the corresponding interleaving semantics of the protocol, while the protocol would also have an abstract partial order semantics in terms of the induced labelled event structure.

First, we need to define a natural notion of approximation for b-bundles. Intuitively, one b-bundle approximates another when it is, up to isomorphism, an initial part of the other. This can be formalized as follows.

Definition 4.7 Let B_1, B_2 be two b-bundles of P. B_1 approximates B_2, written $B_1 \leq B_2$, if there exists an injective homomorphism from B_1 to B_2.

The following observation justifies the naturalness of this definition:

Proposition 4.8 *Let B_1, B_2 be two b-bundles of P. $B_1 \leq B_2$ if and only if B_1 is isomorphic to a sub-b-bundle of B_2.*

Naturally, approximation is preserved by isomorphism. Thus, $\leq$ can be extended to isomorphism classes of b-bundles. Let $IBB(P)$ denote the set of isomorphism classes of b-bundles of P. As one would expect $\leq$ is a partial order on $IBB(P)$.

Proposition 4.9 *$(IBB(P), \leq)$ is a partial order.*

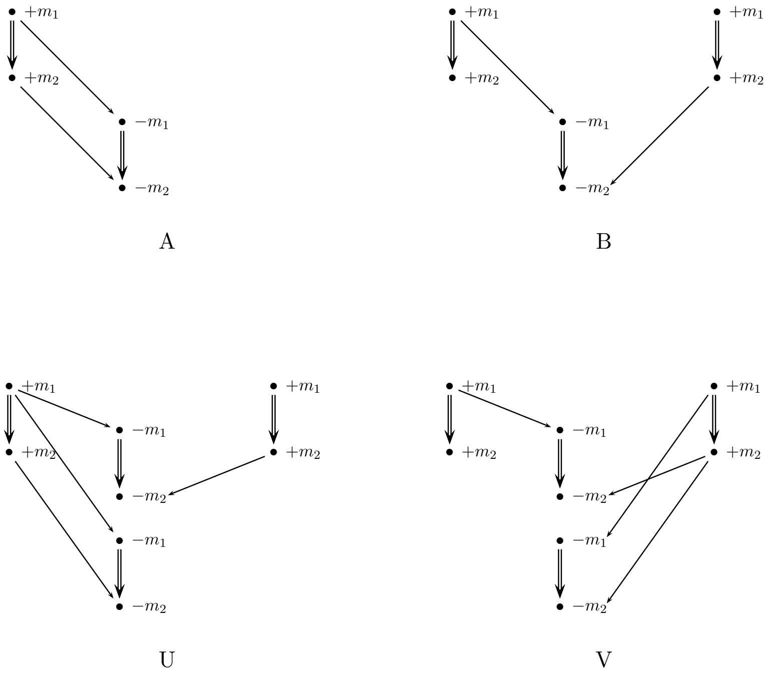


Fig. 2. m_1 and m_2 are ground messages, $+m_1$ stands short for $+$ *fresh* $\emptyset$ *in* m_1, and similarly for $+m_2$. It is clear that there is a protocol P such that A to V are b-bundles of P.

To establish that the b-bundles of P consistently approximate a largest b-bundle we would further like to obtain that $(\mathit{IBB}(P), \leq)$ is a complete lattice. However, we will now demonstrate that this does not hold. Indeed we have:

Proposition 4.10 $(\mathit{IBB}(P), \leq)$ *is neither a lattice nor a complete partial order.*

Proof. To prove this result we will exhibit two b-bundles that have upper bounds but no least upper bound. The b-bundles (which are also bundles) are presented in Figure 2.

Bundle A contains one instance of trace $+m_1\, +m_2$ and one instance of trace $-m_1\, -m_2$, with the send and receive events matched up in the obvious way. Bundle B contains *two* instances of trace $+m_1\ +m_2$ and *one* instance of trace $-m_1\, -m_2$, with the receive events of the latter matched up to send events of *different* strands. Observe that A and B are incomparable: B can clearly not be *injectively* folded onto A; while there cannot be a homomorphism from A to B because there is no strand in B with two outgoing message-flow arrows.

By a similar argument it is clear that any upper bound of A and B must contain at least two instances of trace $+m_1\, +m_2$ and two instances of trace $-m_1\, -m_2$. If two b-bundles contain the same number of events and are comparable with respect to $\leq$ then there will be a bijective homomorphism between them, and hence an isomorphism. Thus, any upper bound of A and B which contains only eight events is, up to isomorphism, a minimal upper bound of A and B.

