BRICS

Basic Research in Computer Science

Composing Strand Spaces

Federico Crazzolara
Glynn Winskel

BRICS Report Series **RS-02-5**

ISSN 0909-0878 **February 2002**

**Copyright © 2002, Federico Crazzolara & Glynn Winskel.
BRICS, Department of Computer Science
University of Aarhus. All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

**See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:**

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/02/5/`

Composing strand spaces

Federico Crazzolara* Glynn Winskel
{fc232, gw104}@cl.cam.ac.uk

Computer Laboratory
University of Cambridge

Abstract

The strand space model for the analysis of security protocols is known to have some limitations in the patterns of nondeterminism it allows and in the ways in which strand spaces can be composed. Its successful application to a broad range of security protocols may therefore seem surprising. This paper gives a formal explanation of the wide applicability of strand spaces. We start with an extension of strand spaces which permits several operations to be defined in a compositional way, forming a process language for building up strand spaces. We then show, under reasonable conditions how to reduce the extended strand spaces to ones of a traditional kind. For security protocols we are mainly interested in their safety properties. This suggests a strand-space equivalence: two strand spaces are equivalent if and only if they have essentially the same sets of bundles. However this equivalence is not a congruence with respect to the strand-space operations. By extending the notion of bundle we show how to define the strand-space operations directly on "bundle spaces". This leads to a characterisation of the largest congruence within the strand-space equivalence. Finally, we relate strand spaces to event structures, a well known model for concurrency.

*BRICS Basic Research in Computer Science, Centre of the Danish National Research Foundation

Contents

1 Introduction **2**

2 Strand spaces **4**

3 Strand spaces with conflict **6**

4 Constructions on strand spaces **10**

5 A process language for strand spaces **14**

6 Bundle spaces **15**

7 A strand space congruence **19**

8 Eliminating conflict **22**

9 Event structures from strand spaces **25**

1 Introduction

Security protocols describe a way of exchanging data over an untrusted medium so that, for example, data is not leaked and authentication between the participants in the protocol is guaranteed. The last few years have seen the emergence of successful intensional, event-based, approaches to reasoning about security protocols. The methods are concerned with reasoning about the events that a security protocol can perform, and make use of a causal dependency that exists between events. Typically, a secrecy property (or some strengthening of it) is established by showing that there cannot be an earliest event in the causal dependency which violates the property, while authentication is often established by showing certain events of one agent depend on certain events of another. The method of strand spaces [THG98b, THG98a, TG00] has been designed to support such an intensional, event-based, style of reasoning and has successfully been applied to a broad number of security protocols.

Security properties such as secrecy and authentication or even anonymity can be expressed as safety properties, properties which stand or fall according to whether they hold for all finite behaviours. The results in this paper express the adequacy of strand spaces and relate strand spaces

to event structures only with respect to the languages i.e. sets of finite behaviours, they generate. This is not unduly restrictive however as in security protocols we are mainly interested in safety properties.

When we were relating strand spaces to a Petri-net semantics for a process language designed to describe security protocols [CW01], we had to face the fact that strand spaces don't compose readily, not using traditional process operations at least. Their form doesn't allow prefixing by a single event. Nondeterminism only arises through the choice as to where input comes from, and there is not a recognisable nondeterministic sum of strand spaces. Even an easy definition of parallel composition by juxtaposition is thwarted if "unique origination" is handled as a global condition on the entire strand space. That strand spaces are able to tackle a broad class of security protocols may therefore seem surprising. A reason for the adequacy of strand spaces lies in the fact that they can sidestep conflict if there are enough replicated strands available, which is the case for a broad range of security protocols.

This paper has four main objectives. Firstly it extends the strand space formalism to allow several operations on strand spaces to be defined. The operations form a strand-space language. Secondly the wide applicability of strand spaces to numerous security protocols and properties is backed up formally. The paper documents part of the work done in proving the relation between nets and strand spaces we reported in [CW01]. Thirdly we address another issue of compositionality. Because we are only interested in safety properties we can make do with languages of strand-space bundles as models of process behaviour. We show how to compose such languages so that they may be used directly in giving the semantics of security protocols. Strand spaces that have substantially the same bundles can be regarded as equivalent and are congruent if they exhibit substantially the same open bundles. This congruence lays the ground for equational reasoning between strand spaces. Finally we show how strand spaces relate to event structures.

In Section 2 we briefly introduce the strand space formalism in its traditional form and discuss some limitations. Section 3 shows how to extend strand spaces in order to compose them, chiefly with conflict to permit their nondeterministic sum. A treatment of "unique origination" on the bundle rather than on the strand space allows us to define parallel composition of strand spaces by juxtaposition. The operations of prefixing, parallel composition and nondeterministic sum of strand spaces are illustrated in Section 5 and form a language for strand spaces. In Sec-

tion 7 we give a congruence relation between terms of the strand-space language based on the underlying bundle languages studied in Section 6. In Section 8 we show that conflict can be eliminated, without upsetting the strand-space behaviour if enough "replication" is introduced. Section 9 describes how strand spaces relate to event structures.

2 Strand spaces

We briefly introduce the strand space formalism of [THG98b] and discuss some apparent limitations.

A strand spaces consists of $\langle s_i \rangle_{i \in I}$, an indexed set of strands. An individual strand s_i, where $i \in I$, is a finite sequence of output or input events carrying output or input actions of the kind $outM$ or inM respectively with M a message built up by encryption and pairing from a set of values (here names) and keys. In the rest of this paper we use n, n_0 to indicate names, A, B, A_0, B_0 to indicate special names which are agent identifiers, and k standing for a cryptographic key. A name whose first appearance in a strand is on an output message, is said to be *originating* on that strand. A value is said to be *uniquely originating* on a strand space if it is originating on only one of its strands.

A strand space has an associated graph whose nodes identify an event of a strand by strand index and position of the event in that strand. Edges are between output and input events concerning the same message and between consecutive events on a same strand. Bundles model protocol runs. A bundle selects from the events of a strand space those that occur in a run of the protocol and shows the causal dependencies among them which determine the partial order of the events in the run. A bundle is a finite and acyclic subgraph of the strand space graph. Each node in the bundle requires all events that precede it on the same strand (together with the edges that denote the strand precedence). Moreover each input node in the bundle has exactly one incoming edge from an output node.

As an example consider a simplified version of the ISO symmetric key two-pass unilateral authentication protocol (see [CJ97]):

$$\begin{array}{l} A \rightarrow B : n \\ B \rightarrow A : \{n, A\}_k \end{array}$$

Agents can engage in a protocol exchange under two different roles. The initiator, here A and the responder, here B. In a protocol round the initiator A chooses a fresh name n and sends it to the responder B. After

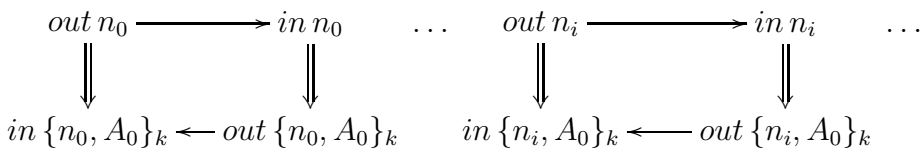


Figure 1: ISO protocol

getting the value, B encrypts it together with the initiator's identifier using a common shared key k. After getting the answer to her challenge, A can decrypt using the shared key and check whether the value sent matches the value received. In that case A can conclude that B is in fact operational. The strand space graph in Figure 1 describes the simple case of only two agents, A_0 and B_0, acting as initiator and responder respectively. For simplicity the graph has been drawn using the actions labelling the events in place of the events themselves. In this simple case the strand space itself forms a bundle. All the names n_i are uniquely originating on that strand space.

Unique origination intends to describe a name as fresh, perhaps chosen at random, and under the assumptions of Dolev and Yao [DY83], unguessable. For a construction of parallel composition of strand spaces it is therefore reasonable to require that names uniquely originating on components remain so on the composed strand space. Simple juxtaposition of strand spaces does not ensure this. For example consider a strand space for the ISO protocol which allows both agents A_0 and B_0 to engage in the protocol in any of the two possible roles. In Figure 2 the strand space formed out of two copies of the one in Figure 1. Figure 3 shows a possible bundle on such strand space. It describes a protocol run with two complete rounds. One in which A_0 is initiator and B_0 responder and another where the roles are inverted. Though the name n_0 is no longer uniquely originating on that strand space. A name's freshness is with respect to a run of a protocol more than to the whole set of possible executions. A notion of unique origination "on the bundle" seems more appropriate.

