The Inductive Approach to Protocol Analysis

CSG 399 Lecture

Last Time

CSP approach:

- Model system as a CSP process
- A specification is a property of traces
  - Often, can be represented as a process $Spec$
  - Message secrecy
  - Correspondence assertion (see notes)
- Checking a specification: $Spec \sqsubseteq P$
  - Every trace of P is a trace of $Spec$

Advantages

- There are well-developped techniques for establishing $\sqsubseteq$ by hand
  - Mechanical proof rules
- There are tools to automatically establish $\sqsubseteq$
  - FDR: a commercial model-checker
  - Requires some conditions on $Spec$ and P to terminate
- There are even tools to automatically create CSP processes from protocols
  - Casper

Question: can we do the same without requiring CSP?

Paulson's Approach

Larry Paulson advocates a simple approach:

- A protocol in a context describes a set of traces
  - These traces are defined inductively
- A specification is again a property of traces
- Checking requires proving that all the traces satisfy the property
  - By induction on the construction of the traces
- Main point: these proofs are big, uninteresting, and better left to machines
  - Use a theorem prover to write the proofs

Inductively Defined Sets

A set S is inductively defined by a set X and (guarded) operations $(f_1, P_1), (f_2, P_2), \ldots$ if S is the smallest set satisfying

(i) $X \subseteq S$

(ii) For every guarded operation (f_i, P_i),

$$\text{if } x \in S \text{ and } P_i(x) \text{ is true, then } f_i(x) \in S$$

Smallest $\equiv S$ is contained in every other set satisfying (i)–(ii)

Example

The natural numbers are inductively defined by $\{0\}$ and the operation $+1$ (no need for guard)

I.e., $\mathbb{N}$ is the smallest set such that

(i) $0 \in \mathbb{N}$

(ii) If $x \in \mathbb{N}$, then $x + 1 \in \mathbb{N}$.

Induction Principle

Theorem: Let S be inductively defined by X and $(f_1, P_1), (f_2, P_2), \ldots$, and let Q be a property of elements of S. If

(i) $Q(x)$ is true for every $x \in X$

(ii) For every (f_i, P_i): whenever $Q(x)$ is true for $x \in S$ with $P_i(x)$, then $Q(f_i(x))$ is true

Then $Q(x)$ is true for every $x \in S$

Special case: natural numbers induction

Traces

A trace is a finite sequence of events

- Says $A\ B\ M$
- Notes $A\ M$

We concentrate on the first kind of event

Thus a trace is just a finite sequence describing who sends a message to who.

- Traces do not record whether messages are received
- Cannot distinguish message no received from message received but never acted upon

Protocols Generate Traces

Let Agents be a set of agents.
Paulson's approach assumes that:

- Agents can participate in an arbitrary number of concurrent protocol interactions
- Agents can play any role in any such interaction
- Agents have an initial state $\mathsf{initState}\ A$

We can associate a set of traces to the agents running a protocol

The set of traces of a protocol will be an inductively defined set (In fact, everything will be inductively defined)

Needham-Schroeder

Recall the Needham-Schroeder protocol:

$$A \longrightarrow B : \{A, n_A\}_{k_B}$$

$$B \longrightarrow A : \{n_A, n_B\}_{k_A}$$

$$A \longrightarrow B : \{n_B\}_{k_B}$$

We assume public keys k_A known for each agent.

Traces of Needham-Schroeder I

Define the set T inductively

The empty trace:

- $\langle\rangle$ is in T

Can start an interaction: If

- t is in T
- $A \neq B$
- $n_A \notin \mathsf{used}\ t$

Then

- $t \frown \langle \mathsf{Says}\ A\ B\ \{A, n_A\}_{k_B} \rangle$ is in T

Traces of Needham-Schroeder II

Can continue an interaction: If

- t is in T
- $A \neq B$
- $n_B \notin \mathsf{used}\ t$
- $\mathsf{Says}\ A'\ B\ \{A, n_A\}_{k_B} \in t$

Then

- $t \frown \langle \mathsf{Says}\ B\ A\ \{n_A, n_B\}_{k_A} \rangle$ is in T

Traces of Needham-Schroeder III

Can continue an interaction: If

- t is in T
- $\mathsf{Says}\ A\ B\ \{A, n_A\}_{k_B} \in t$
- $\mathsf{Says}\ B'\ A\ \{n_A, n_B\}_{k_A} \in t$

Then

- $t \frown \langle \mathsf{Says}\ A\ B\ \{n_B\}_{k_B} \rangle$ is in T

Set parts H

What about the set used t, the set of values used in a trace? We need to give an inductive definition

First consider the set parts H that returns the parts of all messages in H.

