Topics in Concurrency

Harri Bell-Thomas

Contents:

Simple Parallelism and Non-Determinism

Communication by shared variables — introduce parallel composition with the $\|$ operator.
Parallelism by non-deterministic interleaving.
↳ thus study of parallelism ≥ study of non-determinism.

Dijkstra's Language of Guarded Commands.

Booleans expression — b, Arithmetic expressions — a.

Commands — skip | abort | $X := a$ | $c_0 ; c_1$ | if gc fi | do gc od

Guarded Commands — $gc ::= b \to c \mid gc_0 \,[\!]\, gc_1$

$$\frac{\langle b, \sigma\rangle \to \mathit{true}}{\langle b \to c, \sigma\rangle \to \langle c, \sigma\rangle}$$

$$\frac{\langle gc_0, \sigma\rangle \to \langle c, \sigma'\rangle}{\langle gc_0 \,[\!]\, gc_1, \sigma\rangle \to \langle c, \sigma'\rangle} \qquad \frac{\langle gc_1, \sigma\rangle \to \langle c, \sigma'\rangle}{\langle gc_0 \,[\!]\, gc_1, \sigma\rangle \to \langle c, \sigma'\rangle}$$

$$\frac{\langle b, \sigma\rangle \to \mathit{false}}{\langle b \to c, \sigma\rangle \to \text{fail}}$$

$$\frac{\langle gc_0, \sigma\rangle \to \text{fail} \qquad \langle gc_1, \sigma\rangle \to \text{fail}}{\langle gc_0 \,[\!]\, gc_1, \sigma\rangle \to \text{fail}}$$

`abort` has no rules

Conditional:

$$\frac{\langle gc, \sigma\rangle \to \langle c, \sigma'\rangle}{\langle \texttt{if } gc \texttt{ fi}, \sigma\rangle \to \langle c, \sigma'\rangle}$$

no rule in case $\langle gc, \sigma\rangle \to$ fail; then conditional behaves like `abort`

Loop:

$$\frac{\langle gc, \sigma\rangle \to \text{fail}}{\langle \texttt{do } gc \texttt{ od}, \sigma\rangle \to \sigma}$$

$$\frac{\langle gc, \sigma\rangle \to \langle c, \sigma'\rangle}{\langle \texttt{do } gc \texttt{ od}, \sigma\rangle \to \langle c; \texttt{do } gc \texttt{ od}, \sigma'\rangle}$$

in case $\langle gc, \sigma\rangle \to$ fail, the loop behaves like `skip`:

$$\langle \texttt{skip}, \sigma\rangle \to \sigma$$

Example: Euclid's Algorithm

[pre-conditions: $X = m \wedge Y = n \wedge m > 0 \wedge n > 0$]

do $(X > Y \rightarrow X := X - Y) \,[\!]\, (Y > X \rightarrow Y := Y - X)$ od

[post-condition: $X = Y = \gcd(m, n)$]

Communicating Processes

Extend GCL with synchronisation — introduce the notion of channels.

$\alpha ! a$ — evaluate a and send the value on channel α.

$\alpha ? X$ — receive value on channel α and store it in X.

NB. Communication is synchronised and only one process listening on the channel receives.

Transitions may now carry labels when there's a possible of inter-process communication.

$$\frac{}{\langle \alpha ? X, \sigma \rangle \xrightarrow{\alpha ? n} \sigma[n/X]} \qquad \frac{\langle a, \sigma \rangle \rightarrow n}{\langle \alpha ! a, \sigma \rangle \xrightarrow{\alpha ! n} \sigma}$$

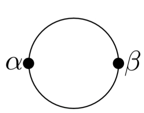
$$\frac{\langle c_0, \sigma \rangle \xrightarrow{\lambda} \langle c_0', \sigma' \rangle}{\langle c_0 \parallel c_1, \sigma \rangle \xrightarrow{\lambda} \langle c_0' \parallel c_1, \sigma' \rangle} \quad (\lambda \text{ might be empty label}) + \text{symmetric}$$

$$\frac{\langle c_0, \sigma \rangle \xrightarrow{\alpha ? n} \langle c_0', \sigma' \rangle \qquad \langle c_1, \sigma \rangle \xrightarrow{\alpha ! n} \langle c_1', \sigma \rangle}{\langle c_0 \parallel c_1, \sigma \rangle \rightarrow \langle c_0' \parallel c_1', \sigma' \rangle} + \text{symmetric}$$

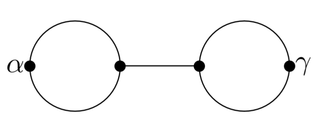
$$\frac{\langle c, \sigma \rangle \xrightarrow{\lambda} \langle c', \sigma' \rangle}{\langle c \setminus \alpha, \sigma \rangle \xrightarrow{\lambda} \langle c' \setminus \alpha, \sigma' \rangle} \quad \lambda \not\equiv \alpha ? n \text{ or } \alpha ! n$$

(Interface Diagrams)

Diagrammatic Views



do $\alpha ? X \rightarrow \beta ! X$ od



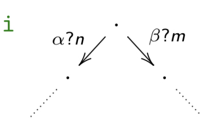
(do $\alpha ? X \rightarrow \beta ! X$ od

$\parallel$ do $\beta ? X \rightarrow \gamma ! X$ od) $\setminus \beta$

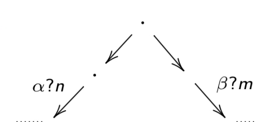
↳ restrict β to be internal-only channel.

NB: Internal vs External Choice

if $(true \wedge \alpha ? X \rightarrow c_0) \,[\!]\, (true \wedge \beta ? X \rightarrow c_1)$ fi



if $(true \rightarrow (\alpha ? X; c_0)) \,[\!]\, (true \rightarrow (\beta ? X; c_1))$ fi



These processes aren't equivalent; consider both's deadlock capacity.

CCS — The Calculus of Communicating Systems.

Simplifies GCL by removing the store.