Nondeterminism in strand spaces arises only through the choice in a bundle of where input comes from. There is no recognisable way of modelling situations in which bundles may be taken either only over one strand space or over another. Juxtaposing strands as we did for example in Figure 2 allows bundles to include events of both components as is the

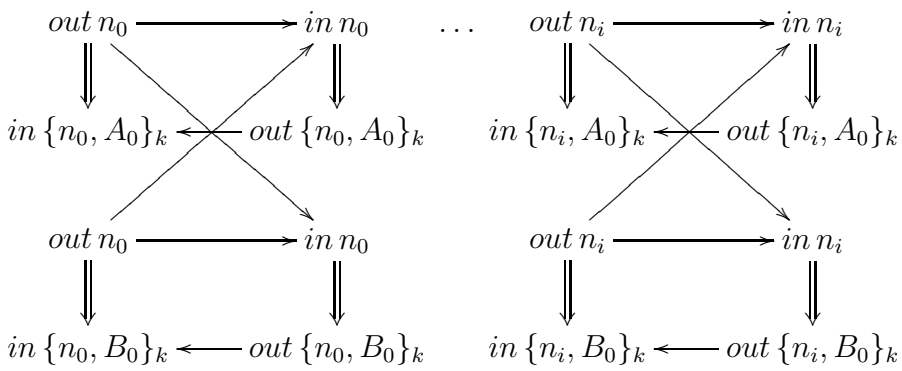


Figure 2: ISO protocol - symmetric roles

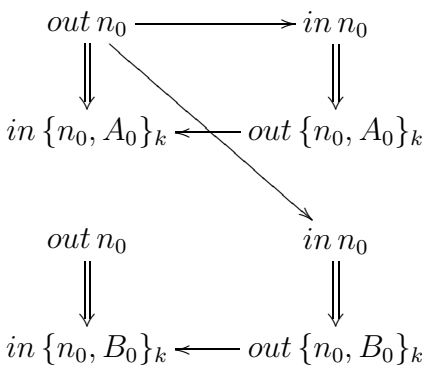


Figure 3: A possible bundle

case for the bundle in Figure 3.

One seems to encounter even more difficulties in the attempt to define a construction of prefixing a strand space with an action. Strands can't branch to parallel sub-strands and prefixing each strand of the space with an action would cause as many repetitions of that action as there are strands participating in a bundle.

3 Strand spaces with conflict

In this section we extend the definition of strand space, introducing a notion of conflict, which we adapt from event structures (see e.g.[Win88]). We differ from the original definition of strand spaces in the treatment

of unique origination which is taken care of in the definition of bundle rather than being a condition on the entire strand space – the "parametric strand spaces" of [CDL+00] achieve a similar effect as to unique origination and are related.

As mentioned in the previous section, the strands of a strand space consist of sequences of output and input actions. Actions are

$$Act = \{out\, new\, \vec{n}\, M \mid M \text{ msg},\ \vec{n} \text{ distinct names}\} \cup \{in\, M \mid M \text{ msg}\}.$$

In $out\, new\, \vec{n}\, M$, the list $\vec{n}$ contains distinct names that are intended to be fresh ("uniquely originating") at the event.

Definition 3.1 A strand space with conflict $(\langle s_i \rangle_{i \in I}, \#)$ consists of:

(i) $\langle s_i \rangle_{i \in I}$ an indexed set of strands. An individual stand s_i, where $i \in I$, is a finite sequence of output or input actions in Act.

(ii) $\# \subseteq I \times I$ a symmetric, irreflexive binary conflict relation on strand indexes.

Strand spaces with an empty conflict relation correspond to those of the standard definition of [THG98b]. We denote by ϵ the empty strand space with no strands and with an empty conflict relation. [1] If λ stands for an empty sequence of actions then the empty strand space is different to a strand space where each strand is the empty sequence of actions $(\langle \lambda \rangle_{i \in I}, \#)$. We write $|s|$ for the length of the sequence s. Given a strand space $(\langle s_i \rangle_{i \in I}, \#)$, given an index $j \in I$, and given l such that $1 \leq l \leq |s_j|$ we write $act(j, l)$ for the action at position l in s_j.

Given a strand space $(\langle s_i \rangle_{i \in I}, \#)$, we can find a strand-space graph

$$(E, \Rightarrow, \rightarrow)$$

associated with it as usual (see [THG98b]). The graph has nodes (events)

$$E = \{(i, l) \mid i \in I\ , 1 \leq l \leq |s_i|\}$$

and edges

- $(i, l) \Rightarrow (i, l + 1)$ iff $(i, l), (i, l + 1) \in E$,
- $(i, l) \rightarrow (j, h)$ iff $act(i, l) = out\, new\, \vec{n}\, M$ and $act(j, h) = in\, M$.

[1] We won't make much use of this particular strand space; it is however the identity for the operations of parallel composition and nondeterministic sum of strand spaces.

For the components of G strand-space graph or subgraph of the strand-space graph we often write E_G, $\Rightarrow_G$, and $\rightarrow_G$, and we write

$$\leq_G = (\Rightarrow_G \cup \rightarrow_G)^*$$

for the reflexive and transitive closure of the relation $\Rightarrow_G \cup \rightarrow_G$. We refer to $\rightarrow_G$ as the *communication edges* of G. The names of a node e are all the names appearing on the action associated with e – the ones that are marked as "new" together with those in the message of the action. Write $names(e)$ for the names of node e. We say that an event is an *input event* if the action associated with it is an input action and we say an event is an *output event* if its action is an output.

Bundles of a strand space describe runs in a computation.

Definition 3.2 A bundle b of a strand space $(\langle s_i \rangle_{i \in I}, \#)$ is a finite, acyclic subgraph of G the graph of $(\langle s_i \rangle_{i \in I}, \#)$ such that:

(i) if $e \Rightarrow_G e'$ and $e' \in E_b$ then $e \Rightarrow_b e'$, *(control precedence)*

(ii) if $e \in E_b$ and $act(e) = in\, M$ then there exists a unique $e' \in E_b$ such that $e' \rightarrow_b e$, *(output-input precedence)*

(iii) if $e, e' \in E_b$ such that $act(e) = out\, new\, \vec{n}\, M$ and $n \in \vec{v} \cap names(e')$ then either $e \Rightarrow_b^* e'$ or there exists an input event e'' such that $n \in names(e'')$ and $e'' \Rightarrow_b^* e'$, *(freshness)*

(iv) if $(i, h), (j, k) \in E_b$ then $\neg(i \# j)$. *(conflict freeness)*

The empty graph, denoted by λ, is a bundle. It will be clear from the context whether λ stands for the empty bundle or whether it denotes the empty sequence of actions. The empty strand space has only one bundle, the empty bundle.

The first two points of the definition of bundle for a strand space with conflict match with the standard definition of [THG98b]. There are two additional requirements. Point (iii) ensures freshness of "new" values in a bundle. Point (iv) doesn't allow events from conflicting strands to appear in a bundle.

Proposition 3.3 *If b is a bundle then $\leq_b$ is a partial order on E_b.*

Proof. A bundle is an acyclic subgraph of the strand-space graph. □

The relation $\leq_b$ determines the partial order that events respect when occurring in a computation described by the bundle. As one would expect, names that are introduced as "new" don't appear on events preceding their introduction and are never introduced as "new" more than once.

Proposition 3.4 *Let b be a bundle of a strand space. If $e, e' \in E_b$ such that $act(e) = out\, new\, \vec{n}\, M$ and $n \in \vec{v} \cap names(e')$ then $e \leq_b e'$ and if $act(e') = out\, new\, \vec{m}\, M'$ then $n \notin \vec{m}$.*

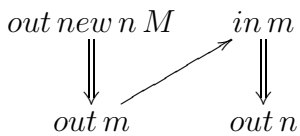
Proof. Suppose that $e \not\leq_b e'$ and therefore $e \not\Rightarrow^* e'$. There exists an input event $e'' \in b$ such that $n \in names(e'')$ and $e'' \Rightarrow_b^* e'$ (freshness). The bundle b is acyclic and each input event in b is preceded by a matching output event (output-input precedence). Therefore there exists an output event $e_1 \in E_b$ such that $n \in names(e_1)$ and such that for every input event e_2 if $e_2 \Rightarrow_b^* e_1$ then $n \notin names(e_2)$. The event e can't precede e_1 on the same strand ($e \not\Rightarrow_b^* e_1$), otherwise $e \leq_b e'$. The events e and e_1 are both in b thus contradicting the freshness property of b.

If $act(e') = out\, new\, \vec{m}\, M'$ and $n \in \vec{m}$ then $e \leq_b e'$ and $e' \leq_b e$, and therefore $e = e'$. □

There are other possible choices for the freshness condition (iii). A weaker condition could be the following:

> if $e, e' \in E_b$ such that $act(e) = out\, new\, \vec{n}\, M$ and such that $n \in \vec{v} \cap names(e')$ then $e \leq_b e'$.

This condition however would allow bundles of the kind



If the two strands are distinct processes, the second strand is not supposed to send the name n without receiving it first from somewhere. Graphs like that are not considered bundles in the original treatment of strand spaces [THG98b] and are also excluded by the slightly more involved freshness condition of Definition 3.2.

We regard two strand spaces as equivalent if they differ only on the indexes of their strands and therefore one strand space can be obtained from the other by a simple "re-indexing" operation. [2]

[2] If the indexes carry structure (some might involve agent names for example) we might refine the permissible re-indexings.

