Denotational semantics

A denotational semantics maps each process term into an abstract model in such a way that we can work out the value of any term inductively:

$\mathcal{S}[\![P \oplus Q]\!]$ is calculated from $\mathcal{S}[\![P]\!]$ and $\mathcal{S}[\![Q]\!]$. Therefore the model must be a *congruence*:

We must be able to compute the semantic value of putting a process P in a *context* $C[\cdot]$ from the semantic value of P.

For example $(traces(P), deadlocks(P))$ is a plausible semantic model for CSP which fails to be a congruence, when $deadlocks(P)$ is the set of P's deadlock traces.

Consider $P \parallel P$ and $P' \parallel P'$, when

$$\begin{array}{rcl} P & = & (a \to P) \mathbin{\Box} (b \to P) \\ P' & = & (a \to P') \sqcap (b \to P') \end{array}$$

Fixed points

Any denotational model must provide a way of computing the semantics of recursion, since we need to be able to define $\mathcal{S}[\![\mu\, p.P]\!]$ in terms of $\mathcal{S}[\![P]\!]$ (P a term involving the process identifier p).

Necessarily $(\alpha\ =)\ \mathcal{S}[\![\mu\, p.P]\!] = \mathcal{S}[\![P[\mu\, p.P/p]]\!]$.

Hence α is a fixed point of the function represented by (the context) P.

So all CSP-definable functions over the semantic model must have fixed points, and there must be a rational basis for deciding which one to use if there are many.

Two theories are commonly used:

Tarski's theorem (in its most general form, a monotonic function on a cpo has a least fixed point).

Banach's theorem (a contraction mapping on a complete metric space has a unique fixed point).

We will use them both (see Appendix A for details of the mathematics involved).

The traces model $\mathcal{T}$

The model consists of the nonempty, prefix-closed subsets of $\Sigma^{*\checkmark} = \Sigma^* \cup \Sigma^*\langle\checkmark\rangle$.

Note that $traces(P)$, as defined as an abstraction from the operational semantics, is well-defined to $\mathcal{T}$.

$\mathcal{T}$ is a complete lattice (arbitrary subsets S have $\bigsqcup S$ and $\bigsqcap S$ existing) under the subset order $\subseteq$ because

- $\bigsqcup S = \bigcup S$ if $S \neq \{\}$
- $\bigsqcup\{\} = \{\langle\rangle\} = traces(STOP)$

Trace refinement is defined

$$P \sqsubseteq_T Q \Leftrightarrow traces(P) \supseteq traces(Q)$$

Because it is the simplest of the standard models, we will use $\mathcal{T}$ to introduce many ideas common across them all.

Operators over $\mathcal{T}$

It is elementary to compute the effect each CSP operator has on traces, for example:

$$
\begin{aligned}
&traces(P \,\Box\, Q) = traces(P \sqcap Q) = \\
&\qquad traces(P) \cup traces(Q) \\[1em]
&traces(a \to P) = \\
&\qquad \{\langle\rangle\} \cup \{\langle a \rangle\hat{}\, s \mid s \in traces(P)\} \\[1em]
&traces(P;\ Q) = \\
&\qquad (traces(P) \cap \Sigma^*) \cup \\
&\qquad \{s\hat{}\, t \mid s\hat{}\langle \checkmark \rangle \in traces(P) \wedge t \in traces(Q)\}
\end{aligned}
$$

Relational representations

All of the trace operators can be thought of as 'lifted' relations on individual traces.

For each trace results from a single trace of each of some subset of the arguments of an operator $\oplus$.

For example:

$$
\begin{array}{rcl}
s[;\ ]_1 u & \Leftrightarrow & s \in \Sigma^* \wedge u = s \\
(s,t)[;\ ]_{1,2} u & \Leftrightarrow & \exists\, s_0 . s = s_0 \hat{}\langle \checkmark \rangle \wedge u = s_0 \hat{} t
\end{array}
$$

Here, $[;\ ]_{1,2}$ shows both arguments contributing,

$[;\ ]_1$ shows only the first, and

we did not need (for ;)

$[;\ ]_2$ (only the second argument), or

$[;\ ]_\bullet$ (neither).