Syntax: Processes:

$$
\begin{array}{rcll}
p & ::= & \mathbf{nil} & \text{nil process} \\
 & \mid & (\tau \to p) & \text{silent/internal action} \\
 & \mid & (\alpha!a \to p) & \text{output} \\
 & \mid & (\alpha?x \to p) & \text{input} \\
 & \mid & (b \to p) & \text{Boolean guard} \\
 & \mid & p_0 + p_1 & \text{non-deterministic choice} \\
 & \mid & p_0 \parallel p_1 & \text{parallel composition} \\
 & \mid & p \backslash L & \text{restriction } (L \text{ a set of channel identifiers}) \\
 & \mid & p[f] & \text{relabelling } (f \text{ a function on channel identifiers}) \\
 & \mid & P(a_1, \cdots, a_k) & \text{process identifier}
\end{array}
$$

Process definitions:

$$P(x_1, \cdots, x_k) \stackrel{\text{def}}{=} p$$

(free variables of $p \subseteq \{x_1, \cdots, x_k\}$)

Example:

$$
\dfrac{\dfrac{\dfrac{\dfrac{(\alpha!3 \to \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil}}{(\alpha!3 \to \mathbf{nil}) + P \xrightarrow{\alpha!3} \mathbf{nil}}}{((\alpha!3 \to \mathbf{nil}) + P) \parallel (\tau \to \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil} \parallel (\tau \to \mathbf{nil})} \qquad (\alpha?x \to \mathbf{nil}) \xrightarrow{\alpha?3} \mathbf{nil}}{(((\alpha!3 \to \mathbf{nil}) + P) \parallel (\tau \to \mathbf{nil})) \parallel (\alpha?x \to \mathbf{nil}) \xrightarrow{\tau} (\mathbf{nil} \parallel (\tau \to \mathbf{nil})) \parallel \mathbf{nil}}}{(((\alpha!3 \to \mathbf{nil} + P) \parallel \tau \to \mathbf{nil}) \parallel \alpha?x \to \mathbf{nil}) \setminus \{\alpha\} \xrightarrow{\tau} ((\mathbf{nil} \parallel \tau \to \mathbf{nil}) \parallel \mathbf{nil}) \setminus \{\alpha\}}
$$

Linking Processes:

def^n $p \frown q \equiv (p[c/out] \parallel q[c/in]) \setminus c$ (where c is a fresh channel)

Buffer, $B \equiv in?x \to (out!x \to B)$

N-ary buffer: $\underbrace{B \frown B \frown \ldots \frown B}_{n \text{ times}}$

Buffer with acknowledgements, $D \equiv in?x \to out!x \to ackout? \to ackin! \to D$

Euclid:

def^n $Step \equiv in?x \to in?y \to (x = y \to gcd!x \to nil \; +$
$x < y \to out!x \to out!(y-x) \to nil \; +$
$y < x \to out!(x-y) \to out!y \to nil)$

def^n $Euclid \equiv Step \frown Euclid$

Pure CCS.

def^n $\Sigma_n \alpha?n.\,p[n/x]$, sum over all possible values of n and substitute it for location x instead of using the store.

Intuition: $\alpha?n$ and $\alpha!n$ are complementary actions; synchronisation only occurs on compl. actions.

Syntax:

p ::=	$\lambda.p$	prefix	λ ranges over τ, a, $\bar{a}$ for any action a
$\mid$	$\sum_{i \in I} p_i$	sum	I is an indexing set
$\mid$	$p_0 \parallel p_1$	parallel	
$\mid$	$p \backslash L$	restriction	L a set of actions
$\mid$	$p[f]$	relabelling	f a function on actions
$\mid$	P		process identifier

Transition Systems.

Given a CCS process p;

$p = (S, i, L, tran)$

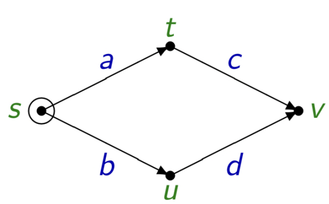
where S is the set of states

i is the initial state

L is the set of labels

$tran \subseteq S \times L \times S$, the transition relation

eg.



$$
\begin{aligned}
S &= \{s, t, u, v\} \\
i &= s \\
L &= \{a, b, c, d\} \\
tran &= \{\ (s, a, t), \\
&\qquad (s, b, u), \\
&\qquad (t, c, v), \\
&\qquad (u, d, v)\ \}
\end{aligned}
$$

CCS Pure CCS

p	$\widehat{p}$	
nil	**nil**	
$(\tau \to p)$	$(\tau.\widehat{p})$	
$(\alpha!a \to p)$	$\overline{\alpha m}.\widehat{p}$	where a evaluates to m
$(\alpha?x \to p)$	$\sum_{m \in \mathbf{Num}} \alpha m.\widehat{p[m/x]}$	
$(b \to p)$	$\widehat{p}$ **nil**	if b evaluates to true if b evaluates to false
$p_0 + p_1$	$\widehat{p_0} + \widehat{p_1}$	
$p_0 \parallel p_1$	$\widehat{p_0} \parallel \widehat{p_1}$	
$p \setminus L$	$\widehat{p} \setminus \{\alpha m \mid \alpha \in L \;\&\; m \in \mathbf{Num}\}$	
$P(a_1, \cdots, a_k)$	$P_{m_1, \cdots, m_k}$	where a_i evaluates to m_i

For every definition $P(x_1, \cdots, x_k)$, we have a collection of definitions $P_{m_1, \ldots, m_k}$ indexed by $m_1, \cdots, m_k \in \mathbf{Num}$.

Recursive Alternative

Replace $P \equiv p$ with $rec(P = p)$, $\dfrac{p[rec(P=p)/P] \xrightarrow{\lambda} p'}{rec(P=p) \xrightarrow{\lambda} p'}$

eg. $rec(P = a.nil + b.P)$

Multiple definitions : $P \to rec_1(P=p, Q=q)$, $Q \to rec_2(P=p, Q=q)$

Generally : $P_j \to rec_j(P_i = p_i)_{i \in I}$, or $P_j \to rec_j(\vec{P} = \vec{p})$

Language Equivalences

A process trace is a (possibly infinite) sequence of actions $(a_1, a_2, \dots a_i, a_{i+1}, \dots)$ such that $p \xrightarrow{a_1} p_1 \xrightarrow{a_2} \dots p_{i-1} \xrightarrow{a_i} p_i \xrightarrow{a_{i+1}} \dots$

Two processes are trace equivalent iff they have the same sets of traces.

