3C03 Concurrency: Modelling Concurrency in FSP

Mark Handley

What do we have to model?

- Process execution speed?
  - Arbitrary speed (abstract away time)
  - Makes model system-independent.

- Concurrency or parallelism?
  - Interleaved model of concurrency
  - a then b, or b then a, or a and b at the same time.

- Relative order of actions?
  - Arbitrary interleaving between processes, but preserve relative order in each process.

The result is an asynchronous model of execution which is independent of scheduling
FSP: Parallel Composition

If \( P \) and \( Q \) are processes then \( (P \mid Q) \) denotes the parallel execution of \( P \) and \( Q \).

- \( \mid \) is used to model parallel composition of processes.
- Names of concurrent processes are preceded by \( \mid \).

Example:

\[
\text{CONVERSE} = (\text{think} \rightarrow \text{talk} \rightarrow \text{STOP}).
\]
\[
\text{ITCH} = (\text{scratch} \rightarrow \text{STOP}).
\]
\[
\mid \mid \text{CONVERSE}_\text{ITCH} = (\text{ITCH} \mid \mid \text{CONVERSE}).
\]
Properties of Parallel Composition

The parallel composition operator has two important algebraic properties:

- **Commutativity**
  - \((P \parallel Q) = (Q \parallel P)\)
  - ordering is not important

- **Associativity**
  - \(((P \parallel Q) \parallel R) = (P \parallel (Q \parallel R)) = (P \parallel Q \parallel R)\)
  - brackets can be omitted

FSP: Process Interactions

Concurrent processes that share actions interact with each other.

- This is used to model synchronisation.

Example:

- MAKER = (make->ready->MAKER).
- USER = (ready->use->USER).
- ||MAKER_USER = (MAKER || USER).

- Product has to be ready before it can be used.
Equivalent LTS

\[ \text{MAKER} = (\text{make} \rightarrow \text{ready} \rightarrow \text{MAKER}). \]
\[ \text{USER} = (\text{ready} \rightarrow \text{use} \rightarrow \text{USER}). \]
\[ \text{||MAKER_USER} = (\text{MAKER} \ || \ \text{USER}). \]

Handshake

An action that is acknowledged by another action is referred to as **handshake**

- Widely used to structure process interactions

**Example:**

\[ \text{MAKER} = (\text{make} \rightarrow \text{ready} \rightarrow \text{used} \rightarrow \text{MAKER}). \]
\[ \text{USER} = (\text{ready} \rightarrow \text{use} \rightarrow \text{used} \rightarrow \text{USER}). \]
\[ \text{||MAKER_USER} = (\text{MAKER} \ || \ \text{USER}). \]

**LTS:**

\[ 0 \rightarrow \text{make} \rightarrow 1 \rightarrow \text{ready} \rightarrow 2 \rightarrow \text{use} \rightarrow 3 \]
FSP: Process Labelling

What happens when we want to reuse a process definition?

Example:

\[
\text{SWITCH} = (\text{on} \to \text{off} \to \text{SWITCH}). \\
\text{||TWOSWITCH} = (\text{SWITCH} \mid | \text{SWITCH}).
\]

- **Does not work!**
  - Both switches share the same actions (on, off), so both switches must always switch together.
  - TWOSWITCH is indistinguishable from SWITCH.

---

FSP: Process Labelling

The process label \( a:P \) prefixes each label in the alphabet of \( P \) with \( a \)

- Avoids name clashes in different instantiations of processes and enables reuse.

Example:

\[
\text{SWITCH} = (\text{on} \to \text{off} \to \text{SWITCH}). \\
\text{||TWOSWITCH} = (a:SWITCH \mid | b:SWITCH).
\]

- **Alphabet of ** \( ||\text{TWOSWITCH} \):
  \{ a.on, a.off, b.on, b.off \}
FSP: Process Labelling (cont’d).

The process label \( \{a_1, \ldots, a_x\} \cdot P \) replaces every label \( n \) in the alphabet of \( P \) with label \( a_1.n, \ldots, a_x.n \).

**Example:**
- RESOURCE = (acquire->release->RESOURCE).
- USER = (acquire->use->release->USER).
- RESOURCE_SHARE =
  \( (a:USER \; | \; b:USER \; | \; \{a,b\}::RESOURCE) \).

---

Equivalent LTSs

- **a:USER**
  - 0: a.acquire
  - 1: a.use
  - 2: a.release
- **{a,b}::RESOURCE**
  - 0: a.acquire
  - 1: a.use
  - 2: a.release
  - 3: b.release
  - 4: b.use

- **a:USER \; | \; b:USER \; | \; \{a,b\}::RESOURCE**
  - 0: a.acquire
  - 1: a.use
  - 2: a.release
  - 3: b.release
  - 4: b.use
FSP: Relabelling

Relabelling functions change names of action labels. The relabelling function is: /{new1/old1, ..., newn/oldn}. Used to synchronise actions with different names in composite processes.

Example:
CLIENT = (call->wait->continue->CLIENT).
SERVER = (request->serve->reply->SERVER).

||CLIENT_SERVER = (CLIENT || SERVER) /{call/request, reply/wait}.
FSP: Hiding

The hiding operator \( \{a_1 \ldots a_x\} \) removes action labels \( a_1 \ldots a_x \) from alphabet of \( P \) and hides them.

- Hidden actions are labelled \( \tau \).
- Hidden actions in different processes are not shared.
- Needed to reduce complexity of large systems for analysis.

Example:

\[
\text{USER} = (\text{acquire} \rightarrow \text{use} \rightarrow \text{release} \rightarrow \text{USER}) \setminus \{\text{use}\}.
\]

FSP: Interfaces

The interface operator \( \@\{a_1 \ldots a_x\} \) hides all actions in the alphabet of \( P \) that do not occur in the set \( a_1 \ldots a_x \).

- Complementary to hiding, and sometimes more convenient.
- Like hiding, used to reduce complexity of resulting LTS.

Example:

\[
\text{USER} = (\text{acquire} \rightarrow \text{use} \rightarrow \text{release} \rightarrow \text{USER}) \@\{\text{acquire}, \text{release}\}.
\]
Alphabet Extension

The alphabet extension $P+\{a_1, \ldots, a_x\}$ extends the alphabet of process $P$ with actions in the set $a_1 \ldots a_x$.

- Used to add elements to the alphabet that are not actions in a process.

Example:

$\text{READ} = (\text{read}[0] \rightarrow \text{read}[2] \rightarrow \text{READ}) + \{\text{read}[0..2]\}$.

has alphabet $\{\text{read.0, read.1, read.2}\}$

Summary

- Parallel Composition
- Process Interactions
- Process Labelling
- Process Relabelling
- Hiding / Interfaces

- Next session: Tutorial on FSP modelling
- Solve Exercises 3 and 4 of tutorial sheet