Consider b-bundles U and V of Figure 2. It is easy to check that both of them are upper bounds of A and B. Further, both of them have only eight events, and thus, up to isomorphism, they must be minimal upper bounds. On the other hand, U and V are not isomorphic: e.g., U has an event labelled by $+m_1$ with two outgoing message-flow arcs while V does not. □

Since $(IBB(P), \leq)$ is not even a complete partial order, $\leq$ cannot be interpreted as a notion of approximation in the information-ordering sense: a b-bundle that is higher in the order does not extend the information of the elements below in a consistent way. It also indicates that a largest b-bundle might simply not exist. Thus, the theory of branching bundles does not turn out to be very satisfying. Analysing the above counter-example will, however, lead us to a satisfying theory of *symbolic* b-bundles.

Remark 4.11 Those readers who are familiar with the strand space model may wonder whether a counter-example could still be obtained if the $\rightarrow$-relation in bundles was disallowed to be forwards-branching (and the Intruder must duplicate messages explicitly). Note that for b-bundles forwards-branching would still be natural, and a more involved counter-example could be constructed.

5 Symbolic Branching Bundles

Let us take another look at the bundles of Figure 2. One could argue that A and B represent the same information with respect to the Intruder's viewpoint. On the one hand, at both, A and B, the information the Intruder *has gained so far* is essentially the same:

> the input to a strand with trace $-m_1\, -m_2$ can be obtained from the send events of a strand with trace $+m_1\, +m_2$, where instances of the latter trace do not require any input.

On the other hand, the information the Intruder *may gain in the future* is also essentially the same at both, A and B:

> for example, to simulate a future of B by a future of A, if the Intruder employs one of the two $+m_1$-events of B as send input to a future strand, he can use the one $+m_1$-event of A in exactly the same way. Furthermore, if in a future of B each of the two $+m_1\, +m_2$ strands is extended by an action such that the actions are different but non-conflicting, then in A a new strand with trace $+m_1\, +m_2$ can be spawned, so as to obtain two non-conflicting strands with analogous traces.

This is why, on second look, it is not surprising that b-bundles do not form an information ordering: there are many inconsistent ways of representing the same information. On the positive side, this also suggests that we may still obtain an information ordering if we work with a notion of *symbolic* b-bundle.

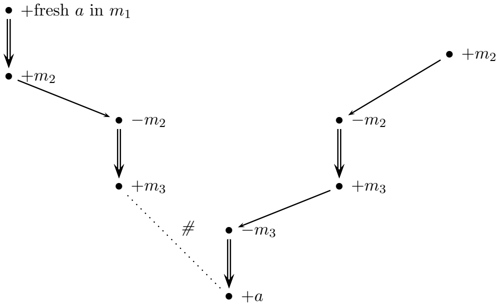


Fig. 3. Assume $a \not\sqsubseteq m_2$ & $a \not\sqsubseteq m_3$.

How could a notion of symbolic b-bundle look like? We would expect that in a symbolic b-bundle all branches that represent essentially the same information are 'glued together', thereby folding a space of b-strands together into a space of *symbolic* b-strands. A space of symbolic b-strands is a b-strand space whose b-strands are considered symbolically. This means we need to relax our notion of conflict. Since a b-strand may now represent several, possibly parallel, sessions, two events that have the same ⇒-predecessor are not necessarily in conflict any more: as long as there is no conflict due to unique origination we could have instantiated parallel sessions.

Definition 5.1 Let $B = (E, \Rightarrow, \rightarrow, GAct, l)$ be a b-strand space of P. Whenever we wish to consider the b-strands of B symbolically then we call B a *symbolic b-strand space*, and redefine the *conflict relation* $\#_B$ by deleting clause (i) from the definition of immediate conflict in Def. 3.2.

Next we need to clarify: when do two branches of a b-bundle represent essentially the same information? One condition is, naturally, that their trace must be the same. However, Figure 3 motivates that we need to adopt a second condition. It illustrates that two strands that have the same trace but different pasts can have different futures: the $+m_3$-event of the strand on the right-hand side can be used as input to the $-m_3$-event of the lowest strand. However, this is not possible for the $+m_3$-event of the strand on the left-hand side, since this would lead to a self-conflict of the $+a$-event due to violation of unique origination. Thus, we only wish to 'glue together' branches when they have the same trace *and* the same past. In more detail: if two branches b and b' have the same trace and for all positions i of the trace, the ith event of b has the same →-predecessor as the ith event of b' then b and b' are to be identified. We can capture this condition more succinctly in terms of events: if two events have the same label and the same predecessors then they are to be identified. Altogether, this leads to the following definition of symbolic b-bundle.