A trace is maximal if it cannot be extended (either infinite or ends in a state from while there is no transition).

Processes are completed trace equivalent if they have the same sets of maximal traces.

Bisimulation

Def^n a strong bisimulation is a relation R between states where:

$$\begin{aligned} p\,R\,q \iff\ & \forall \alpha, p'.\ p \xrightarrow{\alpha} p' \implies \exists q'.\ q \xrightarrow{\alpha} q' \wedge p'\,R\,q' \\ \wedge\ & \forall \alpha, q'.\ q \xrightarrow{\alpha} q' \implies \exists p'.\ p \xrightarrow{\alpha} p' \wedge p'\,R\,q' \end{aligned}$$

Strong bisimilarity is an equivalence on states; $p \sim q$ iff $p\,R\,q$ for some bisimulation R

To show $p \sim q$ we give the relation R.

Bisimilarity is a congruence: $p \sim q \to \alpha.p \sim \alpha.q$

$\wedge\ p + r \sim q + r$

$\wedge\ p \| r \sim q \| r$

$\wedge\ p \backslash L \sim q \backslash L$

$\wedge\ p[f] \sim q[f]$

$+$ and $\|$ are commutative and associative w.r.t. $\sim$, unit nil.

For R, S, R_i $(i \in I)$, all strong bisimulations, then;

- The identity relation is a bisimulation
- R^{-1} (or R^{op}) is a bisimulation
- $R \circ S$ (where the set of states match up) is a bisimulation
- $\bigcup_{i \in I} R_i$ (for all over the same transition system) is a bisimulation

First 3 imply $\sim$ is an equivalence relation, last that $\sim$ is a bisimulation.

CCS Expansion. In general, $p \sim \Sigma\{\alpha.p' \mid p \xrightarrow{\alpha} p'\}$

We can represent anything as prefixing and sums.

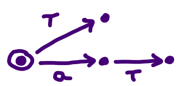
Suppose $p \sim \Sigma_{i \in I}\, \alpha_i . p_i$ and $q \sim \Sigma_{j \in J}\, \beta_j . q_j$

* $p \setminus L \sim \Sigma\{\alpha_i.(p_i \setminus L) \mid \alpha_i \notin L\}$
* $p[f] \sim \Sigma\{ f(\alpha_i).\, p_i[f] \mid i \in I\}$
* $p \parallel q \sim \sum_{i \in I} \alpha_i.(p_i \parallel q) + \sum_{j \in J} \beta_j.(p \parallel q_j) + \Sigma\{\tau.(p_i \parallel q_j) \mid \alpha_i = \bar{\beta}_j\}$

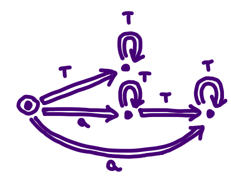
Weak Bisimulation.

Def^n $\overset{\tau}{\Rightarrow} \equiv (\overset{\tau}{\rightarrow}{}^*)$

$\overset{\alpha}{\Rightarrow} \equiv (\overset{\tau}{\Rightarrow} \overset{\alpha}{\rightarrow} \overset{\tau}{\Rightarrow})$



becomes



Def^n Weak Bisimulation is a relation R between state where;

$p \mathrel{R} q \Rightarrow \forall \alpha, p'.\ p \overset{\alpha}{\Rightarrow} p' \rightarrow \exists q'.\ q \overset{\alpha}{\Rightarrow} q' \wedge p' \mathrel{R} q'$

$\forall \alpha, q'.\ q \overset{\alpha}{\Rightarrow} q' \rightarrow \exists p'.\ p \overset{\alpha}{\Rightarrow} p' \wedge p' \mathrel{R} q'$

Note: Weak Bisimulation is not a congruence (merely an observational one).

Specification Logics for Processes

Finitary Hennessy-Milner Logic

Assertions:

$$A ::= T \mid F \mid A_0 \wedge A_1 \mid A_0 \vee A_1 \mid \neg A \mid \langle\lambda\rangle A \mid \langle - \rangle A \mid [\lambda]A \mid [-]A$$

Satisfaction: $s \vDash A$

$s \vDash T$ always

$s \vDash F$ never

$s \vDash A_0 \wedge A_1$ if $s \vDash A_0$ and $s \vDash A_1$

$s \vDash A_0 \vee A_1$ if $s \vDash A_0$ or $s \vDash A_1$

$s \vDash \neg A$ if not $s \vDash A$

$s \vDash \langle\lambda\rangle A$ if there exists s' s.t. $s \xrightarrow{\lambda} s'$ and $s' \vDash A$

$s \vDash \langle - \rangle A$ if there exist s', λ s.t. $s \xrightarrow{\lambda} s'$ and $s' \vDash A$

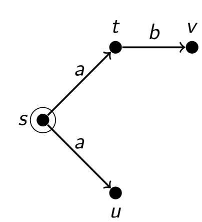
$s \vDash [\lambda]A$ iff for all s' s.t. $s \xrightarrow{\lambda} s'$ have $s' \vDash A$

$s \vDash [-]A$ iff for all s', λ s.t. $s \xrightarrow{\lambda} s'$ have $s' \vDash A$

Alternatively, derived assertions

$$[\lambda]A \equiv \neg\langle\lambda\rangle\neg A \qquad [-]A \equiv \neg\langle - \rangle \neg A$$

Example:



$s \vDash \langle a \rangle T$

$s \vDash [a] T$

$s \nvDash [-] F$

$s \vDash \langle a \rangle \langle b \rangle T$

$s \nvDash [a] \langle b \rangle T$

Non-finitary Hennessey-Milner logic allows an infinite conjunction.

