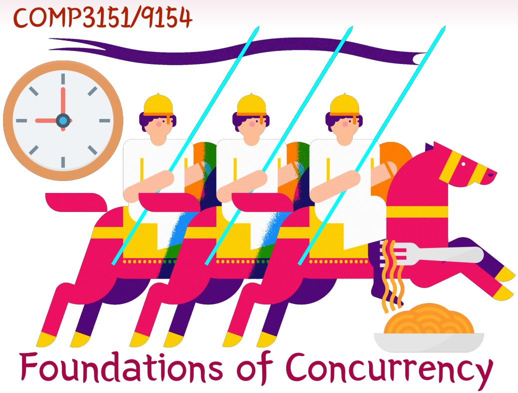


Compositionality and Asynchrony

Johannes Åman Pohjola
CSE, UNSW
Term 2 2022

Where we are at

Last lecture, we looked at proof methods for termination.

This lecture, we will conclude our examination of proof methods with *compositional* techniques, and *asynchronous systems*.

Synchronous Transition Diagrams

Definition

A *synchronous transition diagram* is a parallel composition $P_1 \parallel \ldots \parallel P_n$ of n (sequential) transition diagrams $P_1, \ldots, P_n$ called *processes*.

The processes P_i

- do not share variables
- communicate along channels $C, D, \ldots$ connecting processes by way of
  - *output* statements $C \Leftarrow e$
    for sending the value of expression e along channel C
  - *input* statements $C \Rightarrow x$
    for receiving a value along channel C into variable x

Synchronous Transition Diagrams

Definition

A *synchronous transition diagram* is a parallel composition $P_1 \parallel \ldots \parallel P_n$ of n (sequential) transition diagrams $P_1, \ldots, P_n$ called *processes*.

The processes P_i

- do not share variables
- communicate along channels $C, D, \ldots$ connecting processes by way of
    - *output* statements $C \Leftarrow e$
      for sending the value of expression e along channel C
    - *input* statements $C \Rightarrow x$
      for receiving a value along channel C into variable x

NB

Today, we will assume that all communication channels are unidirectional, and shared between at most 2 processes.

Analysis of AFR and L&G

- Both are only applicable to *closed* systems.

Analysis of AFR and L&G

- Both are only applicable to *closed* systems.
- So we have to reason about the system as a whole, even including users modelled as processes.

Analysis of AFR and L&G

- Both are only applicable to *closed* systems.
- So we have to reason about the system as a whole, even including users modelled as processes.
- In other words: we can't reason *compositionally*. Typically, non-compositional proof methods don't scale, and prevent proof re-use.

Quotes on Compositionality

de Roever et al.

A *compositional* proof method is a method by which the specification of a system can be inferred from the specifications of its constituents, without additional information about their internal structure.

Quotes on Compositionality

de Roever et al.

A *compositional* proof method is a method by which the specification of a system can be inferred from the specifications of its constituents, without additional information about their internal structure.

F. B. Schneider, 1994

Compositionality is a red herring.

One more quote

Lamport (1997) – "Composition: a way to make proofs harder"

Systems are complicated. We master their complexity by building them from simpler components. This suggests that to master the complexity of reasoning about systems, we should prove properties of the separate components and then combine those properties to deduce properties of the entire system. In concurrent systems, the obvious choice of component is the process. So, compositional reasoning has come to mean deducing properties of a system from properties of its processes.
I have long felt that this whole approach is rather silly. You don't design a mutual exclusion algorithm by first designing the individual processes and then hoping that putting them together guarantees mutual exclusion.

Compositionally-Inductive Assertion Network

Key Idea

Handle communication with a special logical variable h, containing the *history* of all communication, i.e. a sequence of pairs of channels and messages $\langle C, x \rangle$. Programs shouldn't write to h.

Compositionally-Inductive Assertion Network

Key Idea

Handle communication with a special logical variable h, containing the *history* of all communication, i.e. a sequence of pairs of channels and messages $\langle C, x \rangle$. Programs shouldn't write to h.

A local assertion network Q is *compositionally-inductive* for a sequential synchronous transition diagram $P = (L, T, s, t)$, written $P \vdash Q$, if

- $\models Q_\ell \wedge b \implies Q_{\ell'} \circ f$ for each $\ell \xrightarrow{b;f} \ell' \in T$.
- $\models Q_\ell \wedge b \implies Q_{\ell'} \circ (f \circ [\![h \leftarrow h \cdot \langle C, e \rangle]\!])$, for each $\ell \xrightarrow{b;C \Leftarrow e;f} \ell' \in T$.
- $\models Q_\ell \wedge b \implies \forall x\, (Q_{\ell'} \circ (f \circ [\![h \leftarrow h \cdot \langle C, x \rangle]\!]))$, for each $\ell \xrightarrow{b;C \Rightarrow x;f} \ell' \in T$.

