Protocol Algebra

Anders Moen Hagalisletto

April 6, 2005

Introduction

Security protocols belong to the core of computer security, combining classical disciplines like

- communication,
- cryptography and
- software engineering.

What are our motivation?

- build a meta-protocol for specification and execution of security protocols
- apply a local view on the modeling and simulation of the protocols
- make it easy to accommodate rapid prototyping and testing of new protocols
- investigate tacit assumptions in protocol descriptions
- make computations with protocols using abstract algebra

An important citation

Martin Abadi (1999):

> " make explicit the tacit implicit assumptions underlying specifications of security protocols."

Requirements of a protocol are of two kinds,

- the intended canonical tacit assumptions (what the protocol designer intend should happen),
- the unintended assumption (what the protocol designer did not think about).

Organization of the talk

1. Present Needham Schöeder Symmetric Key Authentication protocol
2. Define a formal **language for security specification**
3. Extend this language gently, to a **language for protocols**
4. Define formally **text-book protocols**
5. Present the **simulator** written in Maude
6. Investigate the **local assumptions** required in real life execution
7. Show how **abstract algebra** pops in everywhere through composition

Needham Schöeder Symmetric Key Authentication

Uses a trusted third party (TTP) to distribute keys.

$$
\begin{array}{lllll}
(P_1) & A \longrightarrow S & : & A, B, N_A \\
(P_2) & S \longrightarrow A & : & E(K_{AS} : N_A, B, K_{AB}, E(K_{BS} : K_{AB}, A)) \\
(P_3) & A \longrightarrow B & : & E(K_{BS} : K_{AB}, A) \\
(P_4) & B \longrightarrow A & : & E(K_{AB} : N_B) \\
(P_5) & A \longrightarrow B & : & E(K_{AB} : N_B - 1)
\end{array}
$$

(P_1) - (P_5) denotes a sequence or chain if *protocol sentences*. A linear sequence of protocol sentences forms a *protocol*:

(P_1) : initiation of the protocol (P_5) : the end of the protocol.

Every instance of the protocol sentence (P_i) should occur before (P_{i+1}). In case of (P_3), the clause $A \longrightarrow B : E(K_{BS} : K_{AB}, A)$ means that agent A sends to B the message content $E(K_{BS} : K_{AB}, A)$.

A language for protocols $\mathscr{L}_S^P$

The terms in $\mathscr{L}_S$: agents Agents and messages Msg.

Agents are a, b, c, ..., while messages are written m, m_1, m_2, ...

Two distinguished atomic sentences:

$\mathsf{Transmits}(a, b, m)$: "a sends the message m to b"
$\mathsf{Agent}(a)$: "a is an agent".

Definition 1 *Let a and b be agents and m a message, then $\mathscr{L}_S$ is the least language such that:*

(*i*) $\mathsf{Atomic} \subset \mathscr{L}_S$.

(*ii*) *If φ and ψ are $\mathscr{L}_S$ formulas, then $\neg\varphi$ and $\varphi \to \psi$ are $\mathscr{L}_S$ formulas.*

(*iii*) *If φ is a $\mathscr{L}_S$ formula, then so are the formulas $\mathsf{saysTo}_a^b(\varphi)$, $\mathsf{Bel}_a(\varphi)$, $\mathsf{Justify}_a(\varphi)$, and $\varphi \mathcal{U} \psi$.*

Relation between messages and sentences

$\mathsf{saysTo}_a^b(\varphi)$: "the agent a says φ to the agent b".

A single transfer of a message, we write $\mathsf{Transmits}(a, b, m)$.

If F is a sentence then we let $\ulcorner F \urcorner$ denote the sentence considered as a message.

The resulting expression $\ulcorner \ldots \urcorner$ is a total function, and is simply the plain text translation of the logical expression F.

Conversely, if m is a message then $\llcorner m \lrcorner$ gives a sentences if possible, else it is undefined, hence

$$\llcorner \ulcorner F \urcorner \lrcorner = F.$$

Explaining the operators in $\mathscr{L}_S$

$\mathsf{Bel}_a(\varphi)$: "the agent a believes φ holds".

$\mathsf{Justify}_a(\varphi)$: "the agent a has justified that φ is true".

$\varphi \,\mathcal{U}\, \psi$: "φ holds until ψ holds".

The operator *before* $\mathcal{B}$, is definable from $\mathcal{U}$ by

$$\varphi \,\mathcal{B}\, \psi = \neg(\neg\varphi \,\mathcal{U}\, \psi).$$

Single agent properties

The language can be used to define both *single agent properties*, *characters* Moen Haugsand (2004), and general *security properties*.

Examples of characters used in this paper are **honesty**, to be **indifferent** and **conscious**.

$\mathsf{says}_a(\varphi) \rightarrow \mathsf{Bel}_a(\varphi)$		Honest
$\mathsf{says}_a(\varphi) \rightarrow \mathsf{Bel}_a(\varphi) \vee \neg\, \mathsf{Bel}_a\,(\varphi)$		Indifferent
$\mathsf{says}_a(\varphi) \rightarrow \mathsf{Bel}_a(\mathsf{says}_a(\varphi))$	$\mathsf{says}_a(\varphi) \rightarrow \mathsf{Bel}_a(\mathsf{hears}_a(\varphi))$	Conscious

Security properties

A security property is a relation between two or more agents.

An example is the concept of *trust*.

Basic unconditional trust is a relation between two agents.

Let a and b be two agents. Then a trusts b iff a believes ever thing b says:

$$\mathrm{Trust}(a,b) \equiv \forall\varphi(\mathsf{saysTo}_b^a(\varphi) \rightarrow \mathsf{Bel}_a(\varphi))$$

Extending $\mathscr{L}_S$ with primitives for crypto

$\mathscr{L}_S \subset \mathscr{L}_S^P$.

Adding notions for protocols and primitives for encryption.

The terms of $\mathscr{L}_S^P$ contains the terms of $\mathscr{L}_S$ and additionally

(i) variables for agents $x, y, x_1, x_2, \ldots$,
(ii) agent-terms t, t_1, t_2, ... denoting either agent variables or agent names,
(iii) natural numbers N, N_1, N_2, ...,
(iv) terms for nonces $\mathsf{nonce}(N, t)$,
(v) protocol names μ, μ_1, μ_1, ...
(vi) encryption methods s (symmetric) and a (asymmetric), in addition to the indicators i (private) and u (public).
(vii) terms for keys $\mathsf{key}(\mathsf{s}, t_1, t_2)$, $\mathsf{key}(\mathsf{a}, \mathsf{i}, t_1)$, $\mathsf{key}(\mathsf{a}, \mathsf{u}, t_t)$.

Formal definition of $\mathscr{L}_S^P$

Definition 2 *Let t, x, k, o and μ denote respectively an agent-term, agent-variable, key, nonce name and protocol name. Then the language of security protocols $\mathscr{L}_S^P$ is the least language such that:*

(i) $\mathscr{L}_S \subseteq \mathscr{L}_S^P$.

(ii) $\mathsf{isKey}(k)$, $\mathsf{isNonce}(o) \in \mathscr{L}_S^P$.

(iii) $\mathsf{role}(t)$, $\mathsf{playRole}(t, x, \mu) \in \mathscr{L}_S^P$.

(iv) *If* $\varphi \in \mathscr{L}_S^P$, *then both* $E[k : \varphi]$, $D[k : \varphi] \in \mathscr{L}_S^P$.

(v) *If* φ, ξ^{T}, $\xi^{\mathrm{A}} \in \mathscr{L}_S^P$, *then* $\mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ \in \mathscr{L}_S$.

(vi) *If* $\varphi \in \mathscr{L}_S^P$, *then* $\mathsf{Enforce}_a(\varphi) \in \mathscr{L}_S^P$.

Explanation of new concepts

$\mathsf{isKey}(k)$: "k is a key"
$\mathsf{isNonce}(o)$: "o is a nonce"
$\mathsf{nonce}(N, t)$: "the number N generated by the agent t".
$\mathsf{role}(t)$: "agent t plays a role implicitly in a protocol"
$\mathsf{playRole}(t, x, \mu)$: "agent t plays the role x in the protocol named μ".
$E[k : \varphi]$: "the sentence φ encrypted with the key k"
$D[k : \varphi]$: "the sentence φ decrypted with the key k"

Symmetric crypto axiom:

$$D[\mathsf{key}(\mathsf{s}, t_1, t_2) : E[\mathsf{key}(\mathsf{s}, t_1, t_2) : \varphi]] \leftrightarrow \varphi.$$

$\mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ \in \mathscr{L}_S$: "protocol named μ with session number N, the total roles ξ^{T}, the agent specific roles ξ^{A}, and protocol body φ".

$\mathsf{Enforce}_a(\varphi)$: "agent a enforce the sentence φ" or "agent a do φ".

Free variables

Free agent-variables in an agent-term t is defined in the standard way by recursion over the complexity of φ, except:

$$\text{free}(\mathsf{playRole}(t, x, \mu)) = \varnothing \qquad \text{FV-1}$$
$$\text{free}(\mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi) = \text{free}(\xi^{\mathrm{T}}) \cup \text{free}(\varphi) \qquad \text{FV-2}$$

Hence,

1. the free variables in a playRole-statement is not collected (FV1), and
2. in a protocol only the free variables in the set of total roles specified in the conjunction ξ^{T} and the protocol body φ is counted (FV-2), not the roles the agent can play ξ^{A}.

The reason for this will be clear when we discuss how protocols are instantiated in section 31.

Specifying protocols

Protocols can be considered as distributed programs.

A protocol specifies an sequence of events between two or more agents.