Semantics: $s \models \bigwedge_{i \in A} A_i$ iff $s \models A_i$ for all $i \in I$

Def^n $p \asymp q$ iff for all assertions A of H-M logic $p \models A$ iff $q \models A$

Theorem : $\asymp = \sim$. Using this we can demonstrate non-bisimilarity.

ASIDE: Knaster-Tarski Fixed Point Theorem

Specialised to lattice derived from the Power set and inclusion ($\subseteq$) poset operator.

Let $\phi : Pow(S) \to Pow(S)$ be a monotonic function $[X \subseteq Y \to \phi(X) \subseteq \phi(Y)]$

Def^n Prefixed points : $\phi(S) \subseteq S$

Postfixed points : $S \subseteq \phi(S)$

Fixed points : $S = \phi(S)$

Theorem : Minimum Fixed Point, $m = \bigcap \{ S \subseteq X \mid \phi(S) \subseteq S \}$; $\mu X . \phi(X)$

Maximum Fixed Point, $n = \bigcup \{ S \subseteq X \mid S \subseteq \phi(S) \}$; $\nu X . \phi(X)$

Lemma: $R \subseteq \phi(R)$ iff R is a strong bisimulation.

$$\begin{aligned} \sim &= \bigcup \{ R \mid R \text{ is a bisimulation} \} \\ &= \bigcup \{ R \mid R \subseteq \phi(R) \} \\ &= \nu X . \phi(X) \end{aligned}$$

Modal μ-calculus

$A ::= T \mid F \mid A_0 \wedge A_1 \mid A_0 \vee A_1 \mid \neg A \mid \langle \lambda \rangle A \mid \langle - \rangle A \mid X \mid \nu X . A$

NB: To guarantee monotonicity (and thus the existence of the fixed point) we require the variable X to only occur positively in A (for $\nu X . A$). Such, X occurs under an even number of $\neg$s.

$s \models \nu X . A$ iff $s \in \nu X . A$, ie. $s \in \bigcup \{ S \subseteq P \mid S \subseteq A[S/X] \}$ (ϕ here is $S \mapsto A[S/X]$)

Note: $\mu X . A \equiv \neg \nu X . (\neg A[\neg X / X])$

$= \bigcap \{ S \subseteq P \mid A[S/X] \subseteq S \}$

Approximants:

Let $\phi : P(S) \to P(S)$ be monotonic.

Defn ϕ is $\cap$-continuous iff for all decreasing chains $[X_0 \supseteq X_1 \supseteq \dots X_n \supseteq \dots]$ we have $\bigcap_{n\in\omega} \phi(X_n) = \phi(\bigcap_{n\in\omega} X_n)$

ϕ is $\cup$-continuous iff for all increasing chains $[X_0 \subseteq X_1 \subseteq \dots X_n \subseteq \dots]$ we have $\bigcup_{n\in\omega} \phi(X_n) = \phi(\bigcup_{n\in\omega} X_n)$

Both certainly hold if S is finite.

If ϕ is $\cap$-continuous, $\nu X.\phi(X) = \bigcap_{n\in\omega} \phi^n(S)$

If ϕ is $\cup$-continuous, $\mu X.\phi(X) = \bigcup_{n\in\omega} \phi^n(\emptyset)$

Proposition: for a finite state system, $s \models \nu X.\langle a\rangle X$ iff there exists an infinite sequence of a transitions from s.

There are infinite transition systems where $\phi(X) = \langle a\rangle X$ isn't $\cap$-continuous.

Bisimilarity.

* Modal μ-calculus for finite processes can be encoded in infinitary HM logic.
* Bisimilar processes p & q satisfy the same modal-μ assertions.
* But modal-μ equivalence doesn't imply bisimilarity (no infinitary conjunction)

CTL: Computation Tree Logic

CTL is a path based logic; a path from state is a maximal sequence of states, $\pi = (\pi_0, \pi_1, \dots \pi_i, \dots)$, such that $s = \pi_0$ and $\pi_i \to \pi_{i+1}$ for all i.

$$A ::= At \mid A_0 \wedge A_1 \mid A_0 \vee A_1 \mid \neg A \mid T \mid F \mid \mathrm{EX}\, A \mid \mathrm{EG}\, A \mid \mathrm{E}[A_0 \,\mathrm{U}\, A_1]$$

$s \models \mathrm{EX}\, A$	iff	Exists a path from s along which the neXt state satisfies A
$s \models \mathrm{EG}\, A$	iff	Exists a path from s along which Globally each state satisfies A
$s \models \mathrm{E}[A \,\mathrm{U}\, B]$	iff	Exists a path from s along which A holds Until B holds

Derivations :

$$AX\ B \equiv \neg EX \neg B$$
$$EF\ B \equiv E[T \cup B]$$
$$AG\ B \equiv \neg EF \neg B$$
$$AF\ B \equiv \neg EG \neg B$$
$$A[B \cup C] \equiv \neg E[\neg C \cup \neg B \wedge \neg C] \wedge \neg EG \neg C$$

CTL translation to modal-μ:

$$EX\ a \equiv \langle - \rangle A$$
$$EG\ a \equiv \nu Y.\ A \wedge ([-]F \vee \langle - \rangle Y)$$
$$E[a \cup b] \equiv \mu Z. B \vee (A \wedge \langle - \rangle Z)$$

Proposition

$$s \models \nu Y.A \wedge ([-]F \vee \langle - \rangle Y)$$

in a finite-state transition system iff there exists a path π from s such that $\pi_i \models A$ for all i.

Proof:
Take $\varphi(Y) \stackrel{\text{def}}{=} A \wedge ([-]F \vee \langle - \rangle Y)$.

$$\nu Y.\varphi(Y) = \bigcap_{n \in \omega} \varphi^n(T) \quad \text{where} \quad T \supseteq \varphi(T) \supseteq \cdots$$

since φ is monotonic and $\bigcap$-continuous due to the set of states being finite.
By induction, for $n \geq 1$

$s \models \varphi^n(T)$ iff there is a path of length $\leq n$ from s along which all states satisfy A and the final state has no outward transition
or there is a path of length n from s along which all states satisfy A and the final state has some outward transition

Assuming the number of states is k, we have

$$\varphi^k(T) = \varphi^{k+1}(T)$$

and hence $\nu Y.\varphi(Y) = \varphi^k(T)$.