Relational image

For a general binary operator $\oplus$:

$$
\begin{aligned}
traces(P \oplus Q) \ = & \\
& \{u \mid \exists\, s \in traces(P), t \in traces(Q).(s, t, u) \in [\oplus]_{1,2}\} \\
& \cup \{u \mid \exists\, s \in traces(P).(s, u) \in [\oplus]_1\} \\
& \cup \{u \mid \exists\, t \in traces(Q).(t, u) \in [\oplus]_2\} \\
& \cup [\oplus]_\bullet
\end{aligned}
$$

A unary operator requires two relations, for example:

$$
\begin{aligned}
[a \rightarrow]_\bullet &= \{\langle\rangle\} \\
[a \rightarrow]_1 &= \{(s, \langle a \rangle\hat{\ }s) \mid s \in \Sigma^{*\checkmark}\}
\end{aligned}
$$

$$
\begin{aligned}
traces(\oplus(P)) &= [\oplus]_\bullet \cup \\
& \quad \{u \mid \exists\, s \in traces(P).(s, u) \in [\oplus]_1\}
\end{aligned}
$$

Benefits of relational representation

Any operator over $\mathcal{T}$ definable as the lifting of a family of trace relations is fully distributive in each of its arguments, in the sense that $F(\bigsqcap S) = \bigsqcap\{F(P) \mid P \in S\}$ for each nonempty S

Here, $F(\cdot)$ the functiona produced by fixing all arguments of $\oplus$ other than one.

Hence it is monotonic:

$$P \subseteq Q \Rightarrow F(P) \subseteq F(Q)$$

since $P \subseteq Q \Leftrightarrow P \sqcap Q = Q$
which implies $F(Q) = F(P) \sqcap F(Q)$.

and continuous with respect to $\subseteq$:

$$F(\bigcup \Delta) = \bigcup\{F(P) \mid P \in \Delta\}$$

whether $\Delta \neq \{\}$ is directed or not.

Discontinuity in $\sqsubseteq_T$

Since $\subseteq = \sqsupseteq_T$, monotonicity in these two orders is equivalent.

Continuity is not, but the relations are still the key to determining it.

Not all operators are now continuous: for example if $X = \{a_1, a_2, \ldots\}$, and $b \notin X$,

$$P_n = STOP \sqcap \bigsqcap\{a_i \to b \to STOP \mid i \geq n\}$$

are an increasing sequence under $\sqsubseteq_T$ with limit $STOP$.

However, for every n,

$$P_n \setminus X = STOP \sqcap b \to STOP$$

which is inequivalent to $STOP \setminus X$.

Discontinuity also occurs for finite hiding (see exercises).

Continuity in $\sqsubseteq_T$

An operator is always $\sqsubseteq$-continuous if each of the relations representing it is finitary: for any possible trace of $P \oplus Q$ there are only finitely many combinations of traces of P and Q it could have come from.

Except for hiding, the only infinitary relation for a standard operator is the one for renaming $P[\![R]\!]$ when the R used is infinite-to-one (i.e., it relates an infinite set of before-renaming events to a single after-renaming one).

See the book for the proof, a classic directed-set argument.

Examples

Thus $[\backslash\ \{a\}]_1$ is not finitary, for example

$$\langle\overbrace{a, a, \ldots, a}^{n}, b\rangle[\backslash\ \{a\}]_1\langle b\rangle$$

for any n, and

$[;\ ]_{1,2}$ is finitary, since a trace u of length n can only be written $s\hat{}t$ in $n+1$ ways.

The only operators other than hiding *not* to be $\sqsubseteq$-continuous over $\mathcal{T}$ are infinite-to-one renaming and infinite nondeterministic choice (the latter having infinitely many relations representing it).

Note that, in order to get the monotonicity/continuity of any interesting process term, we need the (easy) results about these results being closed under composition.