Definition 3.5 Given $(\langle s_i \rangle_{i \in I}, \#)$ and $(\langle t_j \rangle_{j \in J}, \#')$ two strand spaces write $(\langle s_i \rangle_{i \in I}, \#) \cong (\langle t_j \rangle_{j \in J}, \#')$ if there exists a bijection $\sigma : I \to J$ such that:

(i) $\forall i \in I \,.\, s_i = t_{\sigma(i)}$ and

(ii) $\forall i, j \in I \,.\, i \# j \Leftrightarrow \sigma(i) \#' \sigma(j)$.

The relation $\cong$ is an equivalence relation on strand spaces. A bijection σ which establishes such equivalence is called a *re-indexing of strand spaces*. Moreover take $(\langle s_i \rangle_{i \in I}, \#)$ a strand space, J a set, and $\sigma : I \to J$ a bijection. We define the strand space $(\langle t_j \rangle_{j \in J}, \sigma(\#))$ where

- $\forall j \in J \,.\, t_j = s_{\sigma^{-1}(j)}$ and
- $\forall j, j' \in J \,.\, j \, \sigma(\#) \, j'$ iff $\sigma^{-1}(j) \# \sigma^{-1}(j')$.

The relation $\sigma(\#)$ is irreflexive and symmetric and establishes the equivalence $(\langle s_{\sigma(i)} \rangle_{i \in I}, \sigma(\#)) \cong (\langle s_i \rangle_{i \in I}, \#)$.

Proposition 3.6 *Let $(\langle s_i \rangle_{i \in I}, \#)$ and $(\langle t_j \rangle_{j \in J}, \#')$ be two strand spaces such that $(\langle s_i \rangle_{i \in I}, \#) \cong (\langle t_j \rangle_{j \in J}, \#')$ for a bijection $\sigma : I \to J$. Given b a bundle of $(\langle s_i \rangle_{i \in I}, \#)$ then $\sigma(b)$ obtained from b by changing all strand indexes according to σ is a bundle of $(\langle t_j \rangle_{j \in J}, \#')$.*

Proof. Follows from the assumption that b is a bundle and from the definition of re-indexing on bundles. □

4 Constructions on strand spaces

The extension of the strand-space formalism with a conflict relation and the different treatment of unique origination as illustrated in the previous section allow operations such as prefixing, parallel composition, and sum of strand spaces to be defined in terms of traditional process operations.

The operation of prefixing a strand space with an action is complicated by the strand-space formalism not permitting strands to branch. Only if the strand space to be prefixed is such that every two different strands are in conflict can each strand be prefixed with the action. The

conflict relation in this case doesn't allow repetitions of the prefixing action in bundles. Given α an action and given a strand space $(\langle s_i \rangle_{i \in I}, \#)$ such that for all $i, j \in I$ if $i \neq j$ then $i \# j$, define

$$\alpha.(\langle s_i \rangle_{i \in I}, \#) \stackrel{def}{=} (\langle \alpha s_i \rangle_{i \in I}, \#) \ .$$

We understand the special case of prefixing the empty strand space with an action, to yield the empty strand space:

$$\alpha.\epsilon = \epsilon \ .$$

For example, when prefixing a strand space consisting of only empty strands one obtains a strand space with the same number of strands each one with only one action, the prefixing action. More precisely

$$\alpha.(\langle \lambda \rangle_{i \in I}, \#) = (\langle \alpha \rangle_{i \in I}, \#) \ .$$

The operation of parallel composition of two strand spaces is the disjoint union of their sets of strands and conflict relations. Disjoint union is achieved by tagging the first space with index 0 and the second with index 1. Given strand spaces $(\langle s_i^0 \rangle_{i \in I}, \#^0)$ and $(\langle s_j^1 \rangle_{j \in J}, \#^1)$ define

$$(\langle s_i^0 \rangle_{i \in I}, \#^0) \ || \ (\langle s_j^1 \rangle_{j \in J}, \#^1) \stackrel{def}{=} (\langle s_h \rangle_{h \in H}, \#)$$

where $H = (\{0\} \times I) \cup (\{1\} \times J)$, $s_{(0,i)} = s_i^0$ and $s_{(1,i)} = s_i^1$. Two strands are in conflict only if they belong to the same component of the parallel composition and are in conflict within that component. More precisely for k either 0 or 1 define $h \# h'$ if $h = (k, i)$, $h' = (k, j)$ and $i \#^k j$.

The operation of nondeterministic sum of two strand spaces constructs the same indexed set of strands as the operation of parallel composition. The conflict relation of a summed space however, in addition to the existing conflicts, imposes conflict between every two strands that belong to different components. Given strand spaces $(\langle s_i^0 \rangle_{i \in I}, \#^0)$ and $(\langle s_j^1 \rangle_{j \in J}, \#^1)$ define

$$(\langle s_i^0 \rangle_{i \in I}, \#^0) \ + \ (\langle s_j^1 \rangle_{j \in J}, \#^1) \stackrel{def}{=} (\langle s_h \rangle_{h \in H}, \#)$$

where $H = (\{0\} \times I) \cup (\{1\} \times J)$, $s_{(0,i)} = s_i^0$ and $s_{(1,i)} = s_i^1$. Two strands are in conflict only if they belong to different components or are already in conflict within a component. More precisely for k and k' either 0 or 1 define $(k, i) \# (k', j)$ if $k \neq k'$ or if $k = k'$ and $i \#^k j$.

In both operations of strand-space composition the relation $\#$ is irreflexive and symmetric.

The operations of parallel composition and nondeterministic sum satisfy some desired properties with respect to the equivalence $\cong$ of strand spaces:

Proposition 4.1 *Let S_0 , S_1, and S_2 be strand spaces.*

1. $S_0||\epsilon \cong S_0$
2. $S_0||(S_1||S_2) \cong (S_0||S_1)||S_2$
3. $S_0||S_1 \cong S_1||S_2$

and similarly for $+$.

Proof. Let $S_0 = (\langle s_i^0 \rangle_{i \in I}, \#^0)$, $S_1 = (\langle s_j^1 \rangle_{j \in J}, \#^1)$, and $S_2 = (\langle s_k^2 \rangle_{k \in K}, \#^2)$.

1. The strand space $S_0||\epsilon$ has the same strands as S_0, but indexing set $\{0\} \times I$. Take $\sigma : \{0\} \times I \to I$ to be the projection to the second component which, in this case, is a bijection. It gives an equivalence between the two strand spaces. Every strand $s_{(0,i)}$ in $S_0||\epsilon$ is equal to s_i, therefore $s_{(0,i)} = s_{\sigma(0,i)}$. Let $\#$ be the conflict relation of $S||\epsilon$. If $(0,i) \# (0,i')$ then by definition of $\#$ there is conflict $i \#^0 i'$, therefore $\sigma(0,i) \#^0 \sigma(0,i')$.

2. Both spaces have the same strands but the first has indexing set

$$H = (\{0\} \times I) \cup (\{1\} \times ((\{0\} \times J) \cup (\{1\} \times K)))$$

while the second has indexing set

$$H' = (\{0\} \times ((\{0\} \times I) \cup (\{1\} \times J))) \cup (\{1\} \times K).$$

Consider the following function $\sigma : H \to H'$

$$\sigma(h) = \begin{cases} (0,(0,i)) & \text{if } h = (0,i) \\ (0,(1,j)) & \text{if } h = (1,(0,j)) \\ (1,k) & \text{if } h = (1,(1,k)) \end{cases}$$

which is a bijection and establishes the desired equivalence.

3. Straightforward.

$\square$

The equivalence $S + S \cong S$ does not always hold. For example consider the strand space S composed out of one single strand with index i. The indexing set of $S + S$ is $\{(0, i), (1, i)\}$. There is no bijection between a set of one element and a set of two elements.

Strands are not permitted to branch and the prefixing operation prefixes each single strand of the space. The following proposition holds:

Proposition 4.2 *Let S_0 and S_1 be strand spaces and α an action. Then*

$$\alpha.(S_0 + S_1) = \alpha.S_1 + \alpha.S_0.$$

Proof. Follows from the definition of prefixing operation and sum. □

We can extend the definition of binary parallel composition and nondeterministic sum of strand spaces to operations indexed over a set. Given a collection of strand spaces $(\langle s_i^k \rangle_{i \in I_k}, \#^k)$ indexed by k in a set K, define

$$||_{k \in K}(\langle s_i^k \rangle_{i \in I_k}, \#^k) \stackrel{def}{=} (\langle s_h \rangle_{h \in H}, \#),$$

where $H = \sum_{k \in K} I_k$, $s_{(k,i)} = s_i^k$, and where $(k, i) \# (k', i')$ iff $k = k'$ and $i \#^k i'$. In particular if K is the empty set then the parallel composition yields the empty strand space.

As a special case of parallel composition of strand spaces consider the strand space obtained by composing infinitely many but equal strand spaces. Abbreviate

$$||_{k \in \omega}(\langle s_i \rangle_{i \in I}, \#) \stackrel{def}{=} !\,(\langle s_i \rangle_{i \in I}, \#).$$

One easily observes that

$$!\,(\langle s_i \rangle_{i \in I}, \#) = (\langle s_{(n,i)} \rangle_{(n,i) \in \omega \times I}, !\#)$$

where $!\#$ is the binary relation over $\omega \times I$ such that $(n, i) \,!\# \, (m, i')$ iff $n = m$ and $i \# i'$.