$s \models \nu Y.\varphi(Y)$ iff $s \models \varphi^k(T)$
iff there exists a maxmial A path of length $\leq k$ from s
or there exists a necessarily looping A path of length k from s □

Model Checking

Local model checking with the "silly idea" reduction lemma:

$$p \in \nu X.\phi(x) \iff p \in \phi(\nu X.\{p\} \vee \phi(X))$$

Extend syntax, $A ::= \cdots \mid U \mid \nu X\{p_1, \cdots p_n\}.A$

where : U is an arbitrary subset of states.

$$\nu X\{p_1, \cdots p_n\}.A = \bigcup \{ U \subseteq S \mid U \subseteq \{p_1, \cdots p_n\} \cup A[U/X] \}$$

Algorithm. Given a transition system with a set of basic assertions, $\{U, V, \ldots\}$;

$$
\begin{array}{lll}
p \vdash U & \longrightarrow \text{true} & \text{if } p \in U \\
p \vdash U & \longrightarrow \text{false} & \text{if } p \notin U \\
p \vdash T & \longrightarrow \text{true} & \\
p \vdash F & \longrightarrow \text{false} & \\
p \vdash \neg B & \longrightarrow \text{not}(p \vdash B) & \\
p \vdash A \wedge B & \longrightarrow p \vdash A \text{ and } p \vdash B & \\
p \vdash A \vee B & \longrightarrow p \vdash A \text{ or } p \vdash B & \\
p \vdash \langle a \rangle B & \longrightarrow q_1 \vdash B \text{ or } \ldots \text{ or } q_n \vdash B, & \{q_1, \ldots q_n\} = \{q \mid p \xrightarrow{a} q\} \\
p \vdash \nu X\{\vec{r}\}.B & \longrightarrow \text{true} & \text{if } p \in \{\vec{r}\} \\
p \vdash \nu X\{\vec{r}\}.B & \longrightarrow p \vdash B[\nu X.\{p, \vec{r}\}.B / X] & \text{if } p \notin \{\vec{r}\}
\end{array}
$$

Example:

for $P = a.(a.nil + a.P)$

check $P \vdash \nu X.\langle a \rangle X.$

$\rightarrow P \vdash \langle a \rangle (\nu X\{p\}.\langle a \rangle X)$

$\rightarrow Q \vdash \nu X\{p\}.\langle a \rangle X$

$\rightarrow Q \vdash \langle a \rangle (\nu X\{p, q\}.\langle a \rangle X)$

$\rightarrow P \vdash \nu X\{p, q\}.\langle a \rangle X$

$\rightarrow$ true

and check $P \vdash \mu Y.[-]F \vee \langle - \rangle Y$

NB: $\mu Y.[-]F \vee \langle - \rangle Y = \neg \nu Y.\neg([-]F \vee \langle - \rangle \neg Y)$

$\rightarrow P \vdash \neg \nu Y.\neg([-]F \vee \langle - \rangle \neg Y)$

$\rightarrow P \vdash \neg(\neg[-]F \vee \langle - \rangle \neg \nu Y\{p\}.\neg([-]F \vee \langle - \rangle \neg Y))$

$\rightarrow P \vdash [-]F \vee \langle - \rangle \neg \nu Y\{p\}.\neg([-]F \vee \langle - \rangle \neg Y))$

$\rightarrow Q \vdash \neg \nu Y\{p\}.\neg([-]F \vee \langle - \rangle \neg Y))$

$\rightarrow Q \vdash \neg(\neg[-]F \vee \langle - \rangle \neg Y))$

$\rightarrow Q \vdash [-]F \vee \langle - \rangle \neg Y))$

$\rightarrow R \vdash \neg \nu Y\{p, q\}.\neg([-]F \vee \langle - \rangle \neg Y))$

$\rightarrow R \vdash \neg(\neg[-]F \vee \langle - \rangle \neg Y)$

$\rightarrow R \vdash [-]F \vee \langle - \rangle \neg Y$

$\rightarrow$ true

Well Founded Induction.

NB. A binary relation $\prec$ on set A is well founded iff there are no infintely descending chains.
$\dots \prec a_n \prec \dots \prec a_1 \prec a_0$

Principle:

Let $\prec$ be a well founded relation on a set A, and P a property on A.
Then $\forall a \in A.\ P(a)$
iff $\forall a \in A.\ ((\forall b \prec a.\ P(b)) \Longrightarrow P(a))$
[NB. Vacuosly true when a has no predecessors]

Proof in notes: $(p \vdash A) \to^{*} t \iff (p \vDash A) = t$

Petri Nets

Preamble: ∞-multisets.
Multisets – elements can appear multiple times.
$\omega^{\infty} = \omega \cup \{\infty\}$

An ∞-multiset over a set X is a function, $f: X \to \omega^{\infty}$, number of times an element appears.
$f \leq g$ iff $\forall x \in X.\ f(x) \leq g(x)$
$f + g$ is the ∞-multiset such that $\forall x \in X.\ (f+g)(x) = f(x) + g(x)$
For multiset g, $g \leq f$, $\forall x \in X.\ (f-g)(x) = f(x) - g(x)$

General Petri Net.