Restriction functions

$\mathcal{T}$ can be given a set of restriction functions

$$P \downarrow n = \{s \in P \mid \#s \leq n\}$$

These form a *complete restriction space*:

(a) $P \downarrow 0 = Q \downarrow 0$ for all P, Q

(b) $(P \downarrow n) \downarrow m = P \downarrow min\{n.m\}$

(c) $P \neq Q \Rightarrow \exists\, n . P \downarrow n \neq Q \downarrow n$

(d) If $P_n \downarrow m = P_m$ for all $n > m$ then there exists P^* such that $P^* \downarrow n = P_n$ for all n.

and hence a *complete metric space* under:

$$d(P, Q) = inf\{2^{-n} \mid P \downarrow n = Q \downarrow n\}$$

The restriction functions easily extend componentwise to products $\mathcal{T}^{\Lambda}$:

$$\underline{P} \downarrow n = \langle P_\lambda \downarrow n \mid \lambda \in \Lambda \rangle$$

and hence so does the complete metric.

Constructiveness and UFP

A function is said to be *constructive* if

$$P \downarrow n = Q \downarrow n \Rightarrow F(P) \downarrow n{+}1 = F(Q) \downarrow n{+}1$$

This is equivalent to it being a *contraction mapping*:

$$d(F(P), F(Q) \leq \frac{1}{2} d(P, Q)$$

It is *nondestructive* if

$$P \downarrow n = Q \downarrow n \Rightarrow F(P) \downarrow n = F(Q) \downarrow n$$

$$d(F(P), F(Q) \leq d(P, Q)$$

All basic CSP operators other than hiding are nondestructive, prefixing is constructive, and the these properties compose naturally (see Appendix A).

Hence any *guarded* recursion (without hiding) is constructive.

Any contraction mapping on a complete metric space has a *unique* fixed point (Banach fixed point theorem), giving the mathematical justification of the UFP rule.

Choice of fixed points

We now have three possible theories to provide the fixed points required by recursion:

- Metric (for constructive functions)
- Least fixed point under $\subseteq$
- Least fixed point under $\sqsubseteq$

The last two apply to *all* CSP terms, and in general produce different results.

When the first applies, which it does to all well-behaved programs, all three necessarily agree.

We therefore have to worry about the value of non-constructive recursions.

Some ill-defined recursions

The following are not constructive:

$Z_0 = Z_0$

$Z_1 = a \to (Z_1 \setminus \{a\})$

$Z_2 = (up \to (AZ /\!\!/ u{:}Z_2)) \,\Box\, (iszero \to Z_2)$, where

$$\begin{array}{rclcl} AZ & = & down & \to & (u.iszero \to AZ' \\ & & & & \Box\; u.down \to u.down \to AZ) \\ & & \Box\; up & \to & u.up \to u.up \to AZ \\ AZ' & = & iszero & \to & AZ' \\ & & \Box\; up & \to & AZ \end{array}$$

None is constructive; all have many fixed points.

For example, if P is any process never using a, then $a \to P$ is a fixed point of Z_2.

Which order for fixed points?

$\sqsubseteq$:

Pros: identifies ill-defined processes with refinement-least object.

Cons: this identification gets into trouble with composition; we lose continuity and hence the simple formula for the fixed point

$$\bigsqcup_{n=0}^{\infty} F^n(\perp)$$

$\subseteq$:

Pros: gives the operationally correct value (e.g., $\mu\, p.p$ simply diverges and thus has traces $\{\langle\rangle\}$); continuity.

Cons: discomfort at giving a "good" value to "bad" processes.

The right answer is $\subseteq$, the overriding argument being operational accuracy.

It is plainly vital the $traces(P)$ should be the same whether they are calculated via this denotational semantics, or extracted from the operational semantics.

Deficiencies of $\mathcal{T}$

The discomfort on the last slide should be directed at the model (which ignores important aspects of process behaviour) rather than the choice of fixed point.

We must not expect $\mathcal{T}$ to deal adequately with divergence (the natural consequence of ill-defined recursion) any more than it does deadlock.

Note that the process $(\mu\, p.a \ \rightarrow \ p) \ \setminus \ \{a\}$ (operationally equivalent to Z_0) is mapped to $\{\langle\rangle\}$ by out semantics, so there is no sustainable argument for not mapping Z_0 there.

It is almost inevitable that any semantic model with a top under refinement (ours being $STOP = \{\langle\rangle\}$ either is ignoring some important aspect of process behaviour or has its top unimplementable (a 'miracle').

