Specifying and Analysing Networks of Processes in CSP$_T$ (or In Search of Associativity)

Paul Howells
University of Westminster

Mark d'Inverno
Goldsmiths, University of London

Communicating Process Architectures (CPA 2013)

Outline of Talk

- Aims of Paper
- CSP$_T$'s Parallel Operators
- Roscoe's Parallel Associativity Laws
- Parallel Associativity in CSP$_T$
- Alphabet Diagrams & Event Types for 3 Processes
- "Problem" Event Types & Associativity Constraints
- Associativity Laws
- Using Associativity Law
- Conclusions & Further Work

Aims of Paper

Goal: associativity laws for CSP$_T$'s parallel operators.

- Introduce *alphabet diagrams*: provides very simple static analysis of parallel composition wrt events types.
- Analyse parallel composition of three processes using alphabet diagrams.
- Identify *associativity constraints*.
- Prove associativity laws for CSP$_T$'s parallel operators.
- Illustrate ways to use associativity laws.
- Outline how to extend to more general processes networks.

Introduction to CSP$_T$

Aim: provide a more robust treatment of termination through the consistent and special handling of ✓ by the language (processes and operators) and semantics (failures and divergences).

- Based on Brookes and Roscoe's *improved failure-divergence model* for CSP.

- CSP$_T$ defined by adding a new process axiom that captured our view of termination to original process axioms.

- View of tick (✓) is consistent with Hoare's, i.e. that it is a normal event, and not a *signal* event.

- Three new forms of generalised parallel operators were defined, each with a different form of termination semantics:
  - Synchronous termination: $P \|_{\triangle} Q$
  - Asynchronous termination: $P |||_{\Theta} Q$
  - Race termination: $P |_{\Theta} Q$

- Replaced the original interleaving ($|||$), synchronous ($\|$) & alphabetised (${}_A\|_B$) parallel operators with the synchronous ($\|_{\triangle}$), asynchronous ($|||_{\Theta}$) & race ($|_{\Theta}$) operators.

CSP$_T$'s 3 (+1) Parallel Operators

Operators are *generalised* (or *interface*) style, parameterised by synchronisation sets Δ & Θ.

Synchronous ($\|_\Delta$): requires the successful termination of both *P* & *Q*, synchronised termination on ✓ (✓ $\in \Delta$).

Asynchronous ($|||_\Theta$): requires the successful termination of both *P* & *Q*, terminate asynchronously & do not synchronise on ✓ (✓ $\notin \Delta$).

Race ($|_\Theta$): requires the successful termination of either *P* or *Q*, terminate asynchronously & do not synchronise on ✓ (✓ $\notin \Delta$).

Fails to termination only if both *P* & *Q* fail to terminate.

Whichever of *P* or *Q* terminates first, terminates $P|_\Theta Q$, the other process is aborted.

"+1" parallel operator is $\|_\Delta$, but without the constraint that ✓ must be in the synchronisation set.

Distinguish it by using $\|_\Omega$ ($\varnothing \subseteq \Omega \subseteq \Sigma$).

Can use $\|_\Omega$ to define $\|_\Delta$ & $|_\Theta$, but not $|||_\Theta$ due to its asynchronous termination semantics.

$\|_\Omega$ is not part of the CSP$_T$ language, since would re-introduce problems with ✓.

Roscoe's Parallel Associativity Laws

Roscoe states $\|_X$ is most important parallel operator.

Roscoe's "weak (in that both interfaces are the same)" associativity law:

$$P\|_X(Q\|_X R) \;=\; (P\|_X Q)\|_X R \qquad \langle \|_X\text{−assoc}\rangle$$

He states it's difficult to "...construct a universally applicable and elegant associativity law.", due to types of events that can occur.

His example: $P\|_X(Q\|_Y R)$ and an event that could occur in X but not in Y that both Q and R can perform.

Roscoe's associativity law for ${}_A\|_B$ & law relating it to $\|_X$:

$$(P\,{}_A\|_B\,Q)\,{}_{A\cup B}\|_C\,R \;=\; P\,{}_A\|_{B\cup C}\,(Q\,{}_B\|_C\,R) \qquad \langle {}_A\|_B\text{−assoc}\rangle$$
$$(P\,{}_A\|_B\,Q) \;=\; P\|_{A\cap B}\,Q$$

Results in a non-universal but more useful law for $\|_X$ than $\langle \|_X\text{−assoc}\rangle$.

But does not deal with events in $A \cap B$ that are required to be asynchronous, due to definition of ${}_A\|_B$.