* P, a set of conditions
* T, a set of events
* ${}^{\bullet}t$, a precondition map assigning each event t a multiset of conditions.
* $t^{\bullet}$, a postcondition map assigning each event t an ∞-multiset of conditions.
* Cap, a capacity map, an ∞-multiset of conditions, assigning a capacity in ω^{∞} to each condition.
* M, a marking, an ∞-multiset, $M \leq Cap$, specifying how many tokens are in each condition.
* Token Game:

  For M, M' markings, t an event:
  $M \xrightarrow{t} M'$ iff ${}^{\bullet}t \leq M$ and $M' = M - {}^{\bullet}t + t^{\bullet}$
  t can occur at M iff ${}^{\bullet}t \leq M$ and $M - {}^{\bullet}t + t^{\bullet} \leq Cap$

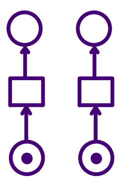
Basic Petri Nets. [Consider sets instead of multisets]

* B, a set of conditions.
* E, a set of events.
* ${}^\bullet e$, $e^\bullet$, pre- and postcondition subsets of B.
* Capacity of any condition is implicitly set to be 1, $\forall b \in B.\ Cap(b) = 1$.
* M, a marking, is now just a subset of conditions;

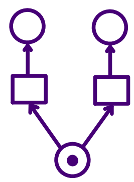
$$M \xrightarrow{e} M' \text{ iff } {}^\bullet q \subseteq M \text{ and } (M \setminus {}^\bullet e) \cap e^\bullet = \emptyset \text{ and } M' = (M \setminus {}^\bullet e) \cup e^\bullet$$

Examples:

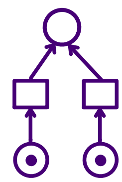
Concurrency



Forwards Conflict

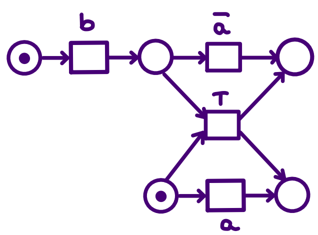


Backwards Conflict



Contact

$P = a.nil \parallel b.\bar{a}.nil$



Safe Nets — there is no marking reachable from the initial in which contact occurs.
Contact occurs when ${}^\bullet e \subseteq M$, $(M \setminus e^\bullet) \cap {}^\bullet e \neq \emptyset$

Persistent Conditions — conditions which after they hold once will persist thereafter (eg broadcasts).

Now; $M \xrightarrow{e} M'$ iff ${}^\bullet e \subseteq M$ and $(e^\bullet \cap (M \setminus (Persistent \cup {}^\bullet e)) = \emptyset$
and $M' = (M \setminus {}^\bullet e) \cup e^\bullet \cup (M \cap Persistent)$

Drawn as ◎.

Security Protocols

Requires analysis based on causal dependencies — event based reasoning.

Example: Needham – Schröeder – Lowe Protocol.

SPL

Infinite set, Names = $\{m, n, \dots A, B, \dots\}$, with name variables $x, y, \dots X, Y, \dots$
Messages are ranged over by message variables, $\psi, \psi', \psi_1, \dots$
Indices shall be used to identify components in parallel compositions.

Name expressions	$v ::= n \mid A \mid \dots \mid x \mid X$
Key expressions	$K ::= Pub(v) \mid Priv(v) \mid Key(v, v')$
Messages	$M ::= \psi \mid v \mid k \mid M_1, M_2 \mid \{M\}_k$
Processes	$p ::= \text{out new } \vec{x}\, M.p \mid \text{in pat } \vec{x}\vec{\psi} M.p \mid \|_{i \in I}\, p_i$

eg. out $M.p$ (list of new variables is empty)
in $M.p$ (list of name and message variables are precisely the free variables in M)
nil (the empty parallel composition)
$!p$ (replication, $\longrightarrow \|_{i \in \omega}\, p$)

NSL in SPL.

$Init(A,B) \equiv$ out new x $\{x, A\}_{PUB(B)}$
in $\{x, y, B\}_{PUB(A)}$
out $\{y\}_{PUB(B)}$

$Resp(B) \equiv$ in $\{x, Z\}_{PUB(B)}$.
out new y $\{x, y, B\}_{PUB(Z)}$.
in $\{y\}_{PUB(B)}$

Dolev-Yao Assumptions.

Viewing all output messages as persistant, DY agent build new messages based on existing ones.

$$
\begin{aligned}
Spy_1 &\equiv \text{in } \psi_1.\text{in } \psi_2.\text{ out } (\psi_1, \psi_2) \\
Spy_2 &\equiv \text{in } (\psi_1, \psi_2).\text{ out } \psi_1.\text{ out } \psi_2 \\
Spy_3 &\equiv \text{in } X.\text{in } \psi.\text{ out } \{\psi\}_{Pub(X)} \\
Spy_4 &\equiv \text{in } Priv(X).\text{in } \{\psi\}_{Pub(X)}.\text{ out } \psi
\end{aligned}
$$

$$Spy \equiv \|_{i \in \{1,2,3,4\}}\, Spy_i$$

Thus; $P_{spy} = !Spy$
$P_{init} = \|_{X,Y \in Agents}\, Init(X, Y)$
$P_{resp}\ \|_{Z \in Agents}\, Resp(Z)$
$NSL = \|_{i \in \{spy, resp, init\}}\, P_i$

A Configuration is a tuple, $\langle p, s, t\rangle$;

- p is a closed process term.
- s is a finite subset of names (names already in use).
- t is a subset of closed messages (messages already outputted to the network).

A Configuration is proper iff;

- $names(p) \subseteq s$
- $A \in s$ for every agent identifier A.
- $\bigcup\{names(M) \mid M \in t\} \subseteq s$.

Transitions are labelled with actions; $\alpha ::= \text{out new } \vec{n}\ M \mid \text{in } M \mid i:\alpha$

Output: if $\vec{n}$ all distinct and not in s

$$\langle \text{out new } \vec{x}\ \ M.p, s, t\rangle \xrightarrow{\text{out new } \vec{n}\ \ M[\vec{n}/\vec{x}]} \langle p[\vec{n}/\vec{x}], s \cup \{\vec{n}\}, t \cup \{M[\vec{n}/\vec{x}]\}\rangle$$

Input: if $M[\vec{n}/\vec{x}][\vec{N}/\vec{\psi}] \in t$

$$\langle \text{in pat } \vec{x}, \vec{\psi}\ M.p, s, t\rangle \xrightarrow{\text{in } M[\vec{n}/\vec{x}][\vec{N}/\vec{\psi}]} \langle p[\vec{n}/\vec{x}][\vec{N}/\vec{\psi}], s, t\rangle$$

Parallel:

$$\frac{\langle p_j, s, t\rangle \xrightarrow{\alpha} \langle p_j', s', t'\rangle \quad j \in I}{\langle \|_{i\in I}\ p_i, s, t\rangle \xrightarrow{j:\alpha} \langle \|_{i\in I}\ p_i', s', t'\rangle}$$

where $p_i' = p_i$ for $j \neq i$

Event Structures

Petri Nets can be unfolded into Occurrence nets; remove loops and forwards conflicts.
Occurrence nets are then translated into Event structures, removing conditions and adding a consistency relation.