$\mathcal{T}$ is in the first category.

Failures and divergences

$\mathcal{T}$ only allows us to see what a CSP process might communicate.

We cannot tell that it *will* communicate anything by knowing $traces(P)$, since

$$traces(P) = traces(P \sqcap STOP \sqcap \mathbf{div})$$

where the right-hand version can choose either to deadlock or diverge instead of behaving like P.

On the other hand, if the current trace is s, and

- $s \notin divergences(P)$
- $(s, X) \notin failures_\perp(P)$

then we can *guarantee* that if we offer P the set X long enough, it will accept a member.

$$(failures_\perp(P), divergences(P))$$

thus gives us a much more complete picture of P.

It has long been the "standard" model for CSP.

Healthiness conditions

Just as members of $\mathcal{T}$ are always nonempty and prefix-closed, we need a number of *healthiness* conditions to characterise which pairs (F, D) (with $F \subseteq \Sigma^{*\checkmark} \times \mathbb{P}(\Sigma^{\checkmark})$ and $D \subseteq \Sigma^{*\checkmark}$) are sensibly considered members of $\mathcal{N}$, the failures/ divergences model.

Seven axioms characterise $\mathcal{N}$.

F1. $traces_\perp(P) = \{t \mid (t, X) \in F\}$ is non-empty and prefix-closed.

F2. $(s, X) \in F \wedge Y \subseteq X \Longrightarrow (s, Y) \in F$

F3. $(s, X) \in F \wedge \forall\, a \in Y . s\hat{\ }\langle a\rangle \notin traces_\perp(P) \Longrightarrow (s, X \cup Y) \in F$

F4. $s\hat{\ }\langle\checkmark\rangle \in traces_\perp(P) \Longrightarrow (s, \Sigma) \in F$

D1. $s \in D \cap \Sigma^* \wedge t \in \Sigma^{*\checkmark} \Longrightarrow s\hat{\ }t \in D$

D2. $s \in D \Longrightarrow (s, X) \in F$

D3. $s\hat{\ }\langle\checkmark\rangle \in D \Longrightarrow s \in D$

Interpretations of axioms

- F1 As for traces model.
- F2 Whenever a process refuses a set, it would refuse a subset.
- F3 Whenever a process refuses a set, it would also refuse any events the process can't do on the given trace.
- F4 The process can opt for $\checkmark$, and so refuse to do anything else!
- D1 Divergences are extension closed.
- D2 We don't care about failures on divergence traces.
- D3 We don't care about divergence occurring after $\checkmark$.

How do we know these are right?

A set of axioms like this *must* be sound: for any node P in an LTS the value

$$(failures_{\perp}(P), divergences(P))$$

should satisfy them,

and preferably *complete*, in the sense that any (F, D) safisfying them is the image of some LTS (or, better, CSP process).

Soundness of our axioms follows easily from the way failures and divergences are extracted from LTS's.

F4,D1,D2,D3 all come directly from the definition.

F1 because if $P \stackrel{t}{\Longrightarrow} Q$ then Q must either diverge or reach a state with stable refusals.

F2 and F3 because of the way refusals arise.

Completeness is rather more challenging....we will come back to it later under the heading of *full abstraction*

Refinement order

Refinement is defined

$$(F, D) \sqsubseteq_{FD} (F', D') \equiv F \supseteq F' \wedge D \supseteq D'$$

Any immediately divergent process such as $\mathbf{div}$ is identified with the bottom element of $\mathcal{N}$:

$$\perp_{\mathcal{N}} = (\Sigma^{*\checkmark} \times \mathbb{P}(\Sigma^{\checkmark}), \Sigma^{*\checkmark})$$

This is an easy consequence of $D1$ and $D2$.