Definition 5.2 A *symbolic b-bundle* B of P is a symbolic b-strand space such that

Axioms (B1) to (B5) and (BB) of the definition of b-bundle hold (the latter with respect to the redefined conflict relation). In addition we impose the following axiom:

SBB for every $e_1, e_2 \in E$, if
(a) $l(e_1) = l(e_2)$, and
(b) $\Rightarrow\text{-}pred(e_1) = \Rightarrow\text{-}pred(e_2)$, and
(c) $\rightarrow\text{-}pred(e_1) = \rightarrow\text{-}pred(e_2)$
then $e_1 = e_2$.

Axiom (SBB) is analogous to, and inspired by, a requirement for Petri net branching processes (Definition 7 of [5]). We also derive a notion of *symbolic bundle.* Naturally, they are based on symbolic b-strands.

Definition 5.3 A *symbolic bundle* is a conflict-free symbolic b-bundle (where the conflict relation is as redefined above).

We will now demonstrate that every b-bundle B of P can indeed be folded onto a symbolic b-bundle. By induction on the causal depth of events we successively identify those events that satisfy conditions (a) to (c) of the definition of symbolic b-bundle. Formally, we define an equivalence relation $\sim$ on $E \cup \{nil\}$ inductively as follows:

(i) $nil \sim nil$,

(ii) $e_1 \sim e_2$ if
(a) $l(e_1) = l(e_2)$,
(b) $\Rightarrow\text{-}pred(e_1) \;\sim\; \Rightarrow\text{-}pred(e_2)$, and
(c) $\rightarrow\text{-}pred(e_1) \;\sim\; \rightarrow\text{-}pred(e_2)$.

It is easy to check that $\sim$ is indeed an equivalence relation. Denote the equivalence class of event e by $[e]_\sim$. Given a b-bundle B, the *folding of B*, denoted by *fold*(B), is defined to be the tuple $(E_f, \Rightarrow_f, \rightarrow_f, GAct, l_f)$ where

- $E_f = \{[e]_\sim \mid e \in E\}$,
- $\forall f_1, f_2 \in E_f$, $f_1 \Rightarrow_f f_2$ if and only if $e_1 \Rightarrow_B e_2$ for some $e_1 \in f_1$, $e_2 \in f_2$,
- $\forall f_1, f_2 \in E_f$, $f_1 \rightarrow_f f_2$ if and only if $e_1 \rightarrow_B e_2$ for some $e_1 \in f_1$, $e_2 \in f_2$,
- $\forall f \in E_f$, $l(f) = a$ if and only if $l(e) = a$ for some (or equivalently all) $e \in f$.

Proposition 5.4 *Let B be a b-bundle of P.*

(i) *fold*(B) *is a symbolic b-bundle of P.*

(ii) *If B is a bundle then fold*(B) *is a symbolic bundle.*

(iii) *fold is a surjective homomorphism from B to fold*(B)*.*

On the other hand, every symbolic b-bundle can be transformed into a b-bundle by disentangling non-conflicting strands that are glued together. In particular, every symbolic bundle can be transformed into a bundle. (These connections will be formalized in an extended version of this paper.)

The transformations give rise to the following observation, which shows how reachability problems on bundles can be reduced to reachability problems on symbolic bundles.

Proposition 5.5 *Assume a protocol P and a finite set of strands S of P. There is a bundle B of P with $S \subseteq \mathit{strands}(B)$ if and only if the events of S are conflict-free and there is a symbolic bundle B' of P with $\mathit{traces}(S) \subseteq \mathit{traces}(B')$. (We use $\mathit{strands}(B)$, $\mathit{traces}(S)$, and $\mathit{traces}(B')$ with the obvious meaning.)*

This shows that it should be adequate to work with symbolic bundles in most situations. Most verification problems for security protocols can be expressed as reachability problems: check whether a situation that represents an attack can be reached. In the strand space model this can be formalized as follows:

Given: A protocol P, and a finite set of strands S.