We define the sum of strand spaces over a set of indexes K in a similar way as we did for the parallel composition of strand spaces over K. The indexing set H and the strands remain the same. Define

$$\sum_{k \in K} S_k \stackrel{def}{=} (\langle s_h \rangle_{h \in H}, \#)$$

where $(k, i) \# (k', i')$ iff either $k \neq k'$ or ($k = k'$ and $i \#^k i'$).

5 A process language for strand spaces

The constructions we have shown in the previous section form a language $\mathcal{S}$ of strand spaces with the following grammar:

$$S \ ::= \ \epsilon \mid L \mid \sum_{j \in J} S_j \mid ||_{j \in J} S_j$$

where $L \in \mathcal{L}$, the language of "sequential strand spaces" given by

$$L \ ::= \ \langle \lambda \rangle \mid \alpha.L \mid \sum_{j \in J} L_j \ .$$

The strand space $\langle \lambda \rangle$ has only one strand which is the empty sequence of actions and with the empty conflict relation.[3] The bundles of strand spaces in $\mathcal{L}$ form linearly ordered sets of events, and therefore can be thought of as runs of a sequential process.

A strand-space term of language $\mathcal{S}$ is a "par" process in the sense that parallel composition is only at the top level and therefore consists of a parallel composition and sum of sequential processes. Of particular interest are "!-par" processes which are those terms of $\mathcal{S}$ of the form $!S$. As shown in Section 8 conflict can be eliminated from such strand spaces.

It will be useful to weaken the definition of bundle to the one of *open bundle*, so that bundles can be composed. An open bundle is a graph with the same structure of a bundle, but where input events need not necessarily be related to output events. In this sense the open bundle is "open" to the environment for communication on input events that are not already linked to output events.

Definition 5.1 An open bundle b of a strand space $(\langle s_i \rangle_{i \in I}, \#)$ is a finite, acyclic subgraph of G graph of $(\langle s_i \rangle_{i \in I}, \#)$ such that:

(i) if $e \Rightarrow_G e'$ and $e' \in E_b$ then $e \Rightarrow_b e'$, *(control precedence)*

(ii) if $e' \rightarrow_b e$ and $e'' \rightarrow_b e$ then $e' = e''$, *(output-input correspondence)*

(iii) if $e, e' \in E_b$ s.t. $act(e) = out\, new \vec{n} M$ and $n \in \vec{n} \cap names(e')$ then either $e \Rightarrow_b^* e'$ or there exists an input event $e'' \in E_b$ such that $n \in names(e'')$, $e'' \not\leq_b e$ and $e'' \Rightarrow_b^* e'$, *(open freshness)*

(iv) if $(i, h), (j, k) \in E_b$ then $\neg(i \# j)$. *(conflict freeness)*

[3] Let the index of the empty strand in $\langle \lambda \rangle$ be a distinguished index $*$.

Proposition 3.4 does not necessarily hold for open bundles. Input events are open to communication from the environment and therefore not necessarily linked to any output event. The additional requirement $e'' \not\leq_b e$ in point (iii) ensures that $e' \not\leq_b e$; so an open bundle can be transformed into a bundle by composition with other open bundles and addition of communication edges.

Proposition 5.2 *If b is open bundle of a strand space in $\mathcal{L}$ then all events in E_b have the same index and $\Rightarrow_b$ is a linear order on E_b.*

Proof. A strand space $(\langle s_i \rangle_{i \in I}, \#)$ denoted by a term in $\mathcal{L}$ is such that for all indexes $i, j \in I$ if $i \neq j$ then $i \# j$. Open bundles are conflict free, therefore all events in E_b have the same index. Let i be the index of the events in E_b and let (i, h) be the event in E_b with greatest position index h. The control precedence of b determines its events to be

$$E_b = \{(i, l) \mid 1 \leq l \leq h\} \ .$$

The events in E_b are linearly ordered by $\Rightarrow_G$ and therefore linearly ordered by $\Rightarrow_b$ (control precedence). □

6 Bundle spaces

The usual semantics of a strand space is in terms of its set of bundles. In this section we show how, by broadening to open bundles, the bundle space can be constructed in a compositional way from bundle spaces. As shown in Section 7 an interesting congruence relation between strand spaces is based on open bundles rather than bundles and the compositional account of the bundle space presented here is useful in showing that such relation is indeed a congruence.

First we introduce two simple operations on open bundles. The first operation takes an open bundle of a strand space and transforms it into an open bundle of that space prefixed by an action. This operation simply adds an initial node and leaves all the remaining graph structure of the original open bundle intact. More precisely let $\alpha.S$ be a strand space in $\mathcal{L}$ and b open bundle of S with at least one event. All events of b share the same index i and are linearly ordered (Proposition 5.2). Control precedence and the shape of the strand space graph of S determines the events of b to be the set

$$E_b = \{(i, h) \mid 1 \leq h \leq k\}$$

and determines $\Rightarrow_b$ to be the smallest binary relation on E_b such that

$$(i,h) \Rightarrow_b (i,h+1) \text{ for } 1 \leq h < k$$

for some $k \geq 1$. Consider the graph

$$b^+ = (E_b^+, \Rightarrow_b^+, \rightarrow_b^+)$$

where

- $E_b^+ = \{(i,h) \mid h \leq k+1\}$,
- $(i,h) \Rightarrow_b^+ (i,h+1)$ for $h \leq k$ and $e \not\Rightarrow_b^+ e'$, otherwise
- $(i,h+1) \rightarrow_b^+ (i,k+1)$ iff $(i,h) \rightarrow_b (i,k)$.

Proposition 6.1 *The graph b^+ is an open bundle of $\alpha.S$.*

Proof. Straightforward. □

The second operation that we introduce shows how to juxtapose open bundles of strand spaces to get an open bundle of the composed space. Take the strand space $||_{j \in J} S_j$ and a finite subset I of J. For every $i \in I$ let b_i be an open bundle of S_i. Define

$$||_{i \in I} b_i = (\bigcup_{i \in I} E_i, \bigcup_{i \in I} \Rightarrow_i, \bigcup_{i \in I} \rightarrow_i)$$

where for each $i \in I$

- $E_i = \{i : e \mid e \in E_{b_i}\}$,
- $i : e \Rightarrow_i i : e'$ iff $e \Rightarrow_{b_i} e'$,
- $i : e \rightarrow_i i : e'$ iff $e \rightarrow_{b_i} e'$.

Tagging an event e with an index i is denoted by $i : e$ and is the event with index the pair consisting of i and the index of e.

Proposition 6.2 *The graph $||_{i \in I} b_i$ is an open bundle of $||_{j \in J} S_j$.*

Proof. Straightforward. □

Let b and d be two open bundles of the same strand space. We write $b \preceq d$ if d can be obtained from b by adding more communication edges. More precisely define

$$b \preceq d \text{ iff } E_b = E_d,\ \Rightarrow_b = \Rightarrow_d, \quad \text{and} \quad \rightarrow_b \subseteq \rightarrow_d\ ,$$

and abbreviate $b\uparrow = \ \{d \mid b \preceq d\}$. In the following definition we make use of a further shorthand notation $j : B$ which stands for $\{j : b \mid b \in B\}$ where j is an index, B a set of bundles and $j : b$ is the bundle obtained by tagging all events of b with j and extending the bundle-edges accordingly.

Definition 6.3 (Bundle space) Let S be a term from the language of strand spaces denoting the strand space $(\langle s_i \rangle_{i \in I}, \sharp)$. The language $\mathcal{B}(S)$ is defined on the structure of S as follows:

$$\begin{aligned}
\mathcal{B}(\epsilon) &= \{\lambda\} \\
\mathcal{B}(\langle\lambda\rangle) &= \{\lambda\} \\
\mathcal{B}(\alpha.S) &= \{\lambda\} \ \cup\ \{(\{(i,1)\}, \emptyset, \emptyset) \mid i \in I\} \ \cup \bigcup_{b \in \mathcal{B}(S)\setminus\{\lambda\}} b^+\uparrow \\
\mathcal{B}(\sum_{j \in J} S_j) &= \bigcup_{j \in J} j : \mathcal{B}(S_j) \\
\mathcal{B}(||_{j \in J} S_j) &= \bigcup_{\substack{I \subseteq_f J \\ b_i \in \mathcal{B}(S_i)}} (||_{i \in I} b_i)\uparrow
\end{aligned}$$

Theorem 6.4 *If S is a strand-space term in $\mathcal{S}$ then the elements of $\mathcal{B}(S)$ are exactly the open bundles of the strand space denoted by S.*

Proof. Let S be the strand-space term. The proof has two parts:

1. If b is an open bundle of S then $b \in \mathcal{B}(S)$. By induction on the structure of S:

   If $S = \epsilon$ or $S = \langle\lambda\rangle$ then $b = \lambda$ and obviously $\lambda \in \mathcal{B}(S)$.