Parallel Associativity in CSP$_T$

Analyse generalised operator $P \|_{\Omega} Q$, due to its role in defining the other operators.

Question: for what values of Λ_1, Λ_2, Π_1, Π_2, Γ_1 and Γ_2 does the following hold?

$$P \|_{\Lambda_1} (Q \|_{\Lambda_2} R) \;\equiv\; Q \|_{\Pi_1} (P \|_{\Pi_2} R) \;\equiv\; (P \|_{\Gamma_1} Q) \|_{\Gamma_2} R$$

Referred to as the (Λ), (Π) and (Γ) processes.

Obviously require constraints on the two synchronisation sets, since none of the following hold in general:

$$\begin{aligned} P \| (Q ||| R) &\equiv (P \| Q) ||| R \\ P ||| (Q \| R) &\equiv (P ||| Q) \| R \\ P ||| (Q\,{}_B\|_C\, R) &\equiv (P ||| Q)\,{}_{A\cup B}\|_C\, R \\ P ||| (Q\,{}_B\|_C\, R) &\equiv (P \| Q) \| R \end{aligned}$$

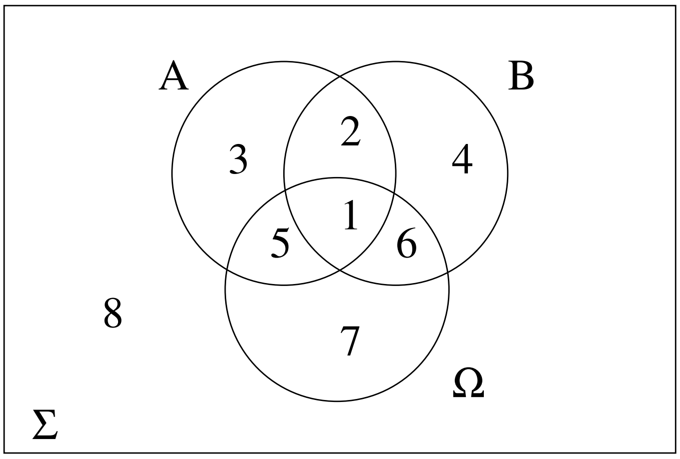
Goal: Identify constraints on synchronisation sets.

Solution: using *alphbet diagrams* to analyse types of events that can occur when *P*, *Q* & *R* are combined in parallel, i.e. (Λ), (Π) & (Γ) processes.

Alphabet Diagrams

Static analysis of parallel composition wrt types of events that could occur during its execution.

Consider the alphabet diagram for $P \|_{\Omega} Q$:



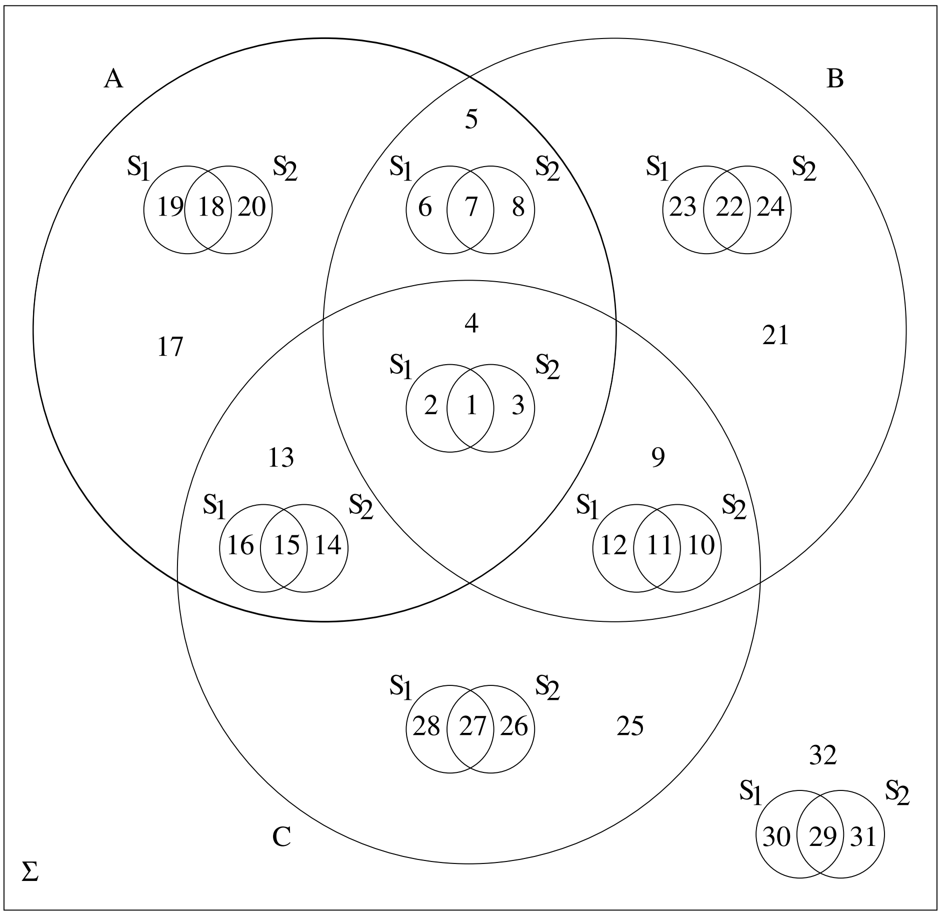
1. *Possible synchronous* events $(A \cap B \cap \Omega)$: occur when P & Q synchronise on them.
2. *Common asynchronous* events $(A \cap B \cap \overline{\Omega})$: P & Q do not synchronise on these, performed by either P or Q.
3. *P's private asynchronous* events $(A \cap \overline{B} \cap \overline{\Omega})$: performed by P.
4. *Q's private* asynchronous events $(\overline{A} \cap B \cap \overline{\Omega})$: as for *P's*.
5. *P's inhibited synchronous* events $(A \cap \overline{B} \cap \Omega)$: only possible for P but must be synchronised with Q, hence, cannot occur.
6. *Q's inhibited synchronous* events $(\overline{A} \cap B \cap \Omega)$: as for *P's*.
7. *Irrelevant synchronous* events $(\overline{A} \cap \overline{B} \cap \Omega)$ & 8. *Irrelevant* events $(\overline{A} \cap \overline{B} \cap \overline{\Omega})$: do not occur.

Alphabet Diagram for 3 Processes

Only certain combinations of events can occur in each of the (Λ), (Π) & (Γ) processes.

The following (logical) alphabet diagram represents each of the three processes one at a time.

S_1 & S_2 represent Λ_1, Λ_2, Π_1, Π_2, Γ_1 & Γ_2 respectively.



There are 32 different types, 28 are relevant.

Includes new (mixed) types of events & natural extension of the types already introduced.

Event Types for 3 Processes

Private asynchronous events: single process asynchronous – *Pa*, *Qa*, *Ra*.

Possible binary synchronous events: pairwise synchronous – *PQs*, *PRs*, *QRs*.

Common binary asynchronous events: pairwise asynchronous – *PQa*, *PRa*, *QRa*.

Possible ternary synchronous events: three way synchronous events – *PQRs*.

Common ternary asynchronous events: three way asynchronous events – *PQRa*.

Common synchronous events: are possible synchronous events because of the first synchronisation set but become common asynchronous events with the third process – *(PQs)Ra*, *(PRs)Qa*, *(QRs)Pa*.

E.g. in $P \|_{\Lambda_1} (Q \|_{\Lambda_2} R)$ only *(QRs)Pa* events can occur.

Synchronous common events: are common asynchronous events under the first synchronisation set but then become possible synchronous events when combined with the third process – *(PQa)Rs*, *(PRa)Qs*, *(QRa)Ps*.

E.g. in $Q \|_{\Pi_1} (P \|_{\Pi_2} R)$ only *(PRa)Qs* events can occur.

Various Inhibited & Irrelevant events: see paper.

“Problem” Event Types

Associativity requires the three alternatives to be equivalent:

- must have the same event types present, &
- event types must contain the same set of events.

From event type analysis clear need constraints on:

- *Private asynchronous* events: *Pa*, *Qa* & *Ra*
  - As a subset of each of these only occur in one of the three processes, depending on the scope of the two sunchronisation sets, must be constrained.
  - E.g. *Pa* contains events which are present in $P\|_{\Lambda_1}(Q\|_{\Lambda_2}R)$ that are not of the same type in the other two processes, i.e. areas 8, 14 & 20.

- *Synchronous common* events: *(PQa)Rs*, *(PRa)Qs* & *(QRa)Ps*
  - Each only occurs in one of the three alternatives, so must be eliminated.
  - E.g. *(QRa)Ps* in $P\|_{\Lambda_1}(Q\|_{\Lambda_2}R)$. (Roscoe’s example.)