Decide: Is there a finite bundle B such that $S \subseteq \mathit{strands}(B)$?

6 The Complete Lattice of Symbolic Branching Bundles

It is straightforward to carry over all concepts and positive results of Section 4 to symbolic b-bundles. In particular, we have a partial order $\leq$ on $\mathit{ISBB}(P)$, the set of isomorphism classes of symbolic b-bundles of P. However, now we obtain:

Theorem 6.1 $(\mathit{ISBB}(P), \leq)$ *is a complete lattice.*

Due to Axiom (SBB) the theorem can be proved analogously to Engelfriet's result on Petri net branching processes [5]. Axiom (SBB) ensures that each event of a symbolic b-bundle is uniquely determined by its label and its history. This will allow us to pick a canonical representative for each isomorphism class of symbolic b-bundles in a way such that the structure of each representative is fully determined by its events. For the canonical representatives the notion of approximation can be captured in a simple way, and it will then be straightforward to prove that they form a complete lattice. In the remainder of this section we formally prove Theorem 6.1 along those lines. As usual we fix a protocol P. We will abbreviate *symbolic b-bundle* by *sb-bundle*.

The space from which we will pick the events of the canonical sb-bundles of P is defined as follows: let CAN be the smallest set such that $\mathit{nil} \in \mathit{CAN}$, and if $a \in \mathit{GAct}$ and p_1 and $p_2 \in \mathit{CAN}$ then $(a, p_1, p_2) \in \mathit{CAN}$. For every event of a sb-bundle of P we define its corresponding "coding" in CAN. Formally, given a sb-bundle B of P, the *canonical coding* cod_B of B is the mapping from E to CAN inductively defined on the depth of events as follows:

- $\mathit{cod}_B(e) = (l(e), \mathit{cod}_B(\Rightarrow\text{-}\mathit{pred}(e)), \mathit{cod}_B(\rightarrowtail\text{-}\mathit{pred}(e)))$,
  where we set $\mathit{cod}_B(\mathit{nil}) = \mathit{nil}$.

We will write cod rather than cod_B whenever B is clear from the context. Similarly we write cod_i for cod_{B_i}, where $i \in \{1, 2\}$.

Lemma 6.2 *For every sb-bundle B of P the mapping cod_B is injective.*

Proof. Let B be a sb-bundle of P. For technical convenience we will understand cod_B as a mapping from $E \cup \{nil\}$ (rather than just E) to CAN. We show that for all $e, e' \in E \cup \{nil\}$, if $cod(e) = cod(e')$ then $e = e'$. We prove this by induction on the depth of e. The case $depth(e) = 0$ is straightforward: both e and e' must be nil.

Case $depth(e) > 0$. Then clearly $depth(e') > 0$. Further, we have $cod(\Rightarrow\text{-}pred(e)) = cod(\Rightarrow\text{-}pred(e'))$ and $cod(\rightarrow\text{-}pred(e)) = cod(\rightarrow\text{-}pred(e'))$. By induction hypothesis this implies $\Rightarrow\text{-}pred(e) = \Rightarrow\text{-}pred(e')$ and $\rightarrow\text{-}pred(e) = \rightarrow\text{-}pred(e')$. Since also $l(e) = l(e')$ it follows from Axiom (SBB) of sb-bundle (Def. 5.2) that $e = e'$. □

We now define the canonical sb-bundles.

Definition 6.3 A sb-bundle B of P is *canonical* if $E \subseteq CAN$ and $cod_B(e) = e$ for every $e \in E$.

It is easy to see that every sb-bundle is represented by an isomorphic canonical sb-bundle. Given a sb-bundle B, define $cod(B)$ to be the result obtained by renaming every $e \in E$ by $cod_B(e)$. Since cod_B is injective (Lemma 6.2) $cod(B)$ is a sb-bundle isomorphic to B, and by definition it is canonical. Thus cod_B is an isomorphism from B to $cod(B)$, and in fact, as we will see below the only one.

It remains to show that canonical sb-bundles are *unique* representatives of their isomorphism class, i.e., that two isomorphic canonical sb-bundles are identical. This will follow from Lemma 6.5 below. First we show that homomorphisms between sb-bundles respect the canonical coding of events: every event is mapped to an event with the same coding.