   If $S = \alpha.S'$ then by Proposition 5.2 all events in b share the same index and are linearly ordered. By control precedence and the shape of the strand-space graph of S the events of b are either the empty set or are $E_b = \{(i,h) \mid h \leq k\}$ for some $k \geq 1$ with common index i. If the set of events of b is empty then $b = \lambda$ and

$\lambda \in \mathcal{B}(\alpha.S')$. The same holds for open bundles with only one node (when $k = 1$). Suppose that $k > 1$ and let G' be the strand-space graph of S'. Consider the graph b' with components

- $E_{b'} \;=\; \{(i,h) \mid h \leq k-1\}$,
- $\Rightarrow_{b'} = \Rightarrow_{G'} \;\cap (E_{b'} \times E_{b'})$,
- $\rightarrow_{b'}$ such that $\rightarrow_{b'}^{+} = \rightarrow_{b} \;\cap (E_{b'}^{+} \times E_{b'}^{+})$.

This graph is an open bundle of S'. In fact b' is a finite and acyclic subgraph of G' satisfying all requirements of Definition 5.1. In particular open freshness follows from that of b and from the linear ordering of the events in b'. Output-input correspondence and conflict freeness hold since b' is a subgraph of b and control precedence follows easily from the shape of $E_{b'}$ and $\Rightarrow_{b'}$. By the induction hypothesis $b' \in \mathcal{B}(S')$ and from $b \in b'^{+}\uparrow$ it follows that $b \in \mathcal{B}(\alpha.S')$.

If $S = \sum_{j \in J} S_j$ then by conflict freeness b has the form $j : b'$ for some index $j \in J$ and open bundle b' of S_j. From the induction hypothesis it follows that $b' \in \mathcal{B}(S_j)$ and therefore $j : b' \in \mathcal{B}(\sum_{j \in J} S_j)$.

Let $S = ||_{j \in J} S_j$ and for every $j \in J$ let G_j be the strand-space graph of $j : S_j$ (obtained via a re-indexing that adds j to each index of S_j) and let $j : b_j$ be the graph with components

- $E_j = E_b \cap E_{G_j}$,
- $\Rightarrow_j = \Rightarrow_b \cap \Rightarrow_{G_j}$,
- $\rightarrow_j = \rightarrow_b \cap \rightarrow_{G_j}$.

The graph b_j is an open bundle of S_j; it is a finite and acyclic subgraph of G_j satisfying all requirements of Definition 5.1. In particular open freshness of b_j follows from that of b since $j : b_j$ is a subgraph of b which satisfies control precedence. By the induction hypothesis $b_j \in \mathcal{B}(S_j)$ for all $j \in J$ and from $b \in (||_{j \in J} b_j)\uparrow$ it follows that $b \in \mathcal{B}(||_{j \in J} S_j)$.

2. Every $b \in \mathcal{B}(S)$ is an open bundle of the strand space denoted by S. By induction on the structure of S:

   If $S = \epsilon$ or $S = \langle \lambda \rangle$ then $b = \lambda$. The empty graph λ is an open bundle of every strand space.

If $S = \alpha.S'$ then either $b = \lambda$ or it has only one node, therefore b is an open bundle of the strand space $\alpha.S'$, or instead $b \in b'^{+}\uparrow$ for some $b' \in \mathcal{B}(S')$. In this last case from the induction hypothesis it follows that b' is an open bundle of S' and by Proposition 6.1 that b'^{+} is an open bundle of $\alpha.S'$. From $b \in b'^{+}\uparrow$ it follows that $b'^{+} \preceq b$ and therefore b is an open bundle of $\alpha.S'$.

If $S = \sum_{j\in J} S_j$ then there exist $j \in J$ and $b' \in \mathcal{B}(S_j)$ such that $b = j : b'$. By the induction hypothesis b' is an open bundle of S_j and therefore $j : b'$ is an open bundle of $\sum_{j\in J} S_j$.

If $S = ||_{j\in J} S_j$ then $b \in (||_{i\in I} b_i)\uparrow$ such that $b_i \in \mathcal{B}(S_i)$ for every $i \in I$. By the induction hypothesis for every $i \in I$ the graph b_i is an open bundle of S_i. Thus by Proposition 6.2 the graph $||_{i\in I} b_i$ is an open bundle of $||_{j\in J} S_i$. From $||_{i\in I} b_i \preceq b$ it follows that b is an open bundle of $||_{i\in I} S_i$.

□

7 A strand space congruence

We have seen an equivalence relation that relates two strand spaces if, via re-indexing, they become the same space. It is easy to check that this relation is a congruence with respect to the operations of the strand-space language we introduced in this paper. It is however a very concrete relation and too discriminating for a model in which security properties are expressed as safety properties on the language of bundles of a strand space. One doesn't want to distinguish between strand spaces that have isomorphic bundle languages. Unfortunately the equivalence relation $\approx$ on strand space terms, obtained by taking term equivalence iff they denote strand spaces with essentially the same bundles, is not a congruence. In this section we study a finer equivalence that takes account of the open bundles of a strand space rather than bundles. This relation turns out to be an interesting congruence, in fact the largest congruence within $\approx$.

Two bundles and more generally two open bundles are isomorphic if they are isomorphic graphs:

Definition 7.1 Given b an open bundle of a strand space S and given b' an open bundle of a strand space S' define $b \cong b'$ iff there exists a bijection $\phi : E_b \to E_{b'}$ such that

(i) if $e \rightarrow_b e'$ then $\phi(e) \rightarrow_{b'} \phi(e')$,

(ii) if $e \Rightarrow_b e'$ then $\phi(e) \Rightarrow_{b'} \phi(e')$,

(iii) $act_S(e) = act_{S'}(\phi(e))$.

Sometimes we write $\phi(b)$ for the open bundle obtained from b through ϕ. This open bundle has events $\phi(E_b)$ and edges

- $\phi(e) \rightarrow_{\phi(b)} \phi(e')$ iff $e \rightarrow_b e'$,
- $\phi(e) \Rightarrow_{\phi(b)} \phi(e')$ iff $e \Rightarrow_b e'$.

Definition 7.2 Let S and S' be two strand-space terms in $\mathcal{S}$. Define $\approx$ the symmetric relation such that $S \approx S'$ iff for every bundle b of S there exists a bundle b' of S' such that $b \cong b'$.

Proposition 7.3 *The relation $\approx$ is an equivalence relation.*

Proof. Straightforward. □

The equivalence relation $\approx$ is not a congruence relation. Consider, for example, the strand-space terms $in\, M.\epsilon$ and $in\, N.\epsilon$ where N and M are two different messages. These two strand-space terms are in the relation $\approx$ – they both have only one bundle, the empty bundle and they can be easily distinguished in a simple context when, for example, composed in parallel with $out\, M.\epsilon$. Then

$$in\, M.\epsilon \mid\mid out\, M.\epsilon \not\approx in\, N.\epsilon \mid\mid out\, M.\epsilon\ .$$

The parallel composition on the left hand side has a bundle of the form

$$in\, M \longleftarrow out\, M$$

(for convenience in the previous bundle we show the action associated to the nodes instead of the nodes themselves). The parallel composition on the right hand side only allows the empty bundle.

A context for a strand-space term in the language $\mathcal{S}$ is defined as follows:

$$C ::= [\ ] \mid \alpha.C \mid ||_{i \in I} T_i \mid \Sigma_{i \in I} T_i$$

where for each context of the form $||_{i \in I} T_i$ or $\Sigma_{i \in I} T_i$ there is exactly one $i \in I$ such that T_i is a context C and $T_i \in \mathcal{S}$ for all $i \in I \setminus \{i\}$. The context $[\ ]$ is a placeholder for a strand-space term. We write $C[S]$ for the term obtained by replacing the strand-space term S for $[\ ]$ in context C in the obvious way. An equivalence relation on strand-space terms is a congruence if it respects all contexts.

Definition 7.4 Let S and S' be two strand-space terms in $\mathcal{S}$. Define $\approx_{\mathcal{B}}$ to be the symmetric relation such that $S \approx_{\mathcal{B}} S'$ iff for every open bundle b of S there exists an open bundle b' of S' such that $b \cong b'$.

Proposition 7.5 *The relation $\approx_{\mathcal{B}}$ is a congruence.*

Proof. The relation $\approx_{\mathcal{B}}$ is obviously an equivalence relation. We show by induction on the structure of contexts that if $S \approx_{\mathcal{B}} S'$ then $C[S] \approx_{\mathcal{B}} C[S']$ for every context C.

Obvious for the context [ ].

Consider the context $\alpha.C$ and let $b \in \mathcal{B}(\alpha.C[S])$. From Definition 6.3 of bundle space it follows that b is one of the following graphs:

- λ and therefore belongs to $\mathcal{B}(\alpha.C[S'])$.
- $(\{(i,1)\}, \emptyset, \emptyset)$ with i index of $\alpha.C[S]$. Then $b \cong (\{(j,1)\}, \emptyset, \emptyset)$ and $(\{(j,1)\}, \emptyset, \emptyset) \in \mathcal{B}(\alpha.C[S'])$ for any j index of $\alpha.C[S']$.
- $b \in b_1^+ \uparrow$ for some $b_1 \in \mathcal{B}(C[S])$. By the induction hypothesis there exists $b_2 \in \mathcal{B}(C[S'])$ such that $b_1 \cong b_2$ and therefore $b_1^+ \cong b_2^+$. If $\phi : E_{b_1^+} \to E_{b_2^+}$ is a bijection such that $\phi(b_1^+) = b_2^+$ then $\phi(b) \in b_2^+ \uparrow$.