The before operator $\varphi \mathcal{B} \psi$, has the semantics that if ψ occurs then φ has occurred previously. Fortunately the concept of ordering events, is definable within $\mathscr{L}_S$ using the before operator $\mathcal{B}$:

$$\varphi_1 \mathcal{B} \varphi_2 \wedge \varphi_2 \mathcal{B} \varphi_3 \wedge \ldots \wedge \varphi_{n-1} \mathcal{B} \varphi_n (\dagger).$$

Note that $(\dagger)$ expresses informally that:

φ_1 occurs before φ_2, that occurs before $\ldots$, that occurs before φ_{n-1} that occurs before φ_n.

Length of chains

Suppose that neither of the φ_i's contain any occurence of the before operator. Let (†) be called a *chain of events* of length n.

Definition 3 *The length of a chain of events φ, denoted $lh(\varphi)$ is defined by;*

(i) $lh(\top) = 0$

(ii) $lh(\varphi) = 1$, *if there are no occurence of* $\mathcal{B}$ *in* φ,

(iii) $lh(\varphi_1 \mathcal{B} \varphi_2) = 2$ *and finally*

(iv) $lh(\varphi_1 \mathcal{B} \varphi_2 \wedge \varphi') = 1 + lh(\varphi')$,
if $\varphi = \varphi_1 \mathcal{B} \varphi_2 \wedge \varphi'$ *is a chain of events.*

Hence there are two classes of peculiar chain of events,

the empty $\top$ and

the singleton chains φ.

Notation and functions

If φ is a chain of events,

head(φ) : first event of φ.
last(φ) : last event in φ.
Convenience: $\mathscr{B}$, permits the user to define a standard ordering directly.

The translation is defined by recursion on the number of conjuncts $(n-1)$ in the chain of events:

$(\varphi_1 \mathcal{B} \varphi_2 \wedge \varphi_2 \mathcal{B} \varphi_3 \wedge \ldots \wedge \varphi_{n-1} \mathcal{B} \varphi_n)^\star = \varphi_1 \mathscr{B} (\varphi_2 \mathcal{B} \varphi_3 \wedge \ldots \wedge \varphi_{n-1} \mathcal{B} \varphi_n)^\star$,
and $(\varphi_1 \mathcal{B} \varphi_2)^\star = \varphi_1 \mathscr{B} \varphi_2$.

Hence the chain of events can be expressed more naturally as
$\varphi_1 \mathscr{B} \varphi_2 \mathscr{B} \varphi_3 \ldots \varphi_{n-1} \mathscr{B} \varphi_n$ (‡). The converse function $(\varphi)_\star$, takes a sentence φ written on form (‡) and returns a chain of events of form (†).

Temporal logic

Temporal logic , can be used to give semantics to the until operator by an ordering relation between points in time,

A *temporal model* is a triple $\mathscr{M} = \langle W, <, V \rangle$, where W is a set of points in time, $<$ a binary relation between points, and V a valuation function ascribing truth values to the atomic formulas. The semantics of temporal logic is given as follows :

(i) $\mathscr{M} \models_x P$ iff $x \in V(P)$, where $P \in$ atomic

(ii) $\mathscr{M} \models_x \neg\varphi$ iff $\mathscr{M} \not\models_x \varphi$

(iii) $\mathscr{M} \models_x \varphi \rightarrow \psi$ iff $\mathscr{M} \models_x \varphi$ implies $\mathscr{M} \models_x \psi$

(iv) $\mathscr{M} \models_x \varphi \mathcal{U} \psi$ iff

$$\exists y(x < y \wedge \mathscr{M} \models_y \psi \wedge \forall z(x < z \wedge z < y \rightarrow \mathscr{M} \models_z \varphi))$$

Lemma 1 *If the ordering relation is transitive,* $\mathcal{B}$ *is transitive.*

Proof: Recapitulate that $\mathcal{B}$ is definable from $\mathcal{U}$ by $\varphi \mathcal{B} \psi = \neg(\neg\varphi \mathcal{U} \psi)$. But according to the definition of the until operator this means:

$\mathscr{M} \models_x \varphi \mathcal{B} \psi$ iff $\forall y(x < y \wedge \mathscr{M} \models_y \psi \rightarrow \exists z(x < z \wedge z < y \wedge \mathscr{M} \models_z \varphi))$

That $<$ is transitive, means simply $x < y \wedge y < z \rightarrow x < z$.

Then the lemma expresses;

$(x < y \wedge y < z \rightarrow x < z) \rightarrow (\varphi_1 \mathcal{B} \varphi_2 \wedge \varphi_2 \mathcal{B} \varphi_3 \rightarrow \varphi_1 \mathcal{B} \varphi_3)$

The argument is carried out by a proof by contradiction. □

Partial orders - Partial Ordered Sets

In computer science partial orders are everywhere:

Definition 4 *A partial order is a pair* $(R, \mathscr{H})$*, where* R *is a binary relation and* $\mathscr{H}$ *is a set, with* $R \subseteq \mathscr{H} \times \mathscr{H}$*, such that* R *is*

reflexive;	$\forall x \in \mathscr{H}\ xRx$
anti-symmetric;	$\forall x \in \mathscr{H} \forall y \in \mathscr{H}\ xRy \wedge yRx \implies x = y$
transitive;	$\forall x \in \mathscr{H} \forall y \in \mathscr{H} \forall z \in \mathscr{H}\ xRy \wedge yRz \implies xRz$

Conjunctions are ACI

$A \wedge (B \wedge C) = (A \wedge B) \wedge C$ associativity

$A \wedge B = B \wedge A$ commutativity

$A \wedge \top = A = \top \wedge A$

$A \wedge A = A$ idempotence

Let φ and ψ be conjunctions, $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_n$ and $\psi = \psi_1 \wedge \ldots \wedge \psi_m$.

Then φ is *included* in ψ, denoted $\varphi \sqsubseteq \psi$, iff
$\forall i \in \{1, \ldots, n\} \exists j \in \{1, \ldots, m\} (\varphi_i = \psi_j)$.

We observe also that $\top \sqsubseteq \psi$ for any ψ.

Lemma 2 *The relation* $\sqsubseteq$ *is a partial order.*

Proof: Left as an exercise $\square$

Exercise 1 *Prove the previous lemma.*

Formal definition of text-book protocols

Definition 5 *A* valid text-book protocol, *is a sentence in* $\mathscr{L}_S^P$, *on the form;* $\mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi$, *such that*

(i) ξ^{T} *and* ξ^{A} *are finite conjunctions of roles;*
$\xi^{\mathrm{T}} = \mathsf{role}(x_1) \wedge \ldots \wedge \mathsf{role}(x_n)$ *and* $\xi^{\mathrm{A}} = \mathsf{role}(y_1) \wedge \ldots \wedge \mathsf{role}(y_m)$,
such that $\xi^{\mathrm{A}} \sqsubseteq \xi^{\mathrm{T}}$.

(ii) φ *is a chain of events* $\varphi_1 \mathscr{B} \varphi_2 \mathscr{B} \ldots \mathscr{B} \varphi_n$,
such that for $1 \leqslant i \leqslant n$, $\varphi_i = \mathsf{Transmits}(x_j, x_k, \ulcorner \xi \urcorner)$,
where x_j *and* x_k *are agent-variables and* $\xi \in \mathscr{L}_S^P$.

(iii) $\mathrm{free}(\xi^{\mathrm{T}}) = \mathrm{free}(\varphi)$.

Exercise 2 *Explain why the previous definition is reasonable*

Needham Schroeder in $\mathscr{L}_S^P$

$$\mathsf{protocol} \quad \text{“Needham Schröder Symmetric Key 1”} \quad 0 \qquad (H_1)$$

$$\mathsf{role}(x) \wedge \mathsf{role}(y) \wedge \mathsf{role}(z) \qquad \top \qquad (H_2)$$

$$\mathsf{Transmits}(x, z, \ulcorner \mathsf{Agent}(x) \wedge \mathsf{Agent}(y) \wedge \mathsf{isNonce}(\mathsf{nonce}(N, x)) \urcorner) \qquad (1)$$

$\mathscr{B}$

$$\mathsf{Transmits}(z, x, \ulcorner E[\mathsf{key}(\mathsf{s}, x, z) \,:\, \mathsf{Agent}(y) \wedge \mathsf{isNonce}(\mathsf{nonce}(N, x)) \wedge \mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, y)) \wedge E[\mathsf{key}(\mathsf{s}, y, z) \,:\, \mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, y)) \wedge \mathsf{Agent}(x)]] \urcorner) \qquad (2)$$

$\mathscr{B}$

$$\mathsf{Transmits}(x, y, \ulcorner E[\mathsf{key}(\mathsf{s}, y, z) \,:\, \mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, z)) \wedge \mathsf{Agent}(x)] \urcorner) \qquad (3)$$

$\mathscr{B}$

$$\mathsf{Transmits}(y, x, \ulcorner E[\mathsf{key}(\mathsf{s}, x, y) \,:\, \mathsf{isNonce}(\mathsf{nonce}(N, y))] \urcorner) \qquad (4)$$

$\mathscr{B}$

$$\mathsf{Transmits}(x, y, \ulcorner E[\mathsf{key}(\mathsf{s}, x, y) \,:\, \mathsf{isNonce}(\mathsf{nonce}(N - 1, y))] \urcorner) \qquad (5)$$

Observation 1 *Needham Schröder Symmetric Key protocol as described in the previous figure is valid text-book protocol.*

Proof: Left as an exercise. □

Text-book protocols (Wenbo Mao 2004), are convenient ways of abstracting away implementation details, and merely focusing of the pure conceptual ideas of designing protocols.