Lemma 6.4 *Let B_1, B_2 be two sb-bundles of P. Let h be a homomorphism from B_1 to B_2. Then $cod_2(h(e)) = cod_1(e)$ for every $e \in E_1$.*

Proof. For all events $e \in E_1$, since by definition of homomorphism, $l_2(h(e)) = l_1(e)$, the first components of $cod_2(h(e))$ and $cod_1(e)$ are the same. Equality of the other two components is proved by induction on the depth of e. In the following we prove equality of the third components. The proof for the second components is similar.

If $\rightarrow\text{-}pred_1(e) = nil$ then $\rightarrow\text{-}pred_2(h(e)) = nil$. This follows from the fact that e must be a send event and since h preserves the labelling $h(e)$ must also be a send event. Then clearly the third components of $cod_2(h(e))$ and $cod_1(e)$ are both nil.

If $\rightarrow\text{-}pred_1(e) \neq nil$ then $\rightarrow\text{-}pred_2(h(e)) \neq nil$, and $h(\rightarrow\text{-}pred_1(e)) = \rightarrow\text{-}pred_2(h(e))$. This follows since h is a homomorphism. But then the equality of the third components of $cod_2(h(e))$ and $cod_1(e)$ follows by induction hypothesis. □

Together with Lemma 6.2 this implies that there is at most one way of how to fold an sb-bundle onto another sb-bundle, and if the homomorphism exists then it will be injective and thus an approximation. For canonical sb-bundles it further means that the homomorphism will be the identity map.

Lemma 6.5 *For sb-bundles B_1 and B_2 of P, there is at most one homomorphism h from B_1 to B_2. If h exists then it is injective. If B_1 and B_2 are canonical then h is the identity on E_1.*

Proof. It is immediate from Lemma 6.2 and Lemma 6.4 that $h(e)$ is the unique element with the same coding as e. If B_1 and B_2 are canonical then $h(e) = \mathit{cod}_2(h(e)) = \mathit{cod}_1(e) = e$. □

In particular, there is exactly one isomorphism between two isomorphic sb-bundles. Also, two isomorphic canonical sb-bundles are identical. Since every sb-bundle B is isomorphic to $\mathit{cod}(B)$ we obtain:

Corollary 6.6 *For sb-bundles B_1 and B_2 of P, B_1 and B_2 are isomorphic if and only if $\mathit{cod}(B_1) = \mathit{cod}(B_2)$.*

Let $\mathit{CSBB}(P)$ denote the set of canonical sb-bundles of P. It follows from Corollary 6.6 that $(\mathit{ISBB}(P), \leq)$ and $(\mathit{CSBB}(P), \leq)$ are isomorphic structures, with the isomorphism that maps the isomorphism class of a sb-bundle B to $\mathit{cod}(B)$. Thus, to show that $(\mathit{ISBB}(P), \leq)$ is a complete lattice, it suffices to show that $(\mathit{CSBB}(P), \leq)$ is one. We will achieve the latter in Lemma 6.9 below and thereby complete the proof of Theorem 6.1.

It is a consequence of Lemma 6.5 that for canonical sb-bundles B_1 and B_2, $B_1 \leq B_2$ if and only if B_1 is a sub-b-bundle of B_2. The situation is even simpler though: as stated in the next lemma the structure of a canonical sb-bundle is fully determined by its events, and thus $B_1 \leq B_2$ if and only if $E_1 \subseteq E_2$.

Lemma 6.7 *Let B be a canonical sb-bundle of P. Then for every $e, e' \in E$,*

(i) *$e \Rightarrow e'$ if and only if $e' = (a, e, p)$ for some $a \in \mathit{GAct}$, $p \in E \cup \{\mathit{nil}\}$,*

(ii) *$e \rightarrow e'$ if and only if $e' = (a, p, e)$ for some $a \in \mathit{GAct}$, $p \in E \cup \{\mathit{nil}\}$,*

(iii) *$l(e) = a$ if and only if $e = (a, p_1, p_2)$ for some $p_1, p_2 \in E \cup \{\mathit{nil}\}$.*

Proof. This is straightforward since by definition of cod_B and canonical sb-bundle, for every $e \in E$, $e = (l(e), \Rightarrow\text{-}\mathit{pred}(e), \rightarrow\text{-}\mathit{pred}(e))$. □

Lemma 6.8 *Let B_1, B_2 be canonical sb-bundles of P. Then $B_1 \leq B_2$ if and only if $E_1 \subseteq E_2$.*

Proof. (If direction) If $E_1 \subseteq E_2$ then by Lemma 6.7 the identity on E_1 is a homomorphism from B_1 to B_2. Hence $B_1 \leq B_2$.