Partial Correctness

Let Q be an assertion network for a process P and Q_s and Q_t be the assertions at the start and end states. We have the **Basic diagram rule**:

$$\frac{P \vdash Q}{\{Q_s\}\ P\ \{Q_t\}}$$

Partial Correctness

Let Q be an assertion network for a process P and Q_s and Q_t be the assertions at the start and end states. We have the **Basic diagram rule**:

$$\frac{P \vdash Q}{\{Q_s\}\ P\ \{Q_t\}}$$

We assume the history is empty initially with the **Initialization rule**:

$$\frac{\{\phi \land h = \varepsilon\}\ P\ \{\psi\}}{\{\phi\}\ P\ \{\psi\}}$$

Partial Correctness

Let Q be an assertion network for a process P and Q_s and Q_t be the assertions at the start and end states. We have the **Basic diagram rule**:

$$\frac{P \vdash Q}{\{Q_s\}\ P\ \{Q_t\}}$$

We assume the history is empty initially with the **Initialization rule**:

$$\frac{\{\phi \wedge h = \varepsilon\}\ P\ \{\psi\}}{\{\phi\}\ P\ \{\psi\}}$$

..the **Consequence rule** allows pre/post-conditions to be strengthened/weakened::

$$\frac{\phi \Rightarrow \phi' \quad \{\phi'\}\ P\ \{\psi'\} \quad \psi' \Rightarrow \psi}{\{\phi\}\ P\ \{\psi\}}$$

Parallel composition rule

Provided ψ_i only makes assertions about (a) local variables in P_i, and (b) the history that directly involves channels used by P_i, we get this *compositional* **Parallel composition rule**:

$$\frac{\{\phi_1\}\ P_1\ \{\psi_1\} \quad \{\phi_2\}\ P_2\ \{\psi_2\}}{\{\phi_1 \wedge \phi_2\}\ P_1 \parallel P_2\ \{\psi_1 \wedge \psi_2\}}$$

Observe that we don't need to prove anything like interference freedom or generate a proof obligation about each possible communication.

Parallel composition rule

Provided ψ_i only makes assertions about (a) local variables in P_i, and (b) the history that directly involves channels used by P_i, we get this *compositional* **Parallel composition rule**:

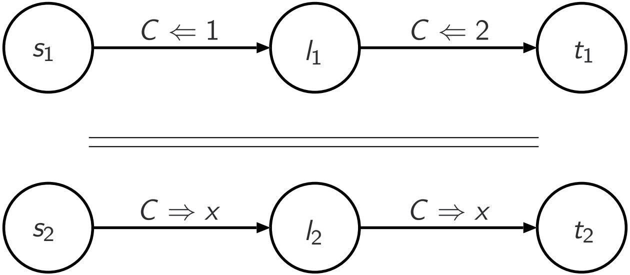
$$\frac{\{\phi_1\}\ P_1\ \{\psi_1\} \quad \{\phi_2\}\ P_2\ \{\psi_2\}}{\{\phi_1 \land \phi_2\}\ P_1 \parallel P_2\ \{\psi_1 \land \psi_2\}}$$

Observe that we don't need to prove anything like interference freedom or generate a proof obligation about each possible communication.

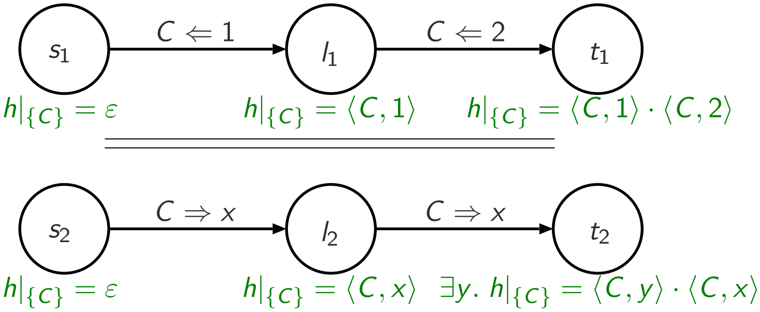
Notation

Define $h|_H$ as the history h filtered to only contain those pairs $\langle C, x\rangle$ where $C \in H$.