The greatest lower bound nonempty S is given by componentwise union:

$$\bigsqcap S = (\bigcup\{F \mid (F, D) \in S\}, \bigcup\{D \mid (F, D) \in S\})$$

If s is a trace of $P = (F, D)$, then P/s is

$$(\{(t, X) \mid (s\hat{\ }t, X) \in F\}, \{t \mid s\hat{\ }t \in D\})$$

Similarly:

$$\begin{array}{rcl} \mathit{initials}(P) & = & \{a \in \Sigma^{\checkmark} \mid \langle a\rangle \in \mathit{traces}_{\perp}(P)\} \\ \mathit{refusals}(P) & = & \{X \subseteq \Sigma^{\checkmark} \mid (\langle\rangle, X) \in F\} \end{array}$$

Minimal and maximal elements

Clearly $\perp_{\mathcal{N}}$ is the least (bottom) element under $\sqsubseteq_{FD}$.

$\mathcal{N}$ has no top element (evidence, we can claim, for it being a better model).

The maximal elements (ones with no proper refinements) are those which

- cannot diverge (for we can always refine a process by replacing D by $\{\}$), and
- has no other refusal sets than those implies by F3:

$$(s, X) \in F \Rightarrow X \cap \mathit{initials}((F, D)/s) = \{\}$$

These are the *deterministic* processes $\mathcal{D}$.

Thanks to F2, the above is equivalent to

$$s\hat{}\langle a\rangle \in \mathit{traces}_{\perp}(P) \Rightarrow (s, \{a\}) \notin \mathit{failures}_{\perp}(P)$$

More about the deterministic processes later!

Completeness

When Σ is finite, $\mathcal{N}$ is a cpo (directed sets have least upper bounds), and furthermore

$$\bigsqcup \Delta = (\bigcap_{(D,F)\in\Delta} F, \bigcap_{(D,F)\in\Delta} D) = (F^\dagger, D^\dagger), \text{ say}$$

All the axioms are trivial to verify for this except F3, which requires a careful directed-set argument: because Σ is finite one can show that if

$$(s,X) \in F^\dagger \quad \text{and} \quad Y \cap \mathit{initials}(F/s) = \{\}$$

contradict F3 then the problem would be the result of a finite subset of Δ and hence a single member which would itself fail F3.

See the book for details.

Thus in this case there is no problem with $\sqsubseteq$-monotonic functions having least fixed points.

Incompleteness

The proof on the last slide depends crucially on the finiteness of Σ, for the result does not hold when Σ is infinite.

This is related to the fact that a chain of nonempty *infinite* sets may have an empty intersection, unlike finite ones.

$$P_n = \bigsqcap\{a_i \to STOP \mid i \geq n\}$$

Clearly $P_n \sqsubseteq_{FD} P_{n+1}$ for all n

And $(\langle\rangle, \Sigma) \notin \mathit{failures}_\perp(P_n)$ for any n, so neither does an upper bound

But any upper bound must have no traces other than $\langle\rangle$

So no upper bound can satisfy F3.

So in this case there are monotonic functions without least fixed points, so we need to find some special property of CSP-definable ones that avoids this.

Alternative order

We can define $P \leq Q$ (the *strong* order) just when
$divergences(Q) \subseteq divergences(P)$ and
$s \in traces(P) \setminus divergences(P)$ implies
$(refusals(P/s) = refusals(Q/s)$ and
$initials(P/s) = initials(Q/s))$

The only way a process P can increase to Q is via fixing the behaviour after a trace s when it stops being a divergence.

The behaviour after s then remains fixed for any $R \geq Q$.

$P \leq Q \Rightarrow P \sqsubseteq Q$, and

$\perp_{\mathcal{N}}$ is still bottom.

Each $\leq$-directed set has $\bigsqcup \Delta$ existing (i.e., it makes $\mathcal{N}$ into a cpo) which is also its $\sqsubseteq$-lub.

The maximal processes are the divergence-free ones.

Since each CSP operator can be shown to be $\leq$-monotonic, this gives us what we need: a $\sqsubseteq$-least fixed point.

Restriction

We can define $P \downarrow n$ to be the ($\sqsubseteq$ and $\leq$) least process which behaves the same up to depth n:
$(F, D) \downarrow n = (F', D')$, where

$$\begin{array}{rcl} D' & = & D \cup \{s\hat{\ }t \mid (s, \{\}) \in F \wedge s \in \Sigma^n\} \\ F' & = & F \cup \{(s, X) \mid s \in D'\} \end{array}$$

$P \downarrow n$ behaves exactly like P until exactly n events have occurred, at which point it diverges unless it has already terminated.