It leaves the reader with the following important puzzles:

What is assumed about the environment? (Dolev Yao)
What is assumed about the participating agents?

Exercise 3 *Give a detailed proof of the observation.*

Agents playing roles

The specification figure 24 is not executable, in the simulator, not even for indifferent agents.

The reason is that

1. the receivers does not have any beliefs about the role's they should play in the protocol,
2. they do not have any beliefs about the roles of the other participants.

The first problem is then solved by specifying which roles Alice, Bob and Server can play explicit, by adjusting the local agent roles ξ^{A} for each agent.

Alice:	$\mathsf{role}(x) \wedge \mathsf{role}(y)$
Bob:	$\mathsf{role}(x) \wedge \mathsf{role}(y)$
Server:	$\mathsf{role}(z)$

Solving the second problem

We assign the roles explicitly.

$$\mathsf{Transmits}(x, z, \ulcorner \mathsf{playRole}(x, x, \mathrm{ns}_1) \wedge \mathsf{playRole}(y, y, \mathrm{ns}_1) \wedge \mathsf{isNonce}(\mathsf{nonce}(N, x)) \urcorner) \quad (1')$$

$$\mathsf{Transmits}(x, y, \ulcorner \mathsf{playRole}(x, x, \mathrm{ns}_1) \wedge \mathsf{playRole}(z, z, \mathrm{ns}_1) \wedge E[\mathsf{key}(\mathsf{s}, y, z) : \mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, z)) \wedge \mathsf{Agent}(x)] \urcorner) \quad (3')$$

A protocol instance would then contain the sentences:

$$\mathsf{Transmits}(\mathrm{Alice}, \mathrm{Server}, \ulcorner \mathsf{playRole}(\mathrm{Alice}, x, \mathrm{ns}_1) \wedge \mathsf{playRole}(\mathrm{Bob}, y, \mathrm{ns}_1) \wedge \mathsf{isNonce}(\mathsf{nonce}(N, \mathrm{Alice})) \urcorner) \quad (1'')$$

$$\mathsf{Transmits}(\mathrm{Alice}, \mathrm{Bob}, \ulcorner \mathsf{playRole}(\mathrm{Alice}, x, \mathrm{ns}_1) \wedge \mathsf{playRole}(\mathrm{Server}, z, \mathrm{ns}_1) \wedge E[\mathsf{key}(\mathsf{s}, \mathrm{Bob}, \mathrm{Server}) : \mathsf{isKey}(\mathsf{key}(\mathsf{s}, \mathrm{Alice}, \mathrm{Server})) \wedge \mathsf{Agent}(\mathrm{Alice})] \urcorner) \quad (3'')$$

A simulator for protocols

- a *meta-protocol*, prescribing valid ways to specify a huge class of protocols.
- written in Maude

Agents + messages that inhabit configurations. Agents looks like:

$$\prec \mathsf{id} \mid \mathsf{bel}, \mathsf{just}, \mathsf{hist}, \mathsf{pers} \succ,$$

- id : denotes the identity of the agent,
- bel : the current set of beliefs,
- just : the set of sentences that the agent has justified,
- hist : denotes the history of interactions with the environment,
- pers : characterize the personality of the agent,

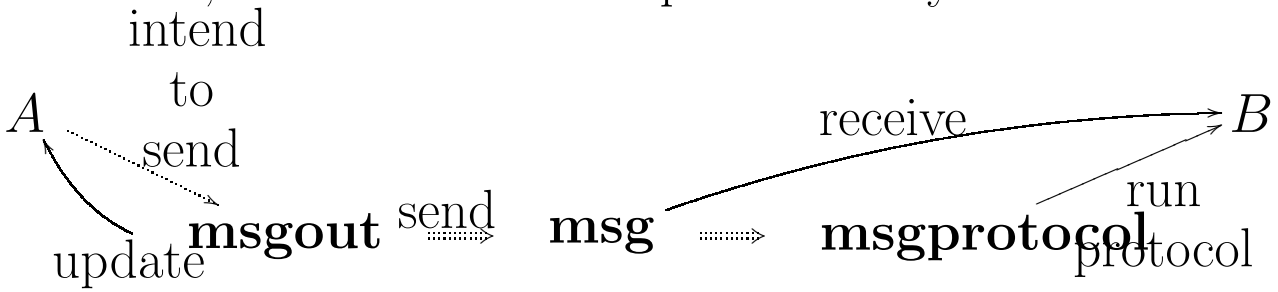
hist : $\langle m_1, \ldots, m_n \rangle$, where $m_1, \ldots, m_n$ denote messages.
$\mathsf{hist} \vdash \mathsf{Transmits}(a, b, m)$ where
$\langle m_1, \ldots, m_n \rangle \vdash m_{n+1} = \langle m_1, \ldots, m_n, m_{n+1} \rangle$.

Message flow in the simulator

The transmission of messages in the protocol simulator can be divided into four steps;

- intending to send,
- in the network,
- receiving, and
- receive as protocol sentences.

The following diagram shows how message transfer is organized in the simulator, the transmission is performed by a sender A and a reciever B:



Sending and receiving messages

Sending - Honest and conscious agent:

$$\prec b \,|\, \mathsf{bel}, \mathsf{just}, \mathsf{hist}, \mathsf{pers} \succ \mathsf{messout}\ \ulcorner F \urcorner\ \mathsf{from}\ a\ \mathsf{to}\ b \longrightarrow$$
$$\prec b \,|\, \mathsf{bel} \cup \{\mathsf{Transmits}(a, b, \ulcorner F \urcorner)\}, \mathsf{just}, \mathsf{hist} \vdash \mathsf{Transmits}(a, b, \ulcorner F \urcorner), \mathsf{pers} \succ$$
$$\mathsf{mess}\ \ulcorner F \urcorner\ \mathsf{from}\ a\ \mathsf{to}\ b \quad \text{if}\ h, c \in \mathsf{pers}\ \text{and}\ F \in \mathsf{bel}$$

Receiving - conscious:

$$\prec b \,|\, \mathsf{bel}, \mathsf{just}, \mathsf{hist}, \mathsf{pers} \succ \quad \mathsf{mess}\ \ulcorner F \urcorner\ \mathsf{from}\ a\ \mathsf{to}\ b \longrightarrow$$
$$\prec b \,|\, \mathsf{bel} \cup \{\mathsf{Transmits}(a, b, \ulcorner F \urcorner), \mathsf{Agent}(a)\}, \mathsf{just}, \mathsf{hist} \vdash \mathsf{Transmits}(a, b, \ulcorner F \urcorner), \mathsf{pers} \succ$$
$$\mathsf{messprotocol}\ \ulcorner F \urcorner\ \mathsf{from}\ a\ \mathsf{to}\ b \quad \text{if}\ c \in \mathsf{pers}$$

Instantiation of a protocol

Two ways to instantiate a protocol, either

1. by starting a new protocol-session, where the initiator plays the subject role $\mathsf{role}(x)$, or

2. by receiving, receiving the first message in a protocol.

Each agent a that should participate in an execution of a protocol $\mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi$ (PR), should posses a local copy of PR, that is;

$$\mathsf{Bel}_a(\mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi).$$

The copy of the protocol contains the roles the agent a can play, ξ^{A}.

For convenience we let the variable x serve the *initator role* in every protocol.

Agent-substitution

$$\mathrm{sub}(t, x, \varphi),$$

"the agent-term t is replaced by the agent-variable x in the formula φ"

Substitution is standard except playRole and protocol:

$$\mathrm{sub}(t, x_i, \mathsf{playRole}(t_1, x_j, \mu)) = \mathsf{playRole}(\mathrm{subst}(t, x_i, t_1), x_j, \mu) \qquad \text{sub-1}$$

$$\mathrm{sub}(t, x_i, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi) =$$
$$\mathsf{protocol}\ \mu\ N\ \mathrm{sub}(t, x_i, \xi^{\mathrm{T}})\ \xi^{\mathrm{A}}\ \mathrm{sub}(t, x_i, \varphi) \qquad \text{sub-2}$$

Instantiation

An *essential substitution*, is a substitution $\mathrm{sub}(t, x, \varphi)$, such that $\mathrm{free}(\mathrm{sub}(t, x, \varphi)) \subsetneq \mathrm{free}(\varphi)$.

An *instantiation* of a protocol PR is a finite sequence of essential substitutions $\mathrm{sub}(t_n, x_n, \ldots \mathrm{sub}(t_1, x_1, \mathsf{PR}))$.

An *executable protocol instance* is a sentence $\mathsf{PR} = \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi$, such that there are no free variables in PR, $\mathrm{free}(\mathsf{PR}) = \varnothing$.

Executable subprotocol instances

The agents participating in the execution of a protocol have different **views** depending on the role they play.

In case of the Needham Schroeder protocol, Bob does not care, and can not know what is going on in protocol sentence (1') and (2'). From Bob's point of view the protocol session starts at sentence (3').

Similarly Server only participates in the first two protocol sentences, and Server is done with its participation after transmitting protocol sentence (2).

Hence Bob and Server run what we call **executable subprotocol instances**.