(Only if direction) If $B_1 \leq B_2$ then there is a homomorphism h from B_1 to B_2. By Lemma 6.5, h is the identity on E_1, and thus $E_1 \subseteq E_2$. □

Since approximation is captured in this simple way it is now straightforward to establish the existence of least upper bounds for canonical sb-bundles. We obtain:

Lemma 6.9 *$(\mathit{CSBB}(P), \leq)$ is a complete lattice.*

Proof. It is immediate from Lemma 6.8 that $\leq$ is a partial order on $\mathit{CSBB}(P)$. Further, we need to show that every subset of $\mathit{CSBB}(P)$ has a least upper bound with respect to $\leq$. Let B_i be a canonical sb-bundle of P for every i in some index set I. Let $B = (E, \Rightarrow, \rightarrow, GAct, l)$ with $E = \bigcup_{i \in I} E_i$, and $\Rightarrow$, $\rightarrow$, and l be defined by the statement of Lemma 6.7.

By Lemma 6.8, if B is a canonical sb-bundle of P then it is the least upper bound of all B_i, $i \in I$, with respect to $\leq$. Thus it only remains to show that B is indeed a canonical sb-bundle of P.

It is easy to check that B satisfies the axioms of being a sb-bundle, using Lemma 6.7 and the fact that each B_i is a canonical sb-bundle. To see that B is canonical, observe that by definition of B, for every $e \in E$, $e = (l(e), \Rightarrow\text{-}pred(e), \rightarrow\text{-}pred(e))$. On the other hand, by definition of cod_B, $cod_B(e) = (l(e), cod_B(\Rightarrow\text{-}pred(e)), cod_B(\rightarrow\text{-}pred(e)))$. But then $cod_B(e) = e$ follows by induction on the depth of e. □

7 Conclusions

In the last section we have shown that for every protocol P, $ISBB(P)$, the set of isomorphism classes of symbolic b-bundles of P, forms a complete lattice with respect to approximation. In particular, this implies that $ISBB(P)$ has a largest element, which captures all possible behaviour of P in a symbolic fashion. We call it the *symbolic unfolding* of P. Thus, every protocol has a strand space semantics in terms of its symbolic unfolding. Further, by the results of Section 4.2 this semantics comes with a notion of computation state in terms of symbolic sub-bundles, a transition relation, and close relations to event structures.

It remains to be investigated whether restricting our attention to the symbolic unfolding is indeed suitable in most situations of security protocol analysis. We will also examine whether it can help with the state space explosion problem in model-checking tools, or whether it is relevant in the context of strand space based analysis tools such as the Cryptographic Protocol Shape Analyzer [4]. On the theoretical side, the relationship between symbolic b-bundles and b-bundles can be further formalized using category theory.

Acknowledgement

I thank the anonymous referees for their valuable comments.

References

[1] Federico Crazzolara and Glynn Winskel. Events in security protocols. In *ACM Conference on Computer and Communications Security*, pages 96–105, 2001.

[2] Federico Crazzolara and Glynn Winskel. Petri nets in cryptographic protocols. In *IEEE IPDPS-01*, page 149, 2001.

[3] Federico Crazzolara and Glynn Winskel. Composing strand spaces. In *FST TCS 2002*, volume 2556 of *Lecture Notes in Computer Science*, pages 97–108, 2002.

[4] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Searching for shapes in cryptographic protocols. In *In TACAS 2007*, volume 4424 of *Lecture Notes in Computer Science*, pages 523–538. Springer, 2007.

[5] Joost Engelfriet. Branching processes of Petri nets. *Acta Inf.*, 28(6):575–591, 1991.

[6] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Symposium on Security and Privacy*. IEEE Computer Society, 1998.

[7] Sibylle Fröschle. The insecurity problem: Tackling unbounded data. In *CSF 2007*, pages 370–384. IEEE, 2007.

[8] MITRE. The strand space method web page. http://www.mitre.org/tech/strands/.