Example 2 once more



Example 2 once more



Example 2 once more cont'd

For the two output transitions we need to show

$$\models h|_{\{C\}} = \varepsilon \implies h|_{\{C\}} = \langle C, 1\rangle \circ [\![h \leftarrow h \cdot \langle C, 1\rangle]\!] \tag{1}$$

$$\models h|_{\{C\}} = \langle C, 1\rangle \implies h|_{\{C\}} = \langle C, 1\rangle \cdot \langle C, 2\rangle \circ [\![h \leftarrow h \cdot \langle C, 2\rangle]\!] \tag{2}$$

which is obvious; and for the two input transitions

$$\models h|_{\{C\}} = \varepsilon \implies \forall x \left(h|_{\{C\}} = \langle C, x\rangle \circ [\![h \leftarrow h \cdot \langle C, x\rangle]\!]\right) \tag{3}$$

$$\models h|_{\{C\}} = \langle C, x\rangle \implies \forall x\, \exists y\, (h|_{\{C\}} = \langle C, y\rangle \cdot \langle C, x\rangle \circ [\![h \leftarrow h \cdot \langle C, x\rangle]\!]) \tag{4}$$

which also works out nicely.

Example 2 once more cont'd

Using the **Basic diagram rule** we may now deduce

$$\{h|_{\{C\}} = \varepsilon\}\ C \Leftarrow 1; C \Leftarrow 2\ \{h|_{\{C\}} = \langle C, 1\rangle \cdot \langle C, 2\rangle\}$$

$$\{h|_{\{C\}} = \varepsilon\}\ C \Rightarrow x; C \Rightarrow x\ \{\exists y.\ h|_{\{C\}} = \langle C, y\rangle \cdot \langle C, x\rangle\}$$

Example 2 once more cont'd

Using the **Basic diagram rule** we may now deduce

$$\{h|_{\{C\}} = \varepsilon\}\ C \Leftarrow 1; C \Leftarrow 2\ \{h|_{\{C\}} = \langle C, 1\rangle \cdot \langle C, 2\rangle\}$$

$$\{h|_{\{C\}} = \varepsilon\}\ C \Rightarrow x; C \Rightarrow x\ \{\exists y.\ h|_{\{C\}} = \langle C, y\rangle \cdot \langle C, x\rangle\}$$

before applying the **parallel composition rule** to obtain

$$\{h|_{\{C\}} = \varepsilon\}\ P\ \{h|_{\{C\}} = \langle C, 1\rangle \cdot \langle C, 2\rangle \wedge \exists y.\ h|_{\{C\}} = \langle C, y\rangle \cdot \langle C, x\rangle\}$$

Example 2 once more cont'd

Using the **Basic diagram rule** we may now deduce

$$\{h|_{\{C\}} = \varepsilon\}\ C \Leftarrow 1; C \Leftarrow 2\ \{h|_{\{C\}} = \langle C, 1\rangle \cdot \langle C, 2\rangle\}$$

$$\{h|_{\{C\}} = \varepsilon\}\ C \Rightarrow x; C \Rightarrow x\ \{\exists y.\ h|_{\{C\}} = \langle C, y\rangle \cdot \langle C, x\rangle\}$$

before applying the **parallel composition rule** to obtain

$$\{h|_{\{C\}} = \varepsilon\}\ P\ \{h|_{\{C\}} = \langle C, 1\rangle \cdot \langle C, 2\rangle \wedge \exists y.\ h|_{\{C\}} = \langle C, y\rangle \cdot \langle C, x\rangle\}$$

which implies (via the rule of consequence):

$$\{h = \varepsilon\}\ P\ \{x = 2\}$$

Example 2 once more cont'd

Using the **Basic diagram rule** we may now deduce

$$\{h|_{\{C\}} = \varepsilon\}\ C \Leftarrow 1; C \Leftarrow 2\ \{h|_{\{C\}} = \langle C, 1\rangle \cdot \langle C, 2\rangle\}$$
$$\{h|_{\{C\}} = \varepsilon\}\ C \Rightarrow x; C \Rightarrow x\ \{\exists y.\ h|_{\{C\}} = \langle C, y\rangle \cdot \langle C, x\rangle\}$$

before applying the **parallel composition rule** to obtain

$$\{h|_{\{C\}} = \varepsilon\}\ P\ \{h|_{\{C\}} = \langle C, 1\rangle \cdot \langle C, 2\rangle \wedge \exists y.\ h|_{\{C\}} = \langle C, y\rangle \cdot \langle C, x\rangle\}$$

which implies (via the rule of consequence):