Event Structures with Binary Conflict.

$\rightarrow (E, \leq, \#)$;

- E, a set of events, partially ordered by
- $\leq$, the causal dependency relation, and
- $\#$, a binary, reflexive, symmetric relation on E, representing conflicts.
  - $\hookrightarrow \forall e \in E. \{e' \mid e' \leq e\}$ is finite if $e \geq e_0 \# e_0' \leq e'$, then $e \# e'$.

e and e' are concurrent if $\neg(e \# e') \wedge e \not\leq e' \wedge e \not\geq e'$.

Configurations, $C^\infty(E)$, are the subsets $x \subseteq E$ which are;
- Consistent : $\forall e, e' \in x. \neg(e \# e')$.
- Down Closed : $\forall e, e'. \ e' \leq e \in x \Rightarrow e' \in x$.

For an event e, the set $[e] \equiv \{e' \in E \mid e' \leq e\}$ describes the whole causal history of e.
$x \subseteq x' \longrightarrow x$ is a subconfiguration / subhistory of x'.
$(C^\infty(E), E)$ is a domain. $C(E)$ is the set of all finite configurations.

General (Consistency-Based) Event Structures.
$\longrightarrow (E, \leq, \mathrm{Con})$;
Con is the family of non empty finite subsets of E, the consistency relation.
$\hookrightarrow \forall e \in E. \ \{e' \mid e' \leq e\}$ is finite,
$\forall e \in E. \ \{e\} \in \mathrm{Con}$,
$Y \subseteq X \in \mathrm{Con} \Rightarrow Y \in \mathrm{Con}$, and
$X \in \mathrm{Con} \wedge e \leq e' \in X \Rightarrow X \cup \{e\} \in \mathrm{Con}$

e and e' are concurrent if $\{e, e'\} \in \mathrm{Con} \wedge e \not\leq e' \wedge e' \not\leq e$.

For Configurations, define;
- Consistency : $\forall X \subseteq x. \ X \in \mathrm{Con}$.
- Down Closed : $\forall e, e'. \ e' \leq e \in x \Rightarrow e' \in x$.

Maps of Event Structures.
map ; $f : E \rightarrow E'$ such that $fx \in C(E') \wedge (e_1, e_2 \in x \wedge f(e_1) = f(e_2) \Rightarrow e_1 = e_2)$
When f is total it restricts to a bijection, $x \cong fx$ for any $x \in C(E)$.

Maps preserve concurrency and locally reflect causal dependency:
$e_1, e_2 \in x \ \wedge \ f(e_1) \leq f(e_2)$ [both $\downarrow$] $\Rightarrow e_1 \leq e_2$.
A total map is rigid when it preserves causal dependencies.

ASIDE : Partial Order, $a \leq b$.
a related to b, b not necessarily related to a.
$\leq$ is;
i. Reflexive, $a \leq a$.
ii. Antisymmetric, $a \leq b \wedge b \leq a \Rightarrow a = b$.
iii. Transitive.

Computation Paths: a partial order, $p = (|p|, \leq_p)$.
$\forall e \in |p|.\ \{e' \in |p| \mid e' \leq_p e\}$ is finite.

- Path p is prime iff it has a top element, $top(p)$
- $|p|$ simply means the set of events in the path.

- Rigid Inclusion between paths, $p = (|p|, \leq_p)$ and $q = (|q|, \leq_q)$.
  $p \hookrightarrow q$ iff $|p| \subseteq |q| \wedge \forall e \in |p|, e' \in |q|.\ e' \leq_p e \Leftrightarrow e' \leq_q e$.

Rigid Families.

A non-empty set of finite paths, $\mathcal{R}$, for which $p \hookrightarrow q \in \mathcal{R} \Rightarrow p \in \mathcal{R}$.
Set of paths closed under rigid embeddings.
eg. $\mathcal{C}(E)$ is a rigid family.

Rigid Structures → Event Structures.

A Rigid Family, $\mathcal{R}$, determines an event structure $Pr(\mathcal{R})$.
↳ the order of $\mathcal{C}(Pr(\mathcal{R}))$ is isomorphic to $(\mathcal{R}, \hookrightarrow)$.

$Pr(\mathcal{R})$ has events P, subset of the prime paths of $\mathcal{R}$.
Causal dependency is given by rigid inclusion.
Consistency given by compatibility with rigid inclusion.

Order isomorphism, $\phi_\mathcal{R} : (\mathcal{R}, \hookrightarrow) \cong (\mathcal{C}(Pr(\mathcal{R})), \subseteq)$, given by
$\forall q \in \mathcal{R}.\ \phi_\mathcal{R}(q) = \{p \in \mathcal{R} \mid p \hookrightarrow q\}$

Its inverse: $\theta_\mathcal{R}(x) = \bigcup x$ on $x \in \mathcal{C}(Pr(\mathcal{R}))$

Products of Event Structures.

Def^n $|A| \times_* |B| = \{(a,*) \mid a \in |A|\} \cup \{(a,b) \mid a \in |A| \wedge b \in |B|\} \cup \{(*,b) \mid b \in B\}$ [with partial projections, π_1, π_2]

Def^n $A \times B = Pr(\mathcal{R})$, where rigid family $\mathcal{R}$ satisfies:
$p \in \mathcal{R}$ iff

i. $|p| \subseteq |A| \times_* |B|$

ii. $\pi_1|p| \in \mathcal{C}(A) \wedge \pi_2|p| \in \mathcal{C}(B)$.
Also, projections are locally injective; $\forall c, c' \in |p|.\ \pi_1(c) = \pi_1(c') \Rightarrow c = c'$.
$\forall c, c' \in |p|.\ \pi_2(c) = \pi_2(c') \Rightarrow c = c'$.

iii. $\leq_p$ least relation s.t. $c \leq_p c'$ if $\pi_1(c) \leq_A \pi_1(c')$ or $\pi_2(c) \leq_B \pi_2(c')$.
Enforced by partial order, causal loops thrown away.

