Chapter 2
Preliminaries

Abstract We describe mathematical concepts and notation used throughout the remainder of this book.

2.1 Sets and Relations

Given a set T, we write $\mathcal{P}(T)$ to denote the powerset of T, i.e., the set of all subsets of T. We write T^* to denote the set of all finite sequences of elements of T. The sequence consisting of n elements $t_0, t_1, \ldots, t_{(n-1)} \in T$ is denoted by $[t_0, t_1, \ldots, t_{(n-1)}]$, or simply by $t_0, t_1, \ldots, t_{(n-1)}$ if no confusion can occur. The empty sequence is denoted by $[\,]$. The concatenation of two sequences t and t' is denoted by $t \cdot t'$. The length of sequence $t = t_0, t_1, \ldots, t_{(n-1)}$ is n and is written as $|t|$. We write t_i to denote t's $(i+1)$th element, i.e., t_0 is the first element of t. We write $e <_t e'$ to denote $\exists i, j{:}\, i < j \wedge t_i = e \wedge t_j = e'$. We write $set(t)$ to denote the set of elements from t, i.e., $set(t) = \{t_i \mid 0 \leq i < |t|\}$. Abusing notation, we write $e \in t$ to denote $e \in set(t)$.

Pairs are written as (x, y). The two components of a pair can be extracted by the projection operator π. More precisely, for all x and y we have that $\pi_1((x, y)) = x$ and $\pi_2((x, y)) = y$.

Let f be a function. We write $dom(f)$ to denote the domain of f and $ran(f)$ to denote the range (i.e., the codomain) of f. We write $f{:}\, A \to B$ to denote a total function, which maps each element of A to elements of B. We write $f : A \nrightarrow B$ to denote a partial function, which maps some elements of A to elements of B. We say f is *injective*, notation *injective*(f), if and only if for all $x, x' \in dom(f)$ we have that $f(x) = f(x') \Rightarrow x = x'$.

A binary relation $R : T \times T$ is a subset of $T \times T$. It can satisfy the following properties (quantifying universally over $x, y, z \in T$):

(*reflexivity*)	$R(x, x)$,
(*irreflexivity*)	$\neg R(x, x)$,
(*symmetry*)	$R(x, y) \Longrightarrow R(y, x)$,
(*asymmetry*)	$R(x, y) \Longrightarrow \neg R(y, x)$,
(*transitivity*)	$R(x, y) \wedge R(y, z) \Longrightarrow R(x, z)$,
(*trichotomicity*)	either $R(x, y)$, $R(y, x)$, or $x = y$.

C. Cremers, S. Mauw, *Operational Semantics and Verification of Security Protocols*, Information Security and Cryptography, DOI 10.1007/978-3-540-78636-8_2,

Let P be one of the properties reflexivity, symmetry or transitivity. Let $R : T \times T$ be a binary relation. We define the P-closure of R as the smallest superset of R that satisfies P. We denote the transitive closure of R by R^+.

A binary relation $<: T \times T$ is a *strict partial order* if it satisfies irreflexivity, asymmetry and transitivity. A *strict total order* is a strict partial order that, in addition, satisfies trichotomicity.

Example 2.1 (Relations) Let T be the set $\{a, b, c, d\}$ and let R be the relation $\{(a, b), (b, c), (c, d)\}$. R satisfies irreflexivity and asymmetry. The reflexive closure of R is $\{(a, b), (b, c), (c, d), (a, a), (b, b), (c, c), (d, d)\}$. The transitive closure R^+ of R is $\{(a, b), (b, c), (c, d), (a, c), (a, d), (b, d)\}$. R^+ satisfies trichotomicity, irreflexivity and asymmetry; thus it is a strict total order.

2.2 BNF Grammars

Sets consisting of character strings can be defined through BNF (Backus-Naur Form) grammars [43]. A BNF grammar consists of a number of derivation rules. The left-hand side of such a derivation rule is called a *symbol*. It denotes the set that is defined through this rule. A derivation rule for the set *setname* has the form

$$setname ::= alt_1 \mid alt_2 \mid \ldots \mid alt_n.$$

The right-hand side of a derivation rule consists of a number of alternatives, which are separated by vertical bars. These alternatives represent all ways in which elements of *setname* can be generated. Every alternative itself can be a symbol, a set, a string, or a combination of these. We write $[exp]$ to denote that exp is optional and $[exp]^*$ to denote zero or more repetitions of exp.