Transitive embedding of chain of events

Technical core of the notion of subprotocol

Definition 6 *A chain of events φ can be* embedded transitively *into a chain of events ψ, written $\varphi \sqsubseteq_{\mathrm{E}} \psi$, if either*

(i) $\varphi = \top$, *or*

(ii) $\varphi = \psi$ *where* $lh(\varphi) = 1$, *or*

(iii) $\varphi \mathcal{B} \psi' \sqsubseteq \psi$, *or (iv)* $\psi' \mathcal{B} \varphi \sqsubseteq \psi$, *or*

(v) φ *is a chain of events of length* $n \geqslant 2$; *that is*
$\varphi = \varphi_1 \mathcal{B} \varphi_2 \wedge \ldots \wedge \varphi_{n-1} \mathcal{B} \varphi_n$, *such that*
for each $\varphi_{i-1} \mathcal{B} \varphi_i \sqsubseteq \varphi$, *with* $1 \leqslant i \leqslant n$,
there is a finite chain of events $\psi_1 \mathcal{B} \psi_2 \wedge \ldots \wedge \psi_{k-1} \mathcal{B} \psi_k \sqsubseteq \psi$,
such that $\varphi_{i-1} = \psi_1$ *and* $\varphi_i = \psi_k$.

Definition 7 $\phi \vdash \varphi_1 \mathcal{B} \varphi_2$ *means that there is a chain of events* $\varphi_1 \mathcal{B} \varphi_{i_1} \wedge \varphi_{i_1} \mathcal{B} \varphi_{i_2} \wedge \ldots \wedge \varphi_{i_{n-1}} \mathcal{B} \varphi_{i_n} \wedge \varphi_{i_n} \mathcal{B} \varphi_2$

Corollary 1 $\phi \vdash \varphi_1 \mathcal{B} \varphi_2 \implies \varphi_1 \mathcal{B} \varphi_2$

Lemma 3 *If* $\phi \sqsubseteq_{\mathrm{E}} \psi$ *and* $\phi \vdash \varphi_1 \mathcal{B} \varphi_2$ *then* $\psi \vdash \varphi_1 \mathcal{B} \varphi_2$.

Proof: Suppose that $\phi \vdash \varphi_1 \mathcal{B} \varphi_2$. Thus we can assume that

$$\varphi_1 \mathcal{B} \varphi_{i_1} \wedge \varphi_{i_1} \mathcal{B} \varphi_{i_2} \wedge \ldots \wedge \varphi_{i_{n-1}} \mathcal{B} \varphi_{i_n} \wedge \varphi_{i_n} \mathcal{B} \varphi_2 \sqsubseteq \phi.$$

Since $\phi \sqsubseteq_{\mathrm{E}} \psi$, we get

$$\varphi_1 \mathcal{B} \varphi_{i_1}, \varphi_{i_1} \mathcal{B} \varphi_{i_2}, \ldots, \varphi_{i_{n-1}} \mathcal{B} \varphi_{i_n}, \varphi_{i_n} \mathcal{B} \varphi_2 \sqsubseteq_{\mathrm{E}} \psi.$$

But according to definition 6, this means that

$$\psi \vdash \varphi_1 \mathcal{B} \varphi_{i_1}, \psi \vdash \varphi_{i_1} \mathcal{B} \varphi_{i_2}, \ldots, \psi \vdash \varphi_{i_{n-1}} \mathcal{B} \varphi_{i_n}, \psi \vdash \varphi_{i_n} \mathcal{B} \varphi_2$$

hence by transitivity of $\mathcal{B}$, $\psi \vdash \varphi_1 \mathcal{B} \varphi_2$. $\square$

Exercise 4 *Give a detailed proof of the corollary.*

Lemma 4 *Transitive embedding $\sqsubseteq_{\mathrm{E}}$ is a partial order.*

Proof: We must prove that $\sqsubseteq_{\mathrm{E}}$ is reflexive, anti-symmetric and transitive. The proof is quite hard and very long. We shall only scetch some highlights!

In case of anti-symmetry, suppose

$$\begin{array}{ll} \phi \sqsubseteq_{\mathrm{E}} \psi & (a) \text{ and} \\ \psi \sqsubseteq_{\mathrm{E}} \phi & (b) \end{array}$$

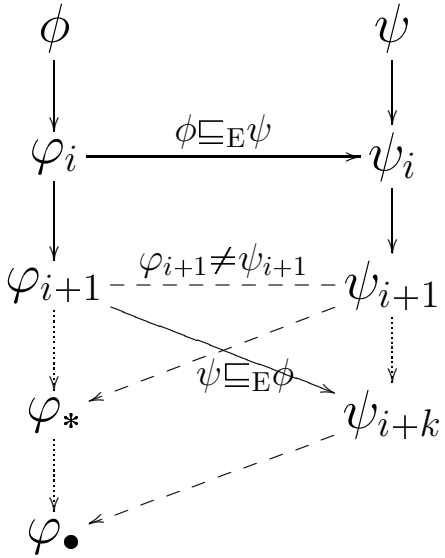
There are 2^5 cases to consider. Three of the cases gives the identity, while the remaining $2^5 - 2$ gives the absurdity of the premiss. In case $\phi = \psi = \perp$ the result follows.

In case both (a) and (b) are of form (ii), $\phi = \psi$ and $\psi = \phi$, the result follows.

Suppose both (a) and (b) are of the form (iv) and suppose futher that the events in a given chain differ.

$$\varphi = \varphi_1 \mathcal{B} \varphi_2 \wedge \ldots \wedge \varphi_{n-1} \mathcal{B} \varphi_n, \text{ and}$$
$$\psi = \psi_1 \mathcal{B} \psi_2 \wedge \ldots \wedge \psi_{k-1} \mathcal{B} \psi_k.$$

Then φ and ψ coincides exactly, with $n = k$ and $\varphi_1 = \psi_1, \ldots, \varphi_n = \psi_n$ hence $\varphi = \psi$ (†). We shall use a ping-pong argument: Suppose for contradiction that (†) is false..(a very long proof)... (Illustrated by the diagram)



$\varphi_{n+1} \neq \varphi_\bullet$ and $\varphi_{n+1} = \varphi_\bullet$ which is a contradiction

The "strange" cases (anti-symmetry)

Consider now the anti-symmetry case where

$$\varphi \mathcal{B} \varphi' \sqsubseteq \psi \text{ and } \psi \mathcal{B} \psi' \sqsubseteq \varphi,$$

hence there are two applications of part (ii) of the definition. This fact can be restated as

$$\psi = \psi_1 \wedge \varphi \mathcal{B} \varphi' \text{ and } \varphi = \varphi_1 \wedge \psi \mathcal{B} \psi'.$$

But this leads to absurdity, since we can substitute and get

$$\varphi = \varphi_1 \wedge \psi_1 \wedge (\varphi \mathcal{B} \varphi') \mathcal{B} \psi'$$

and get an infinitely long sentence. Hence it was no real case,
$\bot \implies \varphi = \psi$.

Transitivity of $\sqsubseteq_{\mathrm{E}}$

$3^5 = 243$ cases must be considered. We list the meaningful ones:

	(i)		(i)		
$\top$	$\sqsubseteq_{\mathrm{E}}$	$\top$	$\sqsubseteq_{\mathrm{E}}$	ϕ_3	$lh(\phi_3) \geqslant 1$
	(i)		$(ii)(iii)(iv)(v)$		
$\top$	$\sqsubseteq_{\mathrm{E}}$	ϕ_2	$\sqsubseteq_{\mathrm{E}}$	ϕ_3	$lh(\phi_2), lh(\phi_3) \geqslant 1$
	(ii)		$(iii)(iv)(v)$		$\phi_1 = \phi_2 = \varphi$
ϕ_1	$\sqsubseteq_{\mathrm{E}}$	ϕ_2	$\sqsubseteq_{\mathrm{E}}$	ϕ_3	$lh(\varphi) = 1$
	$(iii)(iv)$		(v)		$lh(\phi_1) = 1$
ϕ_1	$\sqsubseteq_{\mathrm{E}}$	ϕ_2	$\sqsubseteq_{\mathrm{E}}$	ϕ_3	$lh(\phi_2), lh(\phi_3) \geqslant 2$
	(v)		(v)		
ϕ_1	$\sqsubseteq_{\mathrm{E}}$	ϕ_2	$\sqsubseteq_{\mathrm{E}}$	ϕ_3	$lh(\phi_1), lh(\phi_2), lh(\phi_3) \geqslant 2$

Only 11 meaningful cases, the remaining 232 gives a false premiss. $\square$

Monoids

A structure you find everywhere in computer science is the **monoid**:

Definition 8 *A monoid is a triple* $\langle \mathscr{M}, \circ, \approx \rangle$*, where*

$\mathscr{M}$ *is a set,*

$\circ$ *is a binary operator, and*

$\approx$ *is a relation (interpreted as equality)*

(i) If $A, B \in \mathscr{M}$ then $A \circ B \in \mathscr{M}$

(ii) $\forall A, B, C \in \mathscr{M}\ A \circ (B \circ C) \approx (A \circ B) \circ C$

(iii) $\exists E \in \mathscr{M}\ \forall A \in \mathscr{M}\ E \circ A \approx A \approx A \circ E$

A monoid is **abelian** iff $A \circ B = B \circ A$

A monoid is **naked** iff $\exists! A\ \exists! A^{-1} \in \mathscr{M}$ such that $A^{-1} \circ A \approx E \approx A \circ A^{-1}$

Concatenation and its structure

Protocol names can be concatenated, if μ_1 and μ_2 are protocol names, then $\mu_1\mu_2$ denotes their concatination.

The empty name is denoted ϵ. For any set of of protocol names Θ concatenation is a naked monoid over Θ with ϵ as the unit.