- *Common synchronous* events: *(PQs)Ra*, *(PRs)Qa* & *(QRs)Pa*

  Similar reasons as above.

Associativity Constraints

The "problem" types must either be constrained or eliminated to guarantee associativity.

- For *Pa*, *Qa* & *Ra* the constraints are:

$$A \cap \overline{\Lambda_1} \cap \Lambda_2 = \varnothing$$
$$B \cap \overline{\Pi_1} \cap \Pi_2 = \varnothing$$
$$C \cap \Gamma_1 \cap \overline{\Gamma_2} = \varnothing$$

- For *(PQs)Ra*, *(PRs)Qa* & *(QRs)Pa* the constraints used for *Pa*, *Qa* & *Ra* also eliminate these events.

- For *(PQa)Rs*, *(PRa)Qs* & *(QRa)Ps* the constraints are:

$$A \cap C \cap \Pi_1 \cap \overline{\Pi_2} = \varnothing$$
$$A \cap B \cap \overline{\Gamma_1} \cap \Gamma_2 = \varnothing$$
$$B \cap C \cap \Lambda_1 \cap \overline{\Lambda_2} = \varnothing$$

Constraints for *(QRs)Pa*, *(QRa)Ps*, etc. are eliminating events that are *possible for all three processes but only within the scope of one synchronisation set*.

If Γ_1, Γ_2, Λ_1, Λ_2, Π_1 & Π_2 satisfy these constraints then:

- the problem events are eliminated.
- reduces all of the equalities on the event types which can occur to equalities of just one area in all three processes.

Associativity Laws

Using constraints arrive at associativity law for $\|_\Omega$:

$$P \|_{W \cup X \cup Y} (Q \|_{W \cup Z} R) \;\equiv\; Q \|_{W \cup X \cup Z} (P \|_{W \cup Y} R) \;\equiv\; R \|_{W \cup Y \cup Z} (P \|_{W \cup X} Q)$$

where $W \subseteq \Sigma$, $A \cap Z = \varnothing$, $B \cap Y = \varnothing$, $C \cap X = \varnothing$ and A, B, C are the alphabets of P, Q and R respectively.

W – P, Q & R synchronous events,
X – P & Q synchronous events,
Y – P & R synchronous events,
Z – Q & R synchronous events.

Based on this law have similar ones for CSP$_T$'s parallel operators:

$$\begin{array}{rcccc}
P \|_{W \cup X \cup Y} (Q \|_{W \cup Z} R) & \equiv & Q \|_{W \cup X \cup Z} (P \|_{W \cup Y} R) & \equiv & R \|_{W \cup Y \cup Z} (P \|_{W \cup X} Q) \\
P \|\!|_{W \cup X \cup Y} (Q \|\!|_{W \cup Z} R) & \equiv & Q \|\!|_{W \cup X \cup Z} (P \|\!|_{W \cup Y} R) & \equiv & R \|\!|_{W \cup Y \cup Z} (P \|\!|_{W \cup X} Q) \\
P |_{W \cup X \cup Y} (Q |_{W \cup Z} R) & \equiv & Q |_{W \cup X \cup Z} (P |_{W \cup Y} R) & \equiv & R |_{W \cup Y \cup Z} (P |_{W \cup X} Q)
\end{array}$$

W, X, Y & Z as for $\|_\Omega$ law.

Termination semantics add additional constraints:

- for $\|_\Delta$ – $\checkmark \in W$
- for $\|\!|_\Theta$ & $|_\Theta$ – $\checkmark \notin W, X, Y, Z$

Using Associativity Law

Question: When can you transformation

$$P\|_{\Lambda_1}(Q\|_{\Lambda_2}R) \;\rightarrow\; (P\|_{\Gamma_1}Q)\|_{\Gamma_2}R$$

Answer: when Λ_1 & Λ_2 satisfy the *associativity constraints*.

$$\begin{array}{lr} (1) & A \cap \overline{\Lambda_1} \cap \Lambda_2 \;=\; \varnothing \\ (2) & B \cap C \cap \Lambda_1 \cap \overline{\Lambda_2} \;=\; \varnothing \end{array}$$

If Λ_1 and Λ_2 satisfy these conditions then the process can be re-written as either of the other two forms, by using Λ_1 and Λ_2 to define W, X, Y & Z:

$$W = \Lambda_1 \cap \Lambda_2 \quad X = \overline{C} \cap \Lambda_1 \cap \overline{\Lambda_2} \quad Y = \overline{B} \cap \Lambda_1 \cap \overline{\Lambda_2} \quad Z = \overline{\Lambda_1} \cap \Lambda_2$$

Then use these to define the synchronisation sets for either of the other two processes as specified in the associativity law.