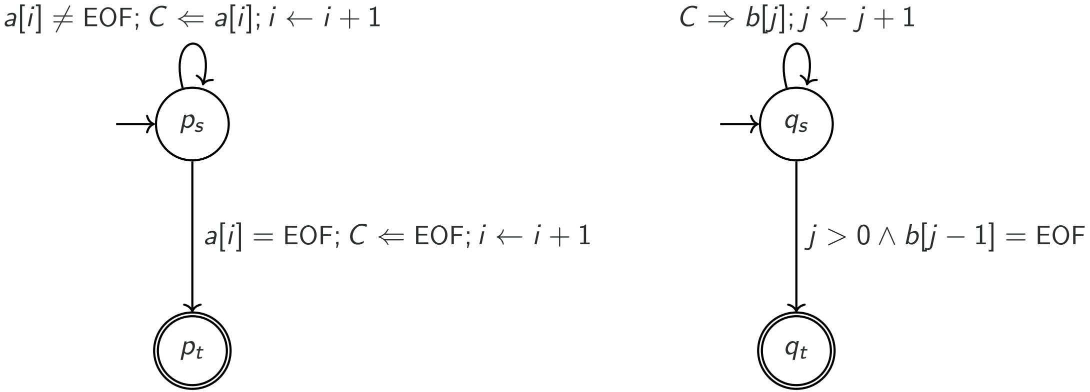
$$\{h = \varepsilon\}\ P\ \{x = 2\}$$

and finally the **initialisation rule** takes us to

$$\{\textsc{True}\}\ P\ \{x = 2\}$$

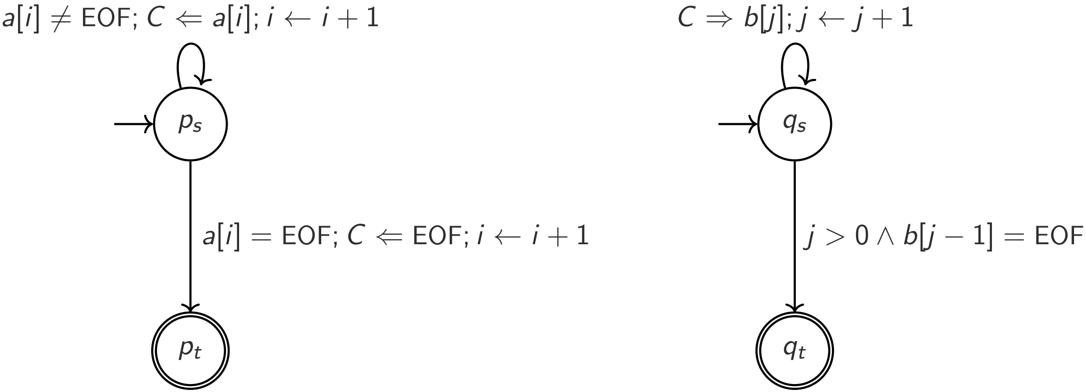
Asynchrony

Consider a process P that sends a file a on the channel C to the process Q, which saves it to b.



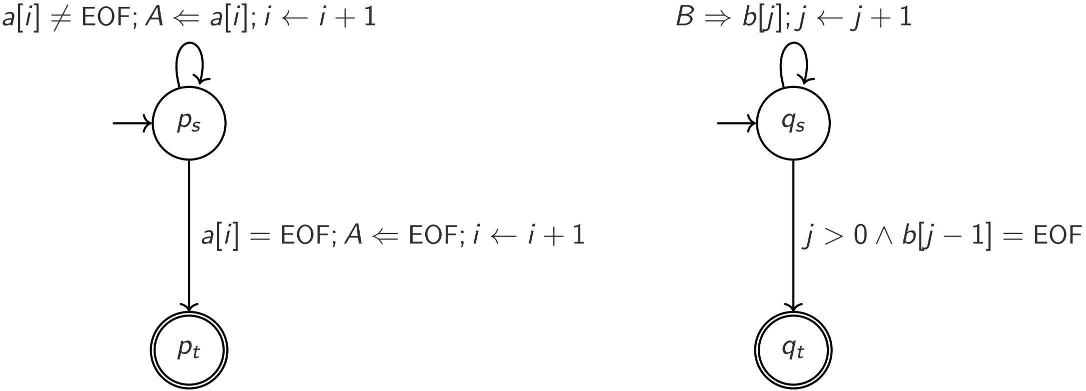
Asynchrony

Consider a process P that sends a file a on the channel C to the process Q, which saves it to b.