Consider the context $\Sigma_{i \in I} T_i$ such that the term T_{i_0} is a context C for the index i_0 in I. Let $b \in \mathcal{B}(\Sigma_{i \in I} T_i[S])$. By Definition 6.3 of bundle space the open bundle b is of the form $i : b_i$ for some $i \in I$ and for some $b_i \in \mathcal{B}(T_i[S])$. If $i \neq i_0$ then $i : b_i \in \mathcal{B}(\Sigma_{i \in I} T_i[S'])$. If instead $i = i_0$ then by the induction hypothesis there exists $b'_i \in \mathcal{B}(C[S'])$ such that $b_i \cong b'_i$ and therefore $i : b'_i \in \mathcal{B}(\Sigma_{i \in I} T_i[S'])$ and $i : b_i \cong i : b'_i$.

Consider the context $||_{i \in I} T_i$ such that the term T_{i_0} is a context C for the index i_0 in I. If $b \in \mathcal{B}(||_{i \in I} T_i[S])$ then $b \in (||_{j \in J} j : b_j) \uparrow$ where $b_j \in \mathcal{B}(T_j)$ when $j \neq i_0$ and $b_j \in \mathcal{B}(C[S])$ for $j = i_0$. By the induction hypothesis there exists $b'_{i_0} \in \mathcal{B}(C[S'])$ such that $b_{i_0} \cong b'_{i_0}$. Let $b'_j = b_j$ for all $j \neq i_0$. It follows that $||_{j \in J} b'_j \in \mathcal{B}(||_{i \in I} T_i[S'])$ and $||_{j \in J} b_j \cong ||_{j \in J} b'_j$. If ϕ is the bijection such that $\phi(||_{j \in J} b_j) = ||_{j \in J} b'_j$ then $\phi(b) \in \mathcal{B}(||_{j \in J} b'_j)$. $\square$

Theorem 7.6 *The relation $\approx_{\mathcal{B}}$ is the largest congruence relation inside $\approx$.*

Proof. Consider the set

$$D = \{R \mid R \text{ congruence relation and } R \subseteq \approx\}$$

Clearly $\bigcup D$ is a congruence relation and the largest congruence inside $\approx$. It remains to show that $\approx_{\mathcal{B}} = \bigcup D$. By Proposition 7.5 the relation $\approx_{\mathcal{B}}$ is a congruence and since $\approx_{\mathcal{B}} \subset \approx$ it follows that $\approx_{\mathcal{B}} \subseteq \bigcup D$. Let S and S' be two strand-space terms and let b be an open bundle of S such that $b \not\cong b'$ for all open bundles b' of S'. If no congruence relation in D contains the pair (S, S') then $\bigcup D \subseteq \approx_{\mathcal{B}}$. Let $R \in D$ and suppose $(S, S') \in R$. We find a context C such that $(C[S], C[S']) \notin R$ and therefore R is not a congruence relation. Consider the strand space T composed of a single strand whose only events are those output events that correspond exactly to the open input events of b. The strand space T itself forms a bundle that we denote by t. Consider the context $T \mid\mid [\ ]$. Let $b_1 \in (t||b)\uparrow$ be the graph that has no open inputs – it is a bundle of $T \mid\mid S$. If there is a bundle b_2 of $T \mid\mid S'$ such that $b_1 \cong b_2$ then $b_2 \in (t'||b')\uparrow$ for some open bundles b' and t' such that $b \cong b'$ and $t \cong t'$. By control precedence either t' is a bundle of T, therefore b' is an open bundle of S' contradicting our assumptions, or b_2 is a bundle of S', therefore b' is an open bundle of S' and we reach a contradiction again. $\square$

8 Eliminating conflict

An agent that participates in a security protocol is often thought of as executing a sequential program during which it sends messages and chooses from a number of available messages which one to input. If one doesn't restrict the number of rounds of the protocol an agent can do one can hope to model the protocol with a strand space with conflict of the form $!\,(\langle s_i \rangle_{i \in I}, \#)$. In this section we show that under these conditions a simpler model, obtained by dropping the conflict relation, exhibits substantially the same behaviour as the more complex strand space with conflict.

Consider a conflict relation $\#$ on a set I and consider the relation $!\#$ on $\omega \times I$ such that $(n, i)\, !\#\, (m, j)$ iff $n = m$ and $i \# j$. The form of the $!\#$ conflict relation suggests the following lemma:

Lemma 8.1 *Given I a set of indexes, a finite set $A \subseteq \omega \times I$, and $\#$ a conflict relation over I. There exists a bijection $\sigma : \omega \times I \to \omega \times I$ such that*

$$\forall (n, i) \in \omega \times I \,.\, \exists m \in \omega \,.\, \sigma(n, i) = (m, i)$$

and such that the set A is conflict free with respect to $\sigma^{-1}(!\#)$, meaning $\neg(\sigma(u) \mathrel{!\#} \sigma(v))$ for all $u, v \in A$.

Proof. By induction on the size of the set A.

The basic case $A = \emptyset$: take σ for example to be the identity function.

The induction step: Given a set A suppose there is a bijection σ satisfying the desired property and such that A is conflict free with respect to $\sigma^{-1}(!\#)$. Take $A' = A \cup \{(n', i')\}$ with $(n', i') \notin A$. Let $\sigma(n', i') = (m', i')$ and distinguish two cases:

If $m \neq m'$ for all $(m, j) \in \sigma(A)$ then A' is conflict free with respect to $\sigma^{-1}(!\#)$. Suppose the contrary. Suppose there is $(n, i) \in A$ such that $\sigma(n, i) \mathrel{!\#} \sigma(n', i')$. If $\sigma(n, i) = (m, i)$ then $m = m'$ by definition of $!\#$.

If there exists $(m, j) \in \sigma(A)$ such that $m = m'$, then consider the following function:

$$\sigma'(n, i) = \begin{cases} (m'', i') & \text{if } (n, i) = (n', i') \\ (m', i') & \text{if } (n, i) = \sigma^{-1}(m'', i') \\ \sigma(n, i) & \text{otherwise} \end{cases}$$

where $m'' \in \omega$ such that $m \neq m''$ for all $(m, j) \in \sigma(A)$. Since A is a finite set, such m'' exists. This function is as σ but swaps the new element (n', i'), that could introduce conflict, with one that doesn't. It is easy to check that σ' is a bijection and that it satisfies the required property. It remains to check that A' is conflict free with respect to ${\sigma'}^{-1}(!\#)$. First observe that $\sigma'(A) = \sigma(A)$ because $(n', i') \notin A$ and $\sigma^{-1}(m'', i') \notin A$ (we chose m'' so that $(m'', i') \notin \sigma(A)$). It follows that A is conflict free with respect to ${\sigma'}^{-1}(!\#)$. Since ${\sigma'}^{-1}(!\#)$ is irreflexive and symmetric we require:

$$\forall (n, i) \in A \,.\, \neg(\sigma'(n, i) \mathrel{!\#} \sigma'(n', i')).$$

Suppose instead that there exists $(n, i) \in A$ such that $\sigma(n, i) \mathrel{!\#} \sigma(n', i')$. Let $\sigma(n, i) = (m, i)$, then $(m, i) \mathrel{!\#} (m'', i')$. From the conflict relation $!\#$ it follows that $m = m''$. On the other hand $(m, i) \in \sigma(A)$. Therefore $m \neq m''$.

□

Together with the previous lemma, the observation that two different copies of the same strand space have same strands at same corresponding positions, yields the following theorem:

Theorem 8.2 *Consider strand spaces $!\,(\langle s_i \rangle_{i \in I}, \emptyset)$ and $!\,(\langle s_i \rangle_{i \in I}, \#)$. Let b be a bundle of $!\,(\langle s_i \rangle_{i \in I}, \emptyset)$. There exists a strand space S such that b is a bundle of S and $S \cong !\,(\langle s_i \rangle_{i \in I}, \#)$.*

Proof. Given b a bundle of $!\,(\langle s_i\rangle_{i\in I}, \emptyset)$ take A the set of indexes of strands participating with nodes in b. The set A is finite because so is b and $A \subseteq \omega \times I$. By Lemma 8.1 there is a bijection $\sigma : \omega \times I \to \omega \times I$ such that

$$\forall (n,i) \in \omega \times I \,.\, \sigma(n,i) = (n', i)$$

for some $n' \in \omega$ and A is conflict free with respect to $\sigma^{-1}(!\#)$. Consider the strand space with conflict

$$S = (\langle t_{(m,i)}\rangle_{(m,i)\in\omega\times I}, \sigma^{-1}(!\#))$$

where $t_{(m,i)} = s_i$ for all $m \in \omega$. The equivalence $S \cong !(\langle s_i\rangle_{i\in I}, \#)$ stands. In fact σ is a bijection such that

(i) $t_{(m,i)} = s_{\sigma(m,i)}$ since $s_{\sigma(m,i)} = s_{(n',i)} = s_i = t_{(m,i)}$

(ii) $(m,i)\,\sigma^{-1}(!\#)\,(u,j)$ iff $\sigma(m,i)\,!\#\,\sigma(u,j)$.