E.g. assuming Λ_1 & Λ_2 satisfy conditions:

$$P\|_{\Lambda_1}(Q\|_{\Lambda_2}R) \;\equiv\; (P\|_{\Gamma_1}Q)\|_{\Gamma_2}R$$

where

$$\begin{aligned} \Gamma_1 &= W \cup X \\ &= (\Lambda_1 \cap \Lambda_2) \cup (\overline{C} \cap \Lambda_1 \cap \overline{\Lambda_2}) \\ &= (\Lambda_1 \cap \Lambda_2) \cup (\Lambda_1 \cap \overline{C}) \end{aligned}$$

$$\begin{aligned} \Gamma_2 &= W \cup Y \cup Z \\ &= (\Lambda_1 \cap \Lambda_2) \cup (\overline{B} \cap \Lambda_1 \cap \overline{\Lambda_2}) \cup (\overline{\Lambda_1} \cap \Lambda_2) \\ &= \Lambda_2 \cup (\Lambda_1 \cap \overline{B}) \end{aligned}$$

Conclusions

- *Associativity constraints* used to prove “strongish” associativity laws for CSP$_T$’s parallel operators.
- Laws not “universally” in Roscoe’s sense, but stronger than existing laws for these style of operators.
- Demonstrated how to apply associativity laws using constraints.
- Provided designers with essential laws & techniques for designing & analysing simple process networks.

Further Work

- Extend to deal with an arbitrary number (n) of processes:

  $$P_1 \parallel_{\Omega_1} (P_2 \parallel_{\Omega_2} (\ldots (P_{n-1} \parallel_{\Omega_{n-1}} P_n) \ldots)$$

  n alphabets, $n-1$ synchronisation sets & 2^{2n-1} event types.

- Simpler for associative networks:
  - use $X_{i,j}$ for synchronous events between P_i and P_j.
  - $X_{i,j}$ is disjoint with all other processes' alphabets:

  $$X_{i,j} \cap (\bigcup_{k \neq i,j} A_k) \;=\; \varnothing$$

- One Reviewer asked for indication of "order of magnitude" of the different types of events present.
  - only *pure* synchronous and asynchronous events.
  - (pure) synchronous event types is $2^n - (n+1)$
  - (pure) asynchronous event types it is $2^n - 1$

- Constraints on the two synchronisation sets for associativity law to hold are *sufficient*.

  Two Reviewers asked are they *necessary*? – probably not.

- Apply associativity constraints within the CSP community, to produce more useful associativity laws.

Appendix A: Operational Semantics for $\|_\Delta$, $|||_\Theta$ & $|_\Theta$

Use Roscoe's *LTS* style of operational semantics.

- Ω represents a terminated process, no transitions.
- τ represents hidden events, e.g. hidden $\checkmark$s.

Firing rules for non-$\checkmark$ events same for all three:

$$\frac{P \xrightarrow{a} P', \quad Q \xrightarrow{a} Q'}{P|_\Theta Q \xrightarrow{a} P'|_\Theta Q'} \qquad [a \in \Theta]$$

$$\frac{P \xrightarrow{a} P'}{P|_\Theta Q \xrightarrow{a} P'|_\Theta Q} \qquad \frac{Q \xrightarrow{a} Q'}{P|_\Theta Q \xrightarrow{a} P|_\Theta Q'} \qquad [a \notin \Theta]$$

$$\frac{P \xrightarrow{\tau} P'}{P|_\Theta Q \xrightarrow{\tau} P'|_\Theta Q} \qquad \frac{Q \xrightarrow{\tau} Q'}{P|_\Theta Q \xrightarrow{\tau} P|_\Theta Q'}$$

Different Termination (✓) Firing Rules

$P\|_\Delta Q$ terminates only when P and Q terminate synchronously.

$$\frac{P \xrightarrow{\checkmark} P' \quad Q \xrightarrow{\checkmark} Q'}{P\|_\Delta Q \xrightarrow{\checkmark} \Omega}$$

$P|||_\Theta Q$ terminates only after both P and Q have terminated asynchronously.

$$\frac{P \xrightarrow{\checkmark} P'}{P|||_\Theta Q \xrightarrow{\tau} \Omega|||_\Theta Q} \qquad \frac{Q \xrightarrow{\checkmark} Q'}{P|||_\Theta Q \xrightarrow{\tau} P|||_\Theta \Omega}$$

Successful termination of the first process to terminate is a hidden event represent by τ.