Definition 9 *We say that a protocol name μ_1 is a* subname *of μ_2, written $\mu_1 \subseteq_{\mathrm{N}} \mu_2$, iff there exists names μ' and μ'', such that $\mu_2 = \mu'\mu_1\mu''$.*

Lemma 5 *If Θ is a set of protocol names, then $\langle \Theta, \subseteq_{\mathrm{N}} \rangle$ is a partial order.*

Proof: We use that the monoid is naked (that name concatination does not have any inverse except the unit). The rest is left as an exercise. □

Exercise 5 *Give a detailed proof of the lemma.*

The subprotocol relation

Definition 10 *A protocol* $\mathsf{PR}_1 = \mathsf{protocol}\ \mu_1\ N_1\ \xi_1^{\mathrm{T}}\ \xi_1^{\mathrm{A}}\ \varphi_1$ *is a* subprotocol *of a protocol* $\mathsf{PR}_2 = \mathsf{protocol}\ \mu_2\ N_2\ \xi_2^{\mathrm{T}}\ \xi_2^{\mathrm{A}}\ \varphi_2$ *, written* $\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_2$*, iff (i)* $\mu_1 \subseteq_{\mathrm{N}} \mu_2$*, (ii)* $\xi_1^{\mathrm{A}} \sqsubseteq \xi_2^{\mathrm{A}}$*, and (iii)* $(\varphi_1)_\star \sqsubseteq_{\mathrm{E}} (\varphi_2)_\star$ *where both* $(\varphi_1)_\star$ *and* $(\varphi_2)_\star$ *are chains of events.*

A protocol PR of length $lh(\mathsf{PR}) = n$ therefore has 2^n subprotocols.

Theorem 1 *The subprotocol relation* $\sqsubseteq_{\mathrm{P}}$ *is a partial order.*

Proof: Reflexivity and transitivity is straightforward.

Suppose that $\phi_1 \sqsubseteq_{\mathrm{P}} \phi_2$ and $\phi_2 \sqsubseteq_{\mathrm{P}} \phi_1$. Obviously $\mu_1 = \mu_2$ since $\mu_1 \subseteq_{\mathrm{N}} \mu_2$ and $\mu_2 \subseteq_{\mathrm{N}} \mu_1$ and we have also $\xi_1^{\mathrm{A}} = \xi_1^{\mathrm{A}}$, since $\sqsubseteq$ is anti-symmetric. By the previous lemma $(\varphi_1)_\star \sqsubseteq_{\mathrm{E}} (\varphi_2)_\star$ and $(\varphi_2)_\star \sqsubseteq_{\mathrm{E}} (\varphi_1)_\star$ implies $(\varphi_1)_\star = (\varphi_2)_\star$. Then we conclude $\phi_1 = \phi_2$. □

Exercise 6 *Fill in all the missing details in the previous theorem.*

Filters

A partially instantiated protocol might therefore be *filtered* to the relevant executable subprotocol.

Definition 11 *Let t, t_1 and t_1 be agent-terms. A filter of text-book protocols through an agent t is defined by recursion as follows:*

(i) $\text{filter}(t, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ ) = \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \text{filter}(t, \varphi)$

(ii) $\text{filter}(t, \varepsilon) = \varepsilon$

(iii) $\text{filter}(t, \mathsf{Transmits}(t_1, t_2, F)\, \mathscr{B}\, \varphi) = \text{filter}(t, \varphi)$ *if both* $t \neq t_1$ *and* $t \neq t_2$.

(iv) $\text{filter}(t, \mathsf{Transmits}(t_1, t_2, F)\, \mathscr{B}\, \varphi) = \mathsf{Transmits}(t_1, t_2, F)\, \mathscr{B}\, \text{filter}(t, \varphi)$ *if either* $t = t_1$ *or* $t = t_2$.

Theorem 2 *Every filter* PF *of a protocol* PR *is a subprotocol of* PR.

Proof: Left as exercise. □

Exercise 7 *Prove the theorem*

Simulator executing a protocol

Let us introduce two new functions, perform and replacer.

$$\prec a \mid \mathsf{bel} \cup \{\text{startProtocol}(\mu, \xi)\} \cup \{\mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ \} \succ\ =$$
$$\prec a \mid \mathsf{bel} \cup \{\text{perform}(a, \text{filter}(a, \text{replace}(\xi, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ ))\} \cup$$
$$\{\mathsf{protocol}\ \mu\ (N+1)\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ \} \succ$$
$$\text{if}\quad \{\mathsf{Agent}(t) \mid \mathsf{playRole}\ (t, x_i, \mu) \sqsubseteq \xi\} \subseteq \mathsf{bel} \qquad (*)$$
$$\text{and}\quad \{x_i \mid \mathsf{playRole}\ (t, x_i, \mu) \sqsubseteq \xi\} = \text{free}(\text{head}(\varphi)) \qquad (**)$$

startProtocol is a signal for a to start a new protocol session as the initiator.

$\xi = \mathsf{playRole}(t_1, x_1, \mu) \wedge \ldots \wedge \mathsf{playRole}(t_2, x_2, \mu)$.

(*) : ensures that the agent a is aware of the participants it intends to start the protocol with.

(**) : states that the first protocol sentence must be executable, the role-variable specified in ξ coincides exactly with the variables in the first protocol sentence.

Sending messages and awaiting for replies

$$\prec a \mid \mathsf{bel} \cup \{\mathrm{perform}(a, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \mathsf{Transmits}\ (a, b, \ulcorner F \urcorner)\ \mathscr{B}\ \varphi\ )\} \succ$$
$$\longrightarrow \prec a \mid \mathsf{bel} \cup \{\mathrm{perform}(a, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ )\} \succ$$
$$\mathsf{messout}\ \ulcorner F \urcorner\ \ \mathsf{from}\ \ a\ \ \mathsf{to}\ \ b$$

If the head of the protocol body is a passive message transmission, perform evolves into an await:

$$\prec a \mid \mathsf{bel} \cup \{\mathrm{perform}(a, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \mathsf{Transmits}\ (b, a, \ulcorner F \urcorner)\ \mathscr{B}\ \varphi\ )\} \succ \longrightarrow$$
$$\prec a \mid \mathsf{bel} \cup \{\mathrm{await}(a, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \mathsf{Transmits}\ (b, a, \ulcorner F \urcorner)\ \mathscr{B}\ \varphi\ )\} \succ$$

The state await blocks a's further behaviour until the appropriate message arrives. When the message arrives, the execution proceeds:

$$\mathsf{messprotocol}\ a\ \ \mathsf{from}\ \ b\ \ \mathsf{to}\ \ \ulcorner F \urcorner$$
$$\prec a \mid \mathsf{bel} \cup \{\mathrm{await}(a, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \mathsf{Transmits}\ (b, a, \ulcorner F \urcorner)\ \mathscr{B}\ \varphi\ )\} \succ \longrightarrow$$
$$\prec a \mid \mathsf{bel} \cup \{\mathrm{perform}(a, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ )\} \succ$$

$$\prec a \mid \mathsf{bel} \cup \{\mathrm{perform}(a, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varepsilon)\} \succ \longrightarrow \prec a \mid \mathsf{bel} \cup \{\mathrm{Done}(a, \mu, N)\} \succ$$

Initializing in the passive role.

Both Alice and Bob can play the initiator and responder roles in the authentication protocol. This is captured by the state **AwaitRespondent**,

$$\prec b \mid \mathsf{bel} \cup \{\mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ \} \succ \longrightarrow$$
$$\prec b \mid \mathsf{bel} \cup \{\mathrm{awaitRespondent}(b, \mathrm{prepProtocol}(b, \mathsf{protocol}\ \mu\ (N+1)\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ ))\}$$
$$\cup \{\mathsf{protocol}\ \mu\ (N+1)\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ \} \succ$$
$$\text{if} \quad \mathrm{awaitRespondent}(b, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ ) \notin \mathsf{bel}$$

The function prepProtocol, prepares the protocol for submission, by specifying the agent roles ξ^{A} to be one of the passive roles.

Then the first message in the filtered subprotocol should contain a set of playRoles:

$$\begin{array}{l}
\mathsf{messprotocol}\ \ulcorner F \urcorner\ \ \mathsf{from}\ \ a\ \ \mathsf{to}\ \ b \\
\prec b \mid \mathsf{bel} \cup \{\mathrm{awaitRespondent}(b, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ )\} \succ \\
\quad \longrightarrow\ \mathsf{messprotocol}\ \ulcorner F \urcorner\ \ \mathsf{from}\ \ a\ \ \mathsf{to}\ \ b \\
\quad\ \prec b \mid \mathsf{bel} \cup \{\mathrm{perform}(b, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ )\} \succ \\
\quad\quad \text{if } \mathrm{free}(\mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ ) = \varnothing
\end{array}$$

Hence, a protocol-session starts only if is executable. The substitution is carried out by the equation;

$$\begin{array}{l}
\prec b \mid \mathsf{bel} \cup \{\mathsf{Transmits}(a, b, \ulcorner F \urcorner)\} \cup \{\mathrm{awaitRespondent}(b, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ )\} \succ \\
\quad = \quad \prec b \mid \mathsf{bel} \cup \{\mathrm{awaitRespondent}(b, \\
\qquad \mathrm{replace}(\mathrm{findPlayRoles}(F), \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ ))\} \succ \\
\qquad \text{if } F \text{ contains playRole}'s.
\end{array}$$

Specifying assumptions explicitly

In the following we shall make several implicit assumptions about the agents involved in the protocol explicit, the agents'

- concrete required beliefs at a given state,
- forced beliefs from a received message, and
- internal encryption and decryption.

The first class of assumptions is represented by the belief operator.