The two strand spaces $!\,(\langle s_i\rangle_{i\in I}, \emptyset)$ and S have the same strand-space graph, therefore since b is a bundle over $!(\langle s_i\rangle_{i\in I}, \emptyset)$, and since A is conflict free with respect to $\sigma^{-1}(!\#)$ follows b bundle of S. □

To summarise, the behaviour in terms of bundles of a replicated strand space with conflict corresponds, modulo re-indexing, to that of the strand space we obtain dropping the conflict relation.

Corollary 8.3 *Consider strand spaces* $!(\langle s_i\rangle_{i\in I}, \emptyset)$ *and* $!(\langle s_i\rangle_{i\in I}, \#)$ *strand spaces.*

1. *If* b *is bundle of* $!\,(\langle s_i\rangle_{i\in I}, \#)$ *then* b *is bundle of* $!\,(\langle s_i\rangle_{i\in I}, \emptyset)$.
2. *If* b *is bundle of* $!\,(\langle s_i\rangle_{i\in I}, \emptyset)$ *then there exists a re-indexing* π *such that* $\pi(b)$ *is a bundle of* $!\,(\langle s_i\rangle_{i\in I}, \#)$.

Proof. The first point is obvious, the second point follows directly form Theorem 8.2 and Proposition 3.6. □

Consequently, the strand space with conflict denoted by a "!-par" process has the same bundles up to re-indexing as a strand space without conflict.

9 Event structures from strand spaces

In this section we show how strand spaces relate to event structures. Recall that a bundle of a strand space is a graph and therefore a set of edges and nodes. It turns out that the bundles of a strand space ordered by inclusion correspond to the finite configurations of a prime event structure. Prime event structures are a particularly simple kind of event structure where the enabling of events can be expressed as a global partial order of causal dependency. Event structures as a model of concurrent processes were introduced in [NPW81, Win80] – see also [Win82, Win87, Win88, WN95].

Definition 9.1 A prime event structure $(E, \#, \leq)$ consists of a set E of events partially ordered by the causal dependency relation $\leq$ and a binary, symmetric, irreflexive conflict relation $\# \subseteq E \times E$, which satisfy

(i) $\{e' \mid e' \leq e\}$ is finite for all $e \in E$, and

(ii) if $e \# e' \leq e''$ then $e \# e''$ for all $e, e', e'' \in E$.

Definition 9.2 The configurations of an event structure $E = (E, \#, \leq)$ consist of those subsets $x \subseteq E$ which are

(i) conflict free: $\forall e, e' \in x \,.\, \neg(e \# e')$ and

(ii) left closed: $\forall e, e'.e' \leq e \in x \Rightarrow e' \in x$.

Write $\mathcal{F}(E)$ for the set of configurations of an event structure and $\mathcal{F}^{fin}(E)$ for its finite configurations.

For a strand space S write $\mathcal{B}$ for the set of bundles of S. Consider the partial order $(\mathcal{B}, \subseteq)$. Say a subset X of $\mathcal{B}$ is *compatible* iff

$$\exists b \in \mathcal{B} \,.\, \forall b' \in X \,.\, b \subseteq b' \ .$$

Proposition 9.3 *Let S be a strand space and $\mathcal{B}$ the set of bundles of S. The partial order $(\mathcal{B}, \subseteq)$ satisfies the following properties:*

1. *if $X \subseteq \mathcal{B}$, X is finite and pairwise compatible, then $\bigcup X \in \mathcal{B}$.* *(coherence)*

2. *if $X \subseteq \mathcal{B}$ and X is compatible, then $\bigcap X \in \mathcal{B}$.* *(stability)*

Proof. Let G be the strand space graph of S.

1. The graph $\bigcup X$ is obviously a finite subgraph of G since X is finite and each element in X is finite. Let e be an event in $\bigcup X$. The set X is pairwise compatible therefore there exists $b \in X$ with $e \in b$ and such that if $e' \rightarrow_{\bigcup X} e$ then $e' \rightarrow_b e$. Since b is a bundle if $e' \Rightarrow_{\bigcup X} e$ then $e' \Rightarrow_b e$ (control precedence). Therefore $\{e' \mid e' \leq_{\bigcup X} e\} = \{e' \mid e' \leq_b e\}$ and it follows that $\bigcup X$ is acyclic. We check the requirements of Definition 3.2:

   (i) Consider an event $e \in \bigcup X$. There is a bundle $b \in X$ with $e \in b$. By control precedence if $e' \Rightarrow_G e$ then $e' \Rightarrow_b e$. From $b \subseteq \bigcup X$ it follows that $e' \Rightarrow_{\bigcup X} e$.

   (ii) Output-input precedence stands for the set $\bigcup X$. Let e be an input event in $\bigcup X$. There is a bundle $b \in \bigcup X$ with $e \in b$. An output event e' exists such that $e' \rightarrow_b e$ (output-input precedence of b) and therefore $e' \rightarrow_{\bigcup X} e$. We require that there is a unique such output event. Suppose the contrary. Suppose there are two output events e', e'' such that $e' \rightarrow_{\bigcup X} e$ and $e'' \rightarrow_{\bigcup X} e$. Then two distinct bundles $b, b' \in X$ exist such that $e' \rightarrow_b e$ and $e'' \rightarrow_{b'} e$. However X is pairwise compatible. Therefore there is a bundle in X containing both b and b'. This can't be the case and we reach a contradiction.

   (iii) Let e, e' be two events in $\bigcup X$ such that $act(e) = out\, new\, \vec{n}\, M$ and $n \in \vec{n} \cap names(e')$. Since X is pairwise compatible, there exists a bundle b' in X which contains both e and e'. The bundle b' respects freshness. Thus either $e \Rightarrow^*_{b'} e'$ or there exists an input event e'' such that $n \in names(e'')$ and $e'' \Rightarrow^*_{b'} e'$. Consequently freshness holds for $\bigcup X$.

   (vi) That the graph $\bigcup X$ does not contain conflicting events follows easily from the pairwise compatibility of X.

   Therefore $\bigcup X$ is a bundle in $\mathcal{B}$.

2. Every element of X is a bundle in $\mathcal{B}$. Therefore $\bigcap X$ forms a finite and acyclic subgraph of G. Following Definition 3.2 we check all requirements and thus show that $\bigcap X \in \mathcal{B}$:

   (i) Consider an event $e \in \bigcap X$. Every bundle $b \in X$ is such that $e \in b$. For each $b \in X$ if $e' \Rightarrow_G e$ then $e' \Rightarrow_b e$ (control precedence). From this it follows that $e' \Rightarrow_{\bigcap X} e$.

(ii) If e is an input event in $\bigcap X$ then every bundle $b \in \bigcap X$ contains the event e and contains an output event e_b such that $e_b \rightarrow_b e$. The set X is compatible. Let $b' \in \mathcal{B}$ be an upper bound. Therefore there exists a unique e' such that $e' = e_b$ for all bundles $b \in X$ and $e' \rightarrow_b e$. The event e' is the unique event such that $e' \rightarrow_{\bigcap X} e$.

(iii) Let e, e' be two events in $\bigcap X$ such that $act(e) = out\, new\, \vec{n}\, M$ and $n \in \vec{n} \cap names(e')$. The events e, e' belong to every bundle in X. Freshness holds for each bundle b in X. Thus either $e \Rightarrow_b^* e'$ or there exists an input event e'' such that $n \in names(e'')$ and $e'' \Rightarrow_b^* e'$. If $e \Rightarrow_{b'}^* e'$ for some bundle b' in X then $e \Rightarrow_G^* e$ and therefore $e \Rightarrow_b^* e'$ for every bundle b in X. The case of a bundle b' such that $e'' \Rightarrow_b^* e'$ is similar. Freshness of $\bigcap X$ follows.

(vi) The elements of X are bundles and therefore conflict free. Consequently $\bigcap X$ is conflict free.

□

Given a bundle $b \in \mathcal{B}$ and an event $e \in E_b$ define

$$\lceil e \rceil_b \stackrel{def}{=} \bigcap \{b' \in \mathcal{B} \mid e \in b' \ \wedge \ b' \subseteq b\} \ .$$

Proposition 9.4 *Let $\mathcal{B}$ be the bundles of a strand space. For every bundle $b \in \mathcal{B}$ and every event $e \in E_b$ the set $\lceil e \rceil_b$ is a bundle in $\mathcal{B}$. For every finite and compatible subset $X \subseteq \mathcal{B}$*

$$\textit{if } \lceil e \rceil_b \subseteq \bigcup X \textit{ then } \exists b' \in X \,.\, \lceil e \rceil_b \subseteq b'$$

We call a bundle $\lceil e \rceil_b$ a prime of $(\mathcal{B}, \subseteq)$.