Rule for termination of remaining process & terminates the parallel composition, transforming it into Ω:

$$\frac{P \xrightarrow{\checkmark} P'}{P|||_\Theta \Omega \xrightarrow{\checkmark} \Omega} \qquad \frac{Q \xrightarrow{\checkmark} Q'}{\Omega|||_\Theta Q \xrightarrow{\checkmark} \Omega}$$

$P|_\Theta Q$ terminates if either P or Q terminates.

$$\frac{P \xrightarrow{\checkmark} P'}{P|_\Theta Q \xrightarrow{\checkmark} \Omega} \qquad \frac{Q \xrightarrow{\checkmark} Q'}{P|_\Theta Q \xrightarrow{\checkmark} \Omega}$$

Appendix B: Example Processes using $\|_\Delta$, $|||_\Theta$ & $|_\Theta$

1. Using $\Theta = \varnothing$ & $\Delta = \{\checkmark\}$

$$\begin{aligned}(a \to \mathit{SKIP} \|_\Delta \mathit{SKIP}) &\equiv a \to \mathit{SKIP}\\ (a \to \mathit{SKIP} |||_\Theta \mathit{SKIP}) &\equiv a \to \mathit{SKIP}\\ (a \to \mathit{SKIP} |_\Theta \mathit{SKIP}) &\equiv (a \to \mathit{SKIP} \,\Box\, \mathit{SKIP}) \sqcap \mathit{SKIP}\end{aligned}$$

2. Using $\Theta = \varnothing$ & $\Delta = \{\checkmark\}$

$$\begin{aligned}(a \to \mathit{STOP}) |||_\Theta \mathit{SKIP} &\equiv\; a \to \mathit{STOP}\\ (a \to \mathit{STOP} \|_\Delta \mathit{SKIP}) &\equiv\; (a \to \mathit{STOP}) |||_\Theta \mathit{SKIP}\\ (a \to \mathit{STOP} |_\Theta \mathit{SKIP}) &\equiv\; (a \to \mathit{SKIP} \,\Box\, \mathit{SKIP}) \sqcap \mathit{SKIP}\end{aligned}$$

From the above:

$$a \to \mathit{STOP} |_\Theta \mathit{SKIP} \;\equiv\; a \to \mathit{SKIP} |_\Theta \mathit{SKIP}$$

3. Using $\Theta = \varnothing$ & $\Delta = \{\checkmark\}$

$$\begin{aligned}a \to \mathit{SKIP} \|_\varnothing b \to \mathit{SKIP} &\equiv a \to \mathit{SKIP} |||_\varnothing b \to \mathit{SKIP}\\ &\equiv (a \to b \to \mathit{SKIP}) \,\Box\, (b \to a \to \mathit{SKIP})\\ a \to \mathit{SKIP} |_\varnothing b \to \mathit{SKIP} &\equiv (a \to (\mathit{SKIP} \sqcap (\mathit{SKIP} \,\Box\, b \to \mathit{SKIP})))\\ &\qquad \Box\, (b \to (\mathit{SKIP} \sqcap (\mathit{SKIP} \,\Box\, a \to \mathit{SKIP})))\end{aligned}$$

Appendix C: Inhibited & Irrelevant Event Types for 3 Processes

Inhibited events: due to the first synchronisation set that has effect on the process – *Pi*, *Qi*, *Ri*.

E.g. $P\|_{\Lambda_1}(Q\|_{\Lambda_2}R)$, *Pi* events are due to Λ_1, *Qi* and *Ri* events are due to Λ_2.

Inhibited private events: are private asynchronous events under the first synchronisation set but are then inhibited by the second synchronisation set which has effect on the process – *(Pa)i*, *(Qa)i*, *(Ra)i*.

E.g. $P\|_{\Lambda_1}(Q\|_{\Lambda_2}R)$ only *(Qa)i* and *(Ra)i* events are present, they are not in Λ_2 but are in Λ_1. No *(Pa)i* events are present since only one synchronisation set affects *P*.

Inhibited synchronous events: – *(PQs)i*, *(PRs)i*, *(QRs)i*.

Inhibited common events: – *(PQa)i*, *(PRa)i*, *(QRa)i*.

Irrelevant synchronous events: – *PQis*, *PRis*, *QRis*.

Irrelevant events: – *PQRi*.