Augmentations.

Let E be an event structure with configuration x.

A path $p = (|p|, \leq_p)$ is an augmentation of x iff $|p| = x \wedge \forall e \in |p|, e' \in |E|.\ e' \leq_E e \Rightarrow e' \leq_p e$.

Define $\wedge$, a partial operation on augmentations; $\wedge : Aug(E) \times Aug(E) \to Aug(E)$ by taking,

$$p \wedge q = \begin{cases} (|p|, \leq_p \cup \leq_q) & \text{if } |p| = |q| \wedge (\leq_p \cup \leq_q)^* \text{ is antisymmetric} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Games and Strategies

Event Structures with Polarity, (A, Pol), where A is an event structure and $pol_A : A \to \{+, 0, -\}$, ascribing events with a polarity; $+$ for player, 0 for neutral, $-$ for opponent.

A Game shall be an event structure with no neutral moves.

NB. $x \subseteq^- y$ mean inclusion in which all intervening events are the Opponents.
$x \subseteq^+ y$ is the same for player or neutral events.

The Dual of a game, $A^\perp$, is the event structure of A with reverse polarities.
↳ a player strategy is a strategy in A, an opponent's one in $A^\perp$.

A Play in A (a polarised event structure) is an augmentation $p = (|p|, \leq_A)$ with $|p| \in C(A)$ s.t.
$\forall a, a' \in |p|.\ a' \rightarrowtriangle_p a \wedge (pol_A(a') = + \vee pol_A(a) = -) \Rightarrow a' \rightarrowtriangle_A a$.

* Write $Plays(A)$ for the set of plays in A.
* If A is a game, the only augmentations allowed of a play p (beyond the immediate causal dependencies of A) are those of the form $\ominus \rightarrowtriangle_p \oplus$

Strategies.

A bare strategy in A (a polarised event structure) is a rigid family $\sigma \subseteq Plays(A)$ that is receptive: $p \in \sigma \wedge |p| \subseteq^- x \in C(A) \Rightarrow \exists q \in \sigma.\ p \hookrightarrow q \wedge |q| = x$

Strategies as Maps of Event Structures.

For a bare strategy, $\sigma : A$, $top : Pr(\sigma) \to A$ is a total map on event structures that;

i. Preserve polarities.

ii. Satisfies Courtesy: $s' \rightarrowtriangle s \wedge pol(s') = + \wedge pol(s) = - \Rightarrow f_\sigma(s') \rightarrowtriangle_A f_\sigma(s)$.

iii. Satisfies Receptivity: $\forall x \in C(Pr(\sigma)).\ f_\sigma(x) \subseteq^- y \in C(A) \Rightarrow \exists! x' \in C(Pr(\sigma)).\ f_\sigma(x') = y$.

Maps to Strategies.

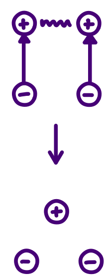
Let $f : S \to A$ be total map on event structures preserving polarity.

Def^n $\sigma(f) = \{ (fx, \leq_{fx}) \mid x \in C(S) \}$, a rigid family where ;

$$a' \leq_{fx} a \iff \exists s', s \in x.\ a' = f(s') \wedge a = f(s) \wedge s' \leq_S s$$

Proposition : the rigid image is a strategy, $\sigma(f) : A$, if f is receptive and courteous.
The converse may fail : $\sigma(f) : A$ but f isn't receptive.

eg. S



Configurations of S = "state of play".

A

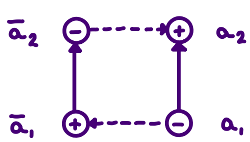
Configurations of A = "positions of the game".

Strategies between Games.

Let A and B be games. A strategy from A to B is a strategy $\sigma : A^\perp \| B$.
↳ NB. often written $A \nrightarrow B$.

Copycat.

$\mathbb{C}_A :$ $A^\perp$ A



$\mathbb{C}_A : A^\perp \| A = \{ (x, \leq_{\mathbb{C}_A} \upharpoonright x) \mid x \in C(\mathbb{C}_A) \}$

In general : see slides.

Interactions of Strategies.

Let A be a game, $\sigma : A$ a strategy and $\tau : A^\perp$ a counterstrategy.
Their interaction, $\tau \circledast \sigma = \{ p \wedge q \mid p \in \sigma \wedge q \in \tau \wedge (p \wedge q) \text{ is defined} \}$
↳ $\tau \circledast \sigma : A^0$ is a bare strategy (all moves are neutral).

In general, $\circledast : \text{Plays}(B^\perp \| C) \times \text{Plays}(A^\perp \| B) \to \text{Plays}(A^\perp \| B^0 \| C)$
where $|p| = x_{A^\perp} \| x_B$ and $|q| = y_{B^\perp} \| y_C$
↳ $q \circledast p \equiv (p \| y_C) \wedge (x_{A^\perp} \| q)$
↳ for $\sigma : A^\perp \| B$, $\tau : B^\perp \| C$,
$\tau \circledast \sigma = \{ q \circledast p \mid p \in \sigma \wedge q \in \tau \wedge q \circledast p \text{ defined} \} : A^\perp \| B^0 \| C$

Composition : $(-)\downarrow : \text{Plays}(A^\perp \| B^0 \| C) \to \text{Plays}(A^\perp \| C)$
$\tau \odot \sigma = \{ (q \circledast p)\downarrow \mid p \in \sigma \wedge q \in \tau \wedge q \circledast p \text{ defined} \} : A^\perp \| C$ (or $A \nrightarrow C$)