A Appendix

A.1 Proofs of Sections 3 and 4

Proof. [Prop. 3.4] If B is a bundle then $\#_B = \emptyset$ and thus Axiom (BB) is trivially satisfied. If B is a b-bundle with $\#_B = \emptyset$ then every b-strand of B is a strand, and (B6) will be satisfied. □

Proof. [Prop. 4.4] (i) The if-direction is immediate. For the only-if direction, it follows from the observation below Def. 4.2 that E' is downwards-closed and $B' = sbb(E')$. (ii) In addition to the previous argument use Prop. 3.4. □

Proof. [Prop. 4.5] We know from Section 3 that $\preceq_B$ is a partial order and that $\#_B$ is symmetric and irreflexive. The first axiom of the definition of event structures follows from Axiom (B5) of b-bundle, and the second from the definition of $\#_B$. □

Proof. [Prop. 4.6] This is immediate from the definition of configuration and Prop. 4.4(ii). □

Proof. [Prop. 4.8] This is routine to check. □

Proof. [Prop. 4.9] Reflexivity is immediate from the definition. Transitivity follows by compositionality of homomorphism. For antisymmetry consider that if there is an injective but non-surjective homomorphism from one b-bundle to another then there clearly cannot be an injective homomorphism in the other direction. □

A.2 Proofs of Section 5

Proposition A.1 *Let B and $fold(B)$ be as given in Section 5.*

(i) *If $f_1 \rightarrow_f f_2$ then $\forall e_2 \in f_2$, $e_1 \rightarrow_B e_2$ for some $e_1 \in f_1$, and analogously: if $f_1 \Rightarrow_f f_2$ then $\forall e_2 \in f_2$, $e_1 \Rightarrow_B e_2$ for some $e_1 \in f_1$.*

(ii) *$(\rightarrow_f \cup \Rightarrow_f)$ is acyclic.*

Proof. (i) follows from the definition of *fold*(B).

(ii) If $(\rightarrow_f \cup \Rightarrow_f)$ is not acyclic then using (i) one can show that there is a $(\rightarrow \cup \Rightarrow)$-path in B from some event e to an event $e' \neq e$ such that $e \sim e'$. However, $e \sim e'$ implies e and e' must be of the same depth. But this can surely not be the case given the existence of the path. Thus, we must have reached a contradiction. □

Proof. (of Prop. 5.4)

(i) We first show that $\mathit{fold}(B)$ is indeed a space of b-strands of P. By Prop. A.1(ii) $\Rightarrow$ is acyclic. By Prop. A.1(i) and the fact that $\Rightarrow$ has no backwards branching, $\Rightarrow_f$ has no backwards branching either. Further, with Prop. A.1(i) and considering that B is a b-strand space of P it is easy to show that every $\Rightarrow_f$-branch must be finite and a strand of P.

Next we check whether the axioms of symbolic b-bundles are satisfied. Axioms (B1) and (B3) follow from the definition of $\mathit{fold}(B)$ and the fact that the respective axiom holds for B. Axiom (B2) follows from Prop. A.1(i) and the fact that (B2) holds for B. Axiom (B4) has already been proved as Prop. A.1(ii). Axiom (B5) follows from the definition. To prove Axiom (BB) assume to the contrary an event $f \in E_f$ with $f \# f$. Then there must be $f_1 \preceq f$ and $f_2 \preceq f$ with $f_1 \#_1 f_2$. By our definition of immediate conflict for symbolic b-strand spaces this means $f_1 \#_1 f_2$ is due to a violation of unique origination. Take any $e \in f$. By Prop. A.1(i) we obtain $e_1 \preceq e$, $e_2 \preceq e$ with $e_1 \#_1 e_2$, a contradiction to the fact that (BB) holds for B.

It only remains to show that Axiom (SBB) is satisfied. But this is straightforward: assuming the contrary leads to a contradiction with the definition of E_f as the equivalence classes of $\sim$.

(ii) If B is a bundle then its set of events is conflict-free (Prop. 3.4). Then $\mathit{fold}(B)$ is also conflict-free, and thus a symbolic bundle.

(iii) Axiom (i) amd (iii) of homomorphism directly follow from the definition of fold. (ii) follows by Prop. A.1(i). The homomorphism is surjective by definition. □

Proof. [Prop. 5.5] If there is a bundle B with $S \subseteq \mathit{strands}(B)$ then clearly S is conflict-free and $\mathit{fold}(B)$ is a symbolic bundle as required.

If there is a symbolic bundle B' with $\mathit{traces}(S) \subseteq \mathit{traces}(B')$ then B' can be extended to a bundle such that $S \subseteq \mathit{strands}(B)$.

□