If it is required that agent a possess a key k, the specification yields

$$\mathsf{Bel}_a(\mathsf{isKey}(k)).$$

The latter classes are specifications on the form, "agent a enforce that a believes "

$$\mathsf{Enforce}_a(\mathsf{Bel}_a(\varphi)).$$

Definition 12 *Let x and y be agent-variables and $\psi \in \mathscr{L}_S^P$. A* valid assumption protocol, *is a sentence in $\mathscr{L}_S^P$, on the form;*
$\mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi$ *, such that*

(i) ξ^{T} *and* ξ^{A} *are finite conjunctions of roles;* $\xi^{\mathrm{T}} = \mathsf{role}(x_1) \wedge \ldots \wedge \mathsf{role}(x_n)$ *and* $\xi^{\mathrm{A}} = \mathsf{role}(y_1) \wedge \ldots \wedge \mathsf{role}(y_m)$ *such that* $\xi^{\mathrm{A}} \sqsubseteq \xi^{\mathrm{T}}$.

(ii) φ *is a chain of events* $\varphi_1 \mathscr{B} \varphi_2 \mathscr{B} \ldots \mathscr{B} \varphi_n$, *such that for* $1 \leqslant i \leqslant n$, *either* $\varphi_i = \mathsf{Transmits}(x, y, \ulcorner\psi\urcorner)$, *or* $\varphi_i = \mathsf{Bel}_x(\psi)$, *or* $\varphi_i = \mathsf{Enforce}_x(\mathsf{Bel}_x(\psi))$.

(iii) $\mathrm{free}(\xi^{\mathrm{T}}) = \mathrm{free}(\varphi)$.

The usage of the double operator, shall be rather restrictive, at the moment only three kind of patterns are permitted; (i) $\mathsf{Enforce}_x(\mathsf{Bel}_x(E[k : \psi]))$; x tries to perform an encryption of formula ψ and adds this new sentence to its beliefs, (ii) $\mathsf{Enforce}_x(\mathsf{Bel}_x(D[k : \psi]))$; x tries to perform a decryption of formula ψ and adds this new sentence to its beliefs, and (iii) $\mathsf{Enforce}_x(\mathsf{Bel}_x(\mathsf{isNonce}(o)))$; x looks up a nonce in one of the messages received.

The filter function extends gently to assumption protocols by

(v) $\quad \mathrm{filter}(t, \mathsf{Bel}_{t_1}(\psi)\,\mathscr{B}\,\varphi) = \mathrm{filter}(t, \varphi)$ if $t \neq t_1$

(vi) $\quad \mathrm{filter}(t, \mathsf{Bel}_{t_1}(\psi)\,\mathscr{B}\,\varphi) = \mathsf{Bel}_{t_1}(\psi)\,\mathscr{B}\,\mathrm{filter}(t, \varphi)$ if $t = t_1$

(vii) $\quad \mathrm{filter}(t, \mathsf{Enforce}_{t_1}(\mathsf{Bel}_{t_1}(\psi)\,\mathscr{B}\,\varphi)) = \mathrm{filter}(t, \varphi)$ if $t \neq t_1$

$(viii)$ $\quad \mathrm{filter}(t, \mathsf{Enforce}_{t_1}(\mathsf{Bel}_{t_1}(\psi)\,\mathscr{B}\,\varphi)) =$

$\qquad\qquad \mathsf{Enforce}_{t_1}(\mathsf{Bel}_{t_1}(\psi))\,\mathscr{B}\,\mathrm{filter}(t, \varphi)$ if $t = t_1$

Needham Schröder with assumptions

The symbol $\star$ denote belief requirements, $\triangleright$ denote transmissions, while $\bullet$ denote enforcements, while ns_2 is an abbreviation for the protocol name.

$$\mathsf{protocol} \qquad \text{“Needham Schroeder Symmetric Key 2”} \qquad (H_1)$$

$$N \quad \mathsf{role}(x) \wedge \mathsf{role}(y) \wedge \mathsf{role}(z) \qquad \top \qquad (H_2)$$

$$\mathsf{Bel}_x(\mathsf{Agent}(y) \wedge \mathsf{Agent}(z) \wedge \mathsf{isNonce}(\mathsf{nonce}(N_1, x))) \qquad (1\star)$$

$$\mathscr{B}$$

$$\mathsf{Transmits}(x, z, \ulcorner \mathsf{playRole}(x, x, \mathrm{ns}_2) \wedge \mathsf{playRole}(y, y, \mathrm{ns}_2) \wedge \mathsf{isNonce}(\mathsf{nonce}(N_1, x)) \urcorner) \qquad (2\triangleright)$$

$$\mathscr{B}$$

$$\mathsf{Bel}_z(\mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, z)) \wedge \mathsf{isKey}(\mathsf{key}(\mathsf{s}, y, z)) \wedge \mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, y))) \qquad (3\star)$$

$$\mathscr{B}$$

The server replies

$$
\begin{aligned}
&\vdots \\
&\mathsf{Enforce}_z(\mathsf{Bel}_z(E[\mathsf{key}(\mathsf{s}, y, z) \,:\, \mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, y)) \wedge \mathsf{Agent}(x)])) && (4\bullet) \\
&\mathscr{B} \\
&\mathsf{Enforce}_z(\mathsf{Bel}_z(\mathsf{isNonce}(\mathsf{nonce}(N_1, x)))) && (5\bullet) \\
&\mathscr{B} \\
&\mathsf{Enforce}_z(\mathsf{Bel}_z(E[\mathsf{key}(\mathsf{s}, x, z) \,:\, \mathsf{isNonce}(\mathsf{nonce}(N_1, x)) \wedge \mathsf{Agent}(y) \wedge \\
&\quad \mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, y)) \wedge \\
&\qquad E[\mathsf{key}(\mathsf{s}, y, z) \,:\, \mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, y)) \wedge \mathsf{Agent}(x)]])) && (6\bullet) \\
&\mathscr{B} \\
&\mathsf{Transmits}(z, x, \ulcorner E(\mathsf{key}(\mathsf{s}, x, z), \\
&\qquad \mathsf{Agent}(y) \wedge \mathsf{isNonce}(\mathsf{nonce}(N_1, x)) \wedge \mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, y)) \wedge \\
&\qquad E[\mathsf{key}(\mathsf{s}, y, z) \,:\, \mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, y)) \wedge \mathsf{Agent}(x)])\urcorner) && (7\triangleright) \\
&\mathscr{B} \\
&\mathsf{Bel}_x(\mathsf{isKey}(\mathsf{key}(\mathsf{s}, x, z))) && (8\star) \\
&\mathscr{B}
\end{aligned}
$$

The authentication starts

$$\mathscr{B}$$

$$\mathsf{Enforce}_x(\mathsf{Bel}_x(D[\mathsf{key}(\mathsf{s},x,z) \,:\, E[\mathsf{key}(\mathsf{s},x,z) \,:\, \mathsf{isNonce}(\mathsf{nonce}(N_1,x)) \wedge \mathsf{Agent}(y) \wedge \mathsf{isKey}(\mathsf{key}(\mathsf{s},x,y)) \wedge E[\mathsf{key}(\mathsf{s},y,z) \,:\, \mathsf{isKey}(\mathsf{key}(\mathsf{s},x,y)) \wedge \mathsf{Agent}(x)]]])) \quad (9\bullet)$$

$$\mathscr{B}$$

$$\mathsf{Transmits}(x,y,\ulcorner E[\mathsf{key}(\mathsf{s},y,z) \,:\, \mathsf{isKey}(\mathsf{key}(\mathsf{s},x,y)) \wedge \mathsf{Agent}(x)]\urcorner) \quad (10\rhd)$$

$$\mathscr{B}$$

$$\mathsf{Bel}_y(\mathsf{isKey}(\mathsf{key}(\mathsf{s},y,z)) \wedge \mathsf{isNonce}(\mathsf{nonce}(N_2,y))) \quad (11\star)$$

$$\mathscr{B}$$

$$\mathsf{Enforce}_y(\mathsf{Bel}_y(D[\mathsf{key}(\mathsf{s},y,z) \,:\, E[\mathsf{key}(\mathsf{s},y,z) \,:\, \mathsf{isKey}(\mathsf{key}(\mathsf{s},x,z)) \wedge \mathsf{Agent}(x)]])) \quad (12\bullet)$$

$$\mathscr{B}$$

The reply from the authenticated subject

$$\mathscr{B}$$

$$\mathsf{Transmits}(y, x, \ulcorner \mathsf{playRole}(x, x, \mathrm{ns}_2) \wedge \mathsf{playRole}(z, z, \mathrm{ns}_2) \wedge E[\mathsf{key}(\mathsf{s}, x, y) \,:\, \mathsf{isNonce}(\mathsf{nonce}(N_2, y))]\urcorner) \quad (13\,\triangleright)$$

$$\mathscr{B}$$

$$\mathsf{Enforce}_x(\mathsf{Bel}_x(D[\mathsf{key}(\mathsf{s}, x, y) \,:\, E[\mathsf{key}(\mathsf{s}, x, y) \,:\, \mathsf{isNonce}(\mathsf{nonce}(N_2, y))]])) \quad (14\,\bullet)$$

$$\mathscr{B}$$

$$\mathsf{Enforce}_x(\mathsf{Bel}_x(\mathsf{isNonce}(\mathsf{nonce}(N_2 - 1, y)))) \quad (15\,\bullet)$$

$$\mathscr{B}$$

$$\mathsf{Enforce}_x(\mathsf{Bel}_x(E[\mathsf{key}(\mathsf{s}, x, y) \,:\, \mathsf{isNonce}(\mathsf{nonce}(N_2 - 1, y))])) \quad (16\,\bullet)$$

$$\mathscr{B}$$

$$\mathsf{Transmits}(x, y, \ulcorner E[\mathsf{key}(\mathsf{s}, x, y) \,:\, \mathsf{isNonce}(\mathsf{nonce}(N_2 - 1, y))]\urcorner) \quad (17\,\triangleright)$$

$$\mathscr{B}$$

$$\mathsf{Enforce}_y(\mathsf{Bel}_y(D[\mathsf{key}(\mathsf{s}, x, y) \,:\, E[\mathsf{key}(\mathsf{s}, x, y) \,:\, \mathsf{isNonce}(\mathsf{nonce}(N_2 - 1, y))]])) \quad (18\,\bullet)$$

Observation 2 *The protocol described in this section is a valid assumption protocol.*

Theorem 3 *The valid text-book version of Needham Schröder is a subprotocol of the valid assumption version.*

Rules for internal events

Assumptions protocols requires either a test whether a certain belief is required or a local computation on an agent. A test is performed on an agent b if the protocol requires a local state of belief for the agent b:

$$\prec b \mid \mathsf{bel} \cup \{\mathrm{perform}(b, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \mathsf{Bel}_b\,(F)\,\mathscr{B}\,\varphi\ )\} \succ \longrightarrow$$
$$\prec b \mid \mathsf{bel} \cup \{\mathrm{perform}(b, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ )\} \succ \qquad \text{if } F \in \mathsf{bel}$$

Local decryption using symmetric keys is carried out by the following equation:

$$\prec b \mid \mathsf{bel} \cup \{D[k\ :\ E[k\ :\ F]]\} \succ = \prec b \mid \mathsf{bel} \cup \{F\} \succ \qquad \text{if } \mathsf{isKey}\,(k) \in \mathsf{bel}$$

Rules for decryption and encryption

The rule for decryption therefore yields,

$$\prec b \mid \mathsf{bel} \cup \{\mathrm{perform}(b, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \ \mathsf{Enforce}_b\ (\mathsf{Bel}_b(D[k\ :\ F]))\,\mathscr{B}\,\varphi\ )\} \succ$$
$$\longrightarrow \quad \prec b \mid \mathsf{bel} \cup \{D[k\ :\ F]\} \cup \{\mathrm{perform}(b, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ )\} \succ$$

where the decryption is taken care of by the equation above.

Local encryption is carried out in the following rule, where it is presupposed that the agent posess the key:

$$\prec b \mid \mathsf{bel} \cup \{\mathrm{perform}(b, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \ \mathsf{Enforce}_b\ (\mathsf{Bel}_b(E[k\ :\ F]))\,\mathscr{B}\,\varphi\ )\} \succ$$
$$\longrightarrow \quad \prec b \mid \mathsf{bel} \cup \{E[k\ :\ F]\} \cup \{\mathrm{perform}(b, \mathsf{protocol}\ \mu\ N\ \xi^{\mathrm{T}}\ \xi^{\mathrm{A}}\ \varphi\ )\} \succ$$
$$\text{if}\ \ \mathsf{isKey}\ (k) \in \mathsf{bel}\ \text{and}\ F \in \mathsf{bel}$$

Protocol algebra

- Explain how protocols can be composed formally
- Prove a bunch of theorems about composition
- Introduce the notion of intrinsically connected subprotocol
- Show how the two subprotocols are related
- Discuss briefly how semantics can be formulated

"Gluing" chain of events

Lemma 6 *Let $\mathscr{C}$ denote the set of chain of events, then $\langle \mathscr{C}, \frown \rangle$ is a monoid.*

Proof: In order to prove that $\langle \mathscr{C}, \frown \rangle$ is a monoid we must prove that:

$$
\begin{array}{ll}
(i) & \text{If } \phi, \psi \in \mathscr{C} \text{ then } \phi \frown \psi \in \mathscr{C} \\
(ii) & \phi_1 \frown (\phi_2 \frown \phi_3) = (\phi_1 \frown \phi_2) \frown \phi_3 \\
(iii) & \exists E \in \mathscr{C} \; \forall \varphi \in \mathscr{C} \; E \frown \phi = \phi = \phi \frown E
\end{array}
$$

We first observe that a chain of events can be on three forms, $\top$ (1), a single event φ (2) and $\varphi_1 \mathcal{B} \varphi_2 \wedge \ldots \wedge \varphi_{n-1} \mathcal{B} \varphi_n$ for chain of events with length $\geqslant 1$ (3). Then definition **??** covers each of the 6 cases needed to prove (i):

In order to prove associativity (ii), there are 3^2 cases to consider: Suppose that $\phi_1 = \varphi_1 \mathcal{B} \varphi_2 \wedge \ldots \wedge \varphi_{n-1} \mathcal{B} \varphi_n$, $\phi_2 = \top$ and ϕ_3 is a single event. Then

$$\phi_1 \frown (\phi_2 \frown \phi_3) = (\varphi_1 \mathcal{B} \varphi_2 \wedge \ldots \wedge \varphi_{n-1} \mathcal{B} \varphi_n) \frown (\top \frown \phi_3) =$$
$$(\varphi_1 \mathcal{B} \varphi_2 \wedge \ldots \wedge \varphi_{n-1} \mathcal{B} \varphi_n) \frown \phi_3 = ((\varphi_1 \mathcal{B} \varphi_2 \wedge \ldots \wedge \varphi_{n-1} \mathcal{B} \varphi_n) \frown \top) \frown \phi_3 \, .$$

by two different applications of part (i). Finally $\top$ can serve as the unit, hence restating (i) gives the result. $\square$

Composition of two protocols

Definition 13 *Let* PR_1 *and* PR_2 *be two arbitrary valid protocols in* $\mathscr{L}_S$. *This means that* $\mathsf{PR}_1 = \mathsf{protocol}\ \mu_1\ N_1\ \xi_1^{\mathrm{T}}\ \xi_1^{\mathrm{A}}\ \varphi_1$ *and* $\mathsf{PR}_2 = \mathsf{protocol}\ \mu_2\ N_2\ \xi_2^{\mathrm{T}}\ \xi_2^{\mathrm{A}}\ \varphi_2$. *Then the composition of* PR_1 *with* PR_2, *denoted* $\mathsf{PR}_1 \boxplus \mathsf{PR}_2$, *is defined as follows:*

$$\mathsf{PR}_1 \boxplus \mathsf{PR}_2 = \mathsf{protocol}\ \mu_1\mu_2\ (N_1 + N_2)\ (\xi_1^{\mathrm{T}} \wedge \xi_2^{\mathrm{T}})\ (\xi_1^{\mathrm{A}} \wedge \xi_2^{\mathrm{A}})\ \varphi_1 \frown \varphi_2$$

Let $\mathscr{E} = \mathsf{protocol}\ \epsilon\ 0\ \top\ \top\ \top$ denote the *empty protocol*.

Let $\mathscr{P}$ denote the set of valid protocols.

Theorem 4 $\langle \mathscr{P}, \boxplus \rangle$ *is a monoid.*

Proof: For associativity suppose that $\mathsf{PR}_1, \mathsf{PR}_2, \mathsf{PR}_3 \in \mathscr{P}$, then

$$
\begin{aligned}
&\mathsf{PR}_1 \boxplus (\mathsf{PR}_2 \boxplus \mathsf{PR}_3) =^1 (\mathsf{protocol}\ \mu_1\ N_1\ \xi_1^{\mathrm{T}}\ \xi_1^{\mathrm{A}}\ \varphi_1\ ) \boxplus (\mathsf{PR}_1 \boxplus \mathsf{PR}_2) =^2 \\
&\mathsf{protocol}\ \mu_1(\mu_2\mu_3)\ N_1{+}(N_2{+}N_3)\ \xi_1^{\mathrm{T}} \wedge (\xi_2^{\mathrm{T}} \wedge \xi_3^{\mathrm{T}})\ \xi_1^{\mathrm{A}} \wedge (\xi_2^{\mathrm{A}} \wedge \xi_3^{\mathrm{A}})\ \varphi_1 \frown (\varphi_2 \frown \varphi_3) \quad =^3 \\
&\mathsf{protocol}\ (\mu_1\mu_2)\mu_3\ (N_1{+}N_2){+}N_3\ (\xi_1^{\mathrm{T}} \wedge \xi_2^{\mathrm{T}}) \wedge \xi_3^{\mathrm{T}}\ (\xi_1^{\mathrm{A}} \wedge \xi_2^{\mathrm{A}}) \wedge \xi_3^{\mathrm{A}}\ (\varphi_1 \frown \varphi_2) \frown \varphi_3 \quad =^4 \\
&\mathsf{protocol}\ \mu_1\mu_2\ (N_1 + N_2)\ (\xi_1^{\mathrm{T}} \wedge \xi_2^{\mathrm{T}})\ (\xi_1^{\mathrm{A}} \wedge \xi_2^{\mathrm{A}})\ \varphi_1 \frown \varphi_2\ \boxplus\ \mathsf{PR}_3 \\
&= (\mathsf{PR}_1 \boxplus \mathsf{PR}_2) \boxplus \mathsf{PR}_3
\end{aligned}
$$

Equation $=^3$ follows, since concatenation of names, $+$, $\wedge$ and $\frown$ are associative. $\square$

Exercise 8 *Prove the conditions closure and and unit.*

Theorem 5 *Let* PR_1, PR_2, PR_3 *and* PR_4 *be arbitrary protocols in* $\mathscr{P}$, *then*

(i) $\quad \mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_3 \wedge \mathsf{PR}_2 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_4 \implies \mathsf{PR}_1, \mathsf{PR}_2 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_3 \boxplus \mathsf{PR}_4$

(ii) $\quad \mathcal{E} \sqsubseteq_{\mathrm{P}} \mathsf{PR}_1$

(iii) $\quad \mathsf{PR}_1, \mathsf{PR}_2, \mathsf{PR}_3 \sqsubseteq_{\mathrm{P}} (\mathsf{PR}_1 \boxplus \mathsf{PR}_2 \boxplus \mathsf{PR}_3)$

Proof:

Part (ii) is accounted for since $\epsilon \sqsubseteq_{\mathrm{N}} \mu_1$ and $\top \sqsubseteq \xi^{\mathrm{T}}, \xi^{\mathrm{A}}, \varphi$, hence we conclude that $\mathsf{protocol}\ \epsilon\ 0\ \top\ \top\ \top\ \sqsubseteq_{\mathrm{P}} \mathsf{protocol}\ \mu_1\ N_1\ \xi_1^{\mathrm{T}}\ \xi_1^{\mathrm{A}}\ \varphi$.

Consider (iii): Since $\sqsubseteq_{\mathrm{P}}$ is reflexive $\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_1$ and since $\mathcal{E} \sqsubseteq_{\mathrm{P}} (\mathsf{PR}_2 \boxplus \mathsf{PR}_3)$, by part (ii) of this theorem, an application of part (i) gives almost what we want; $\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_1 \boxplus (\mathsf{PR}_2 \boxplus \mathsf{PR}_3)$, but finally $\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_1 \boxplus \mathsf{PR}_2 \boxplus \mathsf{PR}_3$, by associativity. The cases $\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_2 \boxplus \mathsf{PR}_2 \boxplus \mathsf{PR}_3$ and $\mathsf{PR}_3 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_1 \boxplus \mathsf{PR}_2 \boxplus \mathsf{PR}_3$ are similar and left to the reader. $\square$

Monotonicity of $\boxplus$

Lemma 7 *Protocol composition is monotone over* $\sqsubseteq_{\mathrm{P}}$.

Proof: Monotonicity of $\boxplus$ over the subprotocol relation amounts to prove:

$$\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_2 \implies \mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} (\mathsf{PR}' \boxplus \mathsf{PR}_2 \boxplus \mathsf{PR}'')$$

for arbitrary protocols PR_1, PR_2, PR' and PR''. Suppose that $\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_2$. By part (ii) of theorem 5, we have $\mathcal{E} \sqsubseteq_{\mathrm{P}} \mathsf{PR}', \mathsf{PR}''$. Then by part (i), apply the instance

$$\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_2 \wedge \mathcal{E} \sqsubseteq_{\mathrm{P}} \mathsf{PR}' \implies \mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} (\mathsf{PR}' \boxplus \mathsf{PR}_2)$$

Then reusing the instance $\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} (\mathsf{PR}' \boxplus \mathsf{PR}_2)$,

$$\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} (\mathsf{PR}' \boxplus \mathsf{PR}_2) \wedge \mathcal{E} \sqsubseteq_{\mathrm{P}} \mathsf{PR}'' \implies \mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} (\mathsf{PR}' \boxplus \mathsf{PR}_2) \boxplus \mathsf{PR}''$$

and then finally the clause for associativity gives the result. $\square$

Intrinsically connected

Part (iii) characterize an important class of subprotocols, the *intrinsically connected* subprotocols. Each of PR_1, PR_2 and PR_3 are intrinsically connected in $\mathsf{PR}_1 \boxplus \mathsf{PR}_2 \boxplus \mathsf{PR}_3$.

Definition 14 *A protocol* PR_1 *is* intrinsically connected *in a protocol* PR_2*, written* $\mathsf{PR}_1 \sqsubseteq_{\mathrm{PI}} \mathsf{PR}_2$*, iff there exists protocols* PR' *and* PR''*, such that*

$$\mathsf{PR}_2 = \mathsf{PR}' \boxplus \mathsf{PR}_1 \boxplus \mathsf{PR}''.$$

Theorem 6 *The relation* $\sqsubseteq_{\mathrm{PI}}$ *is a partial order.*

Proof: By the unit;

$$\mathsf{PR}_1 =^{(iii)} \mathcal{E} \boxplus \mathsf{PR}_1 =^{(iii)} (\mathcal{E} \boxplus \mathsf{PR}_1) \boxplus \mathcal{E} =^{(ii)} \mathcal{E} \boxplus \mathsf{PR}_1 \boxplus \mathcal{E}$$

For anti-symmetry, suppose that $\mathsf{PR}_1 \sqsubseteq_{\mathrm{PI}} \mathsf{PR}_2$ and $\mathsf{PR}_2 \sqsubseteq_{\mathrm{PI}} \mathsf{PR}_1$. This means that (a) $\mathsf{PR}_2 = \mathsf{PR}' \boxplus \mathsf{PR}_1 \boxplus \mathsf{PR}''$ and (b) $\mathsf{PR}_1 = \mathsf{PR}''' \boxplus \mathsf{PR}_2 \boxplus \mathsf{PR}''''$. Then by substitution and associativity (ii),

$$\mathsf{PR}_1 = \mathsf{PR}''' \boxplus (\mathsf{PR}' \boxplus \mathsf{PR}_1 \boxplus \mathsf{PR}'') \boxplus \mathsf{PR}'''' = (\mathsf{PR}''' \boxplus \mathsf{PR}') \boxplus \mathsf{PR}_1 \boxplus (\mathsf{PR}'' \boxplus \mathsf{PR}'''').$$

By two applications of monoid theorem part (iii) we know that $\mathsf{PR}''' \boxplus \mathsf{PR}' = \mathcal{E}$ and $\mathsf{PR}'' \boxplus \mathsf{PR}'''' = \mathcal{E}$. Then since $\langle \mathscr{P}, \boxplus \rangle$ is not a group, the only inverse is $\mathcal{E}^{-1} = \mathcal{E}$, hence $\mathsf{PR}' = \mathsf{PR}'' = \mathsf{PR}''' = \mathsf{PR}'''' = \mathcal{E}$. Then by substituting the latter fact into (a) yields $\mathsf{PR}_2 = \mathcal{E} \boxplus \mathsf{PR}_1 \boxplus \mathcal{E}$, hence by the unit again, $\mathsf{PR}_2 = \mathsf{PR}_1$, which was exactly what we wanted. □

Exercise 9 *Prove that* $\sqsubseteq_{\mathrm{PI}}$ *is transitive.*

Corollary 2 $\sqsubseteq_{\mathrm{PI}} \subseteq \sqsubseteq_{\mathrm{P}}$

Proof: We must prove that

$$\mathsf{PR}_1 \sqsubseteq_{\mathrm{PI}} \mathsf{PR}_2 \implies \mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_2.$$

Suppose therefore that $\mathsf{PR}_1 \sqsubseteq_{\mathrm{PI}} \mathsf{PR}_2$, hence $\mathsf{PR}_2 = \mathsf{PR}' \boxplus \mathsf{PR}_1 \boxplus \mathsf{PR}''$. Then finally theorem of monotonicity gives $\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}' \boxplus \mathsf{PR}_1 \boxplus \mathsf{PR}''$, which means $\mathsf{PR}_1 \sqsubseteq_{\mathrm{P}} \mathsf{PR}_2$. $\square$

Given a protocol PR with $lh(\mathsf{PR}) = n$, PR has

$$1 + \sum_{i=1}^{n} i = 1 + \frac{n \times (n+1)}{2}$$

intrinsically connected subprotocols.

Counting subprotocols

Huge difference in the number of subprotocols:

Protocol	subprotocols $\sqsubseteq_{\mathrm{P}}$	intrinsically connected $\sqsubseteq_{\mathrm{PI}}$
Text-book version:	$2^5 = 32$	$1 + \frac{5\times(5+1)}{2} = 16$
Assumption version:	$2^{18} = 262144$ (256 Kilo Byte)	$1 + \frac{18\times(18+1)}{2} = 172$

Protocol composition is not abelian. If two protocols PR_1 and PR_2 can be considered 'equal' under communativity they are said to be *order-independent*, written $\mathsf{PR}_1 \boxplus \mathsf{PR}_2 \sim \mathsf{PR}_2 \boxplus \mathsf{PR}_1$. The comparison relation $\sim$ should be considered as *semantically equivalence*. Semantical equivalence can be considered as a useful concept for explaining the behaviour of a protocol.

Protocol composition gives a huge class of nonsensical protocols, because of the relaxed closure condition of the monoid. An important subclass of reasonable composed protocols is the *agent-connected* protocols. Two protocols PR_1 and PR_2 are *agent-connected* if they share variables, that is, $\text{free}(\mathsf{PR}_1) \cap \text{free}(\mathsf{PR}_2) \neq \varnothing$.

Conclusion

The simulator is written in standard Maude, about 1500 lines of code including examples.

The full example with assumptions runs with 4489 rewrites in 10ms cpu (448900 rewrites/second).

Our approach seems fruitful to investigate both sound runs and attacks of a large class of protocols in one single framework.

Several problems remain to investigated.

A future task is to see how the security properties in Moen Haugsand (2004), and in particular the justification operator, barely mentioned in this paper, can be connected to perform analysis of the more general security mechanisms.

In the future we hope to be specify and simulate a library of authentication protocols and known attacks represented in Clark Jacob:1997.

References

[1] Abadi, M.: Security Properties and their Properties, 1999.

[2] Clark, J. and Jacob, J.: A Survey of Authentication Protocol Literature. Verson 1.0.

[3] Fagin R.,Halpern J.Y., Moses, Y., and Vardi, M. Y.: Reasoning about knowledge, The MIT Press, 1995

[4] Mao, Wenbo: Modern Cryptography: Theory and Practice, Prentice Hall, 2004

[5] Hagalisletto, A. M. and Haugsand J.: A Formal Language for Specifying Security Properties. Proceedings for the Workshop on Specification and Automated Processing of Security Requirements - SAPS'04. pp. 245 - 255