Proof. That the elements $\lceil e \rceil_b$ are bundles in $\mathcal{B}$ follows from the stability property of $(\mathcal{B}, \subseteq)$ (Proposition 9.3). The set $\{b' \in \mathcal{B} \mid e \in b' \ \wedge \ b' \subseteq b\}$ is in fact compatible.

Let $X \subseteq \mathcal{B}$ be finite and compatible. If $\lceil e \rceil_b \subseteq \bigcup X$ then $e \in b'$ for some $b' \in X$. The two bundles b' and $\lceil e \rceil_b$ in $\mathcal{B}$ are compatible (they are both included in $\bigcup X$), therefore from the stability property of $(\mathcal{B}, \subseteq)$ it follows that $b' \cap \lceil e \rceil_b \in \mathcal{B}$. We know that $e \in b'$ and that $b' \cap \lceil e \rceil_b \subseteq b$. Therefore $\lceil e \rceil_b \subseteq b' \cap \lceil e \rceil_b \subseteq b'$. □

The primes form a basis for the partial order $(\mathcal{B}, \subseteq)$ as the following proposition shows.

Proposition 9.5 *Let S be a strand space and $\mathcal{B}$ its bundles. Every bundle $b \in \mathcal{B}$ can be obtained as the union of the primes included in b*

$$b = \bigcup \{p \mid p \subseteq b,\ p \text{ prime}\}.$$

Proof. Obviously $\bigcup\{p \mid p \subseteq b,\ p \text{ prime}\} \subseteq b$. On the other hand, if $e \in b$ then $e \in \lceil e \rceil_b$. The prime $\lceil e \rceil_b$ is a bundle included in b and therefore if $e' \Rightarrow_b e$ then $e' \Rightarrow_{\lceil e \rceil_b} e$, and if $e' \rightarrow_b e$ then $e' \rightarrow_{\lceil e \rceil_b} e$. Thus $b \subseteq \bigcup\{\lceil e \rceil_b \mid e \in b\}$. □

Let $\mathcal{B}$ the set of bundles of a strand space S. Consider the following structure

$$Pr(\mathcal{B}) \stackrel{def}{=} (P, \#, \subseteq)$$

where P is the set of primes of $\mathcal{B}$ and where $p \# p'$ iff the two primes p and p' are not compatible.

Theorem 9.6 *The structure $Pr(\mathcal{B})$ is a prime event structure. The map $\phi : (\mathcal{B}, \subseteq) \cong (\mathcal{F}^{fin} Pr(\mathcal{B}), \subseteq)$ such that $\phi(b) = \{p \mid p \subseteq b,\ p \text{ prime}\}$ is an isomorphism of partial orders with inverse map $\theta : \mathcal{F}^{fin} Pr(\mathcal{B}) \rightarrow \mathcal{B}$ given by $\theta(x) = \bigcup x$.*

Proof. It is easy to check that $Pr(\mathcal{B})$ is a prime event structure. The relation $\subseteq$ is a partial order of primes. Bundles and in particular primes are finite sets. Thus for each prime p the set $\{p' \mid p' \subseteq p\}$ is finite. Moreover if p, p', p'' are primes such that $p \# p' \subseteq p''$ then $p \# p''$ – otherwise p and p'' would be compatible and therefore p and p' would be compatible too.

The map ϕ is obviously well-defined and so is θ. In fact x is a finite configuration of $Pr(\mathcal{B})$ thus a finite set of bundles in $\mathcal{B}$ which is conflict free and left closed. Since it is conflict free it is pairwise compatible and therefore from the coherence property of Proposition 9.3 it follows that $\bigcup x \in \mathcal{B}$.

It is clear that both maps ϕ and θ are order preserving. We show that they are mutual inverses and therefore give the required isomorphism. From Proposition 9.5 it follows that $\theta(\phi(b)) = b$ for all $b \in \mathcal{B}$. Let x be a finite configuration of $Pr(\mathcal{B})$. We require that $\phi(\theta(x)) = x$, i.e., $\{p \mid p \subseteq \bigcup x,\ p \text{ prime}\} = x$. If p is a prime such that $p \subseteq \bigcup x$, then there exists a bundle p' in x (another prime) such that $p \subseteq p'$ (Proposition 9.4). The configuration x is left-closed, thus $p \in x$. On the other hand, x is a set of primes and therefore $x \subseteq \{p \mid p \subseteq \bigcup x,\ p \text{ prime}\}$. □

Observation 9.7 Since $Pr(\mathcal{B})$ is a prime event structure the partial order $(\mathcal{F}Pr(\mathcal{B}), \subseteq)$ is a Scott domain of information, in particular a prime algebraic domain (see [NPW81, Win87, Win88]). This domain however can include configurations which are infinite and therefore are not bundles in the usual sense.

The prime event structure $Pr(\mathcal{B})$ underlies the strand space semantics Syverson gave to the BAN logic [Syv99].

References

[CDL+00] I. Cervesato, N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. Relating strands and multiset rewriting for security protocol analysis. In P. Syverson, editor, *13th IEEE Computer Security Foundations Workshop - CSFW'00*, Cambridge, UK, 3-5 July 2000. IEEE Computer Society Press.

[CJ97] J. Clark and J. Jacob. A survey of authentication protocol literature: Version 1.0. 1997.

[CW01] F. Crazzolara and G. Winskel. Events in security protocols. In *Proceedings of the Eight ACM Conference on Computer and Communications Security*, Philadelphia, November 2001. ACM Press.

[DY83] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.

[NPW81] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, Event structures and Domains. *Theoretical Computer Science*, 13, 1981.

[Syv99] P. Syverson. Towards a Strand Semantics for Authentication Logic. *Electronic Notes in Theoretical Computer Science*, (20), 1999.

[TG00] J. Thayer and J. Guttman. Authentication tests. In *2000 IEEE Symposium on Security and Privacy*, Oakland CA, May 2000.

[THG98a] J. Thayer, J. Herzog, and J. Guttman. Honest ideals on strand spaces. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 1998.

[THG98b] J. Thayer, J. Herzog, and J. Guttman. Strand spaces: Why is a security protocol correct? In *1998 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1998.

[Win80] G. Winskel. *Events in computation*. PhD thesis, Comp. Sc. Dept. University of Edinburgh, 1980.

[Win82] G. Winskel. Event structure semantics of ccs and related languages. In *Proceedings of the International Colloquium on Automata, Languages and Programming - ICALP 82*, volume 140 of *LNCS*. Springer–Verlag, 1982. Extended version, Computer Science Report, University of Aarhus, 1982.

[Win87] G. Winskel. Event structures. In *Proceedings of the Advanced Course on Petri nets*, volume 255 of *LNCS*. Springer–Verlag, 1987.

[Win88] G. Winskel. An introduction to event structures. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*. Springer–Verlag, 1988.

[WN95] G. Winskel and M. Nielsen. *Models for concurrency*, volume IV. Oxford University Press, 1995.

Recent BRICS Report Series Publications

RS-02-5 Federico Crazzolara and Glynn Winskel. *Composing Strand Spaces*. February 2002. 30 pp.

RS-02-4 Olivier Danvy and Lasse R. Nielsen. *Syntactic Theories in Practice*. January 2002. 34 pp. This revised report supersedes the earlier BRICS report RS-01-31.

RS-02-3 Olivier Danvy and Lasse R. Nielsen. *On One-Pass CPS Transformations*. January 2002. 18 pp.

RS-02-2 Lasse R. Nielsen. *A Simple Correctness Proof of the Direct-Style Transformation*. January 2002.

RS-02-1 Claus Brabrand, Anders Møller, and Michael I. Schwartzbach. *The `<bigwig>` Project*. January 2002. 36 pp. This revised report supersedes the earlier BRICS report RS-00-42.

RS-01-55 Daniel Damian and Olivier Danvy. *A Simple CPS Transformation of Control-Flow Information*. December 2001.

RS-01-54 Daniel Damian and Olivier Danvy. *Syntactic Accidents in Program Analysis: On the Impact of the CPS Transformation*. December 2001. 41 pp. To appear in the *Journal of Functional Programming*. This report supersedes the earlier BRICS report RS-00-15.

RS-01-53 Zoltán Ésik and Masami Ito. *Temporal Logic with Cyclic Counting and the Degree of Aperiodicity of Finite Automata*. December 2001. 31 pp.

RS-01-52 Jens Groth. *Extracting Witnesses from Proofs of Knowledge in the Random Oracle Model*. December 2001. 23 pp.

RS-01-51 Ulrich Kohlenbach. *On Weak Markov's Principle*. December 2001. 10 pp.

RS-01-50 Jiří Srba. *Note on the Tableau Technique for Commutative Transition Systems*. December 2001. 19 pp. To appear in Nielsen and Engberg, editors, *Foundations of Software Science and Computation Structures*, FoSSaCS '02 Proceedings, LNCS 2303, 2002.