It is inductively defined by

- $H \subseteq \text{parts } H$
- If $(x, y) \in \text{parts } H$ then $x \in \text{parts } H$
- If $(x, y) \in \text{parts } H$ then $y \in \text{parts } H$
- If $\{M\}_k \in \text{parts } H$ then $M \in \text{parts } H$

$\mathbf{Set}$ used t

Straightforward definition:

$$\begin{aligned} \mathsf{used}\ \langle\rangle &= \cup_B \mathsf{parts}\ (\mathsf{initState}\ B) \\ \mathsf{used}\ t \frown \langle \mathsf{Says}\ A\ B\ M \rangle &= (\mathsf{parts}\ \{M\}) \cup (\mathsf{used}\ t) \end{aligned}$$

This does not look like an inductively defined set...

But it can be put in that form... Consider $(x, t) \in \mathsf{used}$...

Adversary

The adversary is called Spy in Paulson's paper

To account for the adversary, we only need to add one rule to the inductive definition of the traces of a protocol: If

- t is in T
- $M \in \mathsf{known}\ t$
- $B \neq \mathsf{Spy}$

Then

- $t \frown \langle \mathsf{Says\ Spy}\ B\ M \rangle$ is in T

Set known t

The set of messages known to the adversary in trace t

Definition:

$$\mathsf{known}\ t = \mathsf{synth}\ (\mathsf{analz}\ (\mathsf{spies}\ t))$$

where

- spies t: set of messages the adversary has intercepted in t
- analz H: set of messages the adversary can extract from the messages in H
- synth H: set of messages the adversary can synthesize from messages in H

Set $\mathsf{synth}\ H$

Messages the adversary can synthesize from messages in H

Inductively defined:

- $\mathsf{Agents} \subseteq \mathsf{synth}\ H$
- $H \subseteq \mathsf{synth}\ H$
- If $x \in \mathsf{synth}\ H$ and $y \in \mathsf{synth}\ H$ then $(x, y) \in \mathsf{synth}\ H$
- If $x \in \mathsf{synth}\ H$ and $k \in H$ then $\{x\}_k \in \mathsf{synth}\ H$

Set $\mathsf{analz}\ H$

Messages the adversary can extract from the messages in H

Inductively defined:

- $H \subseteq \mathsf{analz}\ H$
- If $(x, y) \in \mathsf{analz}\ H$ then $x \in \mathsf{analz}\ H$
- If $(x, y) \in \mathsf{analz}\ H$ then $y \in \mathsf{analz}\ H$
- If $\{x\}_k \in \mathsf{analz}\ H$ and $k^{-1} \in \mathsf{analz}\ H$ then $x \in \mathsf{analz}\ H$

Set $\mathsf{spies}\ t$

Messages the adversary can intercept in t

Straightforward definition:

$$
\begin{aligned}
\mathsf{spies}\ \langle\rangle &= \mathsf{initState}\ \mathsf{Spy} \\
\mathsf{spies}\ t \frown \langle \mathsf{Says}\ A\ B\ M \rangle &= \{M\} \cup (\mathsf{spies}\ t)
\end{aligned}
$$

Again, this can be made into a properly inductively defined set

So?

So now, given a protocol, a set of agents, and an adversary:

- We have an inductively defined set of traces T
- Finitary description of an infinite set of traces

How do you establish that something is true of all traces?

- By applying the induction principle corresponding to T
  - If a property is true of a trace and remains true if you add an event to the trace according to the protocol, then the property is true of all traces corresponding to the protocol