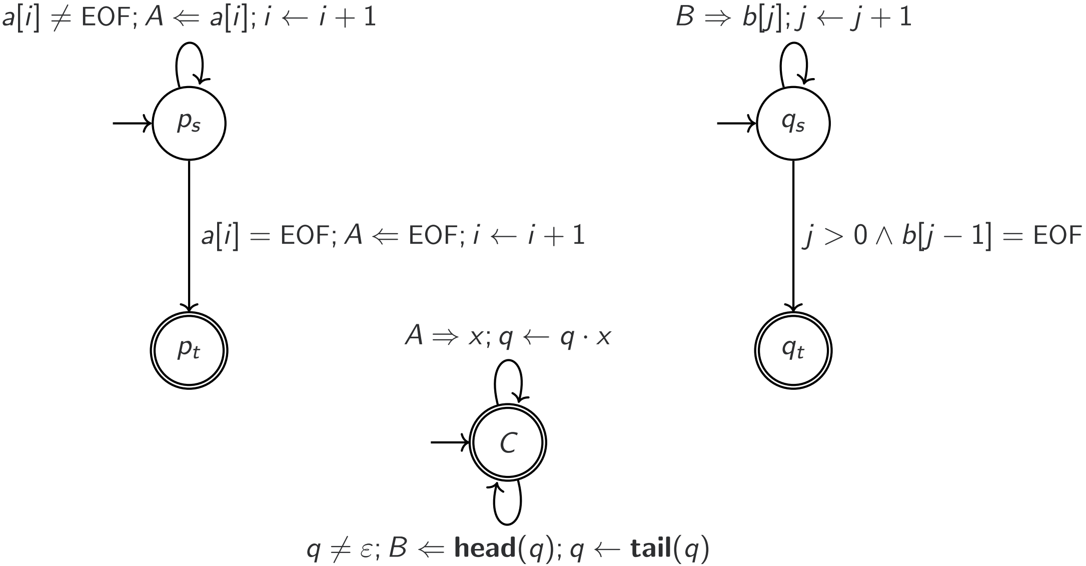


How do we verify this if C is *asynchronous*?

Convert to Synchronous



Convert to Synchronous



Compositionally

By adding an extra process with two synchronous channels to explicitly manage the queue, we convert this asynchronous system to a synchronous one.
We can now use AFR, Levin and Gries or the compositional method.

Compositionally

By adding an extra process with two synchronous channels to explicitly manage the queue, we convert this asynchronous system to a synchronous one.

We can now use AFR, Levin and Gries or the compositional method. Using the compositional method, we have the desired postcondition:

$$\exists i.\ a[i] = \mathsf{EOF} \wedge a[0 \ldots i] = b[0 \ldots i]$$

Compositionally

By adding an extra process with two synchronous channels to explicitly manage the queue, we convert this asynchronous system to a synchronous one.
We can now use AFR, Levin and Gries or the compositional method. Using the compositional method, we have the desired postcondition:

$$\exists i.\ a[i] = \mathsf{EOF} \wedge a[0 \ldots i] = b[0 \ldots i]$$

And the following assertion network:

$$\begin{aligned}
\mathcal{Q}(p_s) &\equiv \hat{h}|_{\{A\}} = a[0 \ldots i] \wedge \mathsf{EOF} \notin a[0 \ldots i] \\
\mathcal{Q}(p_t) &\equiv \hat{h}|_{\{A\}} = a[0 \ldots i] \wedge \mathsf{EOF} \notin a[0 \ldots i-1] \wedge a[i-1] = \mathsf{EOF} \\
\mathcal{Q}(q_s) &\equiv \hat{h}|_{\{B\}} = b[0 \ldots j] \\
\mathcal{Q}(q_t) &\equiv \hat{h}|_{\{B\}} = b[0 \ldots j] \wedge b[j-1] = \mathsf{EOF} \\
\mathcal{Q}(C) &\equiv \hat{h}|_{\{A\}} = \hat{h}|_{\{B\}} \cdot q
\end{aligned}$$

Proof obligations will be informally described.

What Now?

If time allows, we'll take a brief detour into the world of *process algebra*, a high level formalism for describing concurrent systems.

Either way, we'll then discuss distributed algorithms.