These turn $\mathcal{N}$ into a complete restriction spaces, and furthermore

$$P = \bigsqcup \{P \downarrow n \mid n \in \mathbb{N}\}$$

Calculating the semantics of processes

We have to give clauses for the failures and divergences of each operator. Some of the straightforward ones:

$$failures_\perp(STOP) = \{(\langle\rangle, X) \mid X \subseteq \Sigma^\checkmark\}$$
$$divergences(STOP) = \{\}$$

$$failures_\perp(a \to P) = \{(\langle\rangle, X) \mid a \notin X\}$$
$$\cup \{(\langle a\rangle \hat{\ } s, X) \mid (s, X) \in failures_\perp(P)\}$$
$$divergences(a \to P) = \{\langle a\rangle \hat{\ } s \mid s \in divergences(P)\}$$

$$failures_\perp(P \sqcap Q) = failures_\perp(P) \cup failures_\perp(Q)$$
$$divergences(P \sqcap Q) = divergences(P) \cup divergences(Q)$$

Distinguishing $\Box$ and $\sqcap$

As we know, these two operators have identical traces.

We can tell between $(a \to P) \sqcap (b \to Q)$ and $(a \to P) \Box (b \to Q)$ using failures: the first has $(\langle\rangle, \{a\})$ and $(\langle\rangle, \{b\})$ and the second doesn't.

The following is a naïve definition of $\Box$:

$$
\begin{aligned}
&\mathit{divergences}(P \Box Q) = \\
&\qquad \mathit{divergences}(P) \cup \mathit{divergences}(Q) \\
&\mathit{failures}_\perp(P \Box Q) = \\
&\qquad \{(\langle\rangle, X) \mid (\langle\rangle, X) \in \mathit{failures}_\perp(P) \\
&\qquad\qquad\qquad \cap\, \mathit{failures}_\perp(Q)\} \\
&\qquad \cup \{(s, X) \mid (s, X) \in \mathit{failures}_\perp(P) \\
&\qquad\qquad\qquad \cup\, \mathit{failures}_\perp(Q) \wedge s \neq \langle\rangle\}
\end{aligned}
$$

Unfortunately, this may fail F4 and/or D2.

For example, it says $\mathbf{div} \Box (a \to P)$ can't refuse a even though D2 says it should.

Revised definition of $\Box$

The ‘obvious’ definition did not build in our ways of dealing with divergence and ✓. We can correct this:

$$
\begin{aligned}
& \mathit{divergences}(P \Box Q) = \\
& \qquad \mathit{divergences}(P) \cup \mathit{divergences}(Q) \\
& \mathit{failures}_\perp(P \Box Q) = \\
& \qquad \{(\langle\rangle, X) \mid (\langle\rangle, X) \in \mathit{failures}_\perp(P) \\
& \qquad\qquad\qquad\qquad\qquad\qquad \cap\, \mathit{failures}_\perp(Q)\} \\
& \qquad \cup \{(s, X) \mid (s, X) \in \mathit{failures}_\perp(P) \\
& \qquad\qquad\qquad\qquad\qquad \cup\, \mathit{failures}_\perp(Q) \wedge s \neq \langle\rangle\} \\
& \qquad \cup \{(\langle\rangle, X) \mid \langle\rangle \in \mathit{divergences}(P) \\
& \qquad\qquad\qquad\qquad\qquad \cup\, \mathit{divergences}(Q)\} \\
& \qquad \cup \{(\langle\rangle, X) \mid X \subseteq \Sigma \wedge \\
& \qquad\qquad\qquad \langle \checkmark \rangle \in \mathit{traces}(P) \cup \mathit{traces}(Q)\}
\end{aligned}
$$

Several other definitions need similar additions to preserve the divergence conditions D1 and/or D2, but this is the only place the special treatment of F4 is required.

Parallel

$$
\begin{array}{l}
\mathit{divergences}(P \underset{X}{\|} Q) = \\
\qquad \{u\hat{\ }v \mid \exists\, s \in \mathit{traces}_\perp