Example 2.2 (Grammars) Let *FuncName* denote a set of function names, *FuncName* $= \{f, g, h\}$, and let *Const* denote a set of constants, *Const* $= \{c, d, e\}$. Let "(", ")", "," and "+" be strings. The following BNF grammar defines the set *Term*, which contains sums of function applications.

$$FuncApp ::= FuncName(Const\,[\,,\ Const]^*),$$
$$Term ::= FuncApp \mid Term + Term.$$

The first derivation rule defines the application of a function name to at least one constant. These constants are separated by commas. The second rule states that every function application, as defined by the first rule, is a term. Further, the recursive occurrence of the symbol *Term* on the right-hand side states that two terms separated by a + also form a term.

Some elements of *Term* are $f(c)$, $f(d, d, d, c)$, $h(c) + f(d)$, and $g(c, c) + h(d, c) + h(d, c) + f(e, e, c, e)$. The following strings are *not* elements of *Term*: $c, c + c$, $f()$, and $f(c) + (f(d) + f(e))$.

2.3 Labelled Transition Systems

A *labelled transition system* (LTS) is a four-tuple $(S, L, \rightarrow, s_0)$, where

(i) S is a set of states;
(ii) L is a set of labels;
(iii) $\rightarrow : S \times L \times S$ is a ternary transition relation;
(iv) $s_0 \in S$ is the initial state.

We abbreviate $(p, \alpha, q) \in \rightarrow$ as $p \xrightarrow{\alpha} q$. A *finite execution* of a labelled transition system $P = (S, L, \rightarrow, s_0)$ is an alternating sequence σ of states and labels, starting with s_0 and ending with a state s_n, such that if $\sigma = [s_0, \alpha_1, s_1, \alpha_2, \ldots, \alpha_n, s_n]$ then $s_i \xrightarrow{\alpha_{i+1}} s_{i+1}$ for all $0 \leq i < n$. If $[s_0, \alpha_1, s_1, \alpha_2, \ldots, \alpha_n, s_n]$ is a finite execution of LTS P, then $[\alpha_1, \alpha_2, \ldots, \alpha_n] \in L^*$ is called a *finite trace* of P. Throughout this book, the first element of a trace will have index 1.

A labelled transition system can be defined by means of a set of transition rules. A transition rule defines a number of premises $Q_1, Q_2, \ldots, Q_n (n \geq 0)$ which must all hold before a conclusion of the form $p \xrightarrow{\alpha} q$ can be drawn:

$$\frac{Q_1 \quad Q_2 \quad \ldots \quad Q_n}{p \xrightarrow{\alpha} q}.$$

Example 2.3 (LTS) We define a labelled transition system that manipulates a counter. Let $S = \mathbb{B} \times \mathbb{N}$, i.e., a state is a pair of a Boolean and a natural number. If the Boolean is *false*, an error has occurred. The initial state is $s_0 = (\mathit{true}, 0)$. The set of labels is defined as $L = \{\mathit{inc}, \mathit{dec}, \mathit{error}, \mathit{reset}\}$. The transition relation is defined by the following four transition rules:

$$\frac{b = \mathit{true}}{(b, n) \xrightarrow{\mathit{inc}} (b, n+1)}, \qquad \frac{b = \mathit{true} \quad n > 0}{(b, n) \xrightarrow{\mathit{dec}} (b, n-1)},$$

$$\frac{}{(b, n) \xrightarrow{\mathit{error}} (\mathit{false}, n)}, \qquad \frac{}{(b, n) \xrightarrow{\mathit{reset}} (\mathit{true}, 0)}.$$

The first two rules express that the counter can be incremented and decremented if there is no error. The counter cannot be decremented below 0. The third and fourth rules have no premises. The third rule expresses that at any time an error may occur, bringing the counter into an erroneous state. The counter can be brought back into its initial state through a reset.

An example execution of this LTS is the following:

$$\begin{aligned}(\mathit{true}, 0) &\xrightarrow{\mathit{inc}} (\mathit{true}, 1) \xrightarrow{\mathit{inc}} (\mathit{true}, 2) \xrightarrow{\mathit{dec}} (\mathit{true}, 1) \\ &\xrightarrow{\mathit{error}} (\mathit{false}, 1) \xrightarrow{\mathit{reset}} (\mathit{true}, 0) \xrightarrow{\mathit{inc}} (\mathit{true}, 1).\end{aligned}$$

This gives rise to the trace $[\mathit{inc}, \mathit{inc}, \mathit{dec}, \mathit{error}, \mathit{reset}, \mathit{inc}]$.

http://www.springer.com/978-3-540-78635-1

Operational Semantics and Verification of Security Protocols

Cremers, C.; Mauw, S.

2012, XIV, 174 p., Hardcover
ISBN: 978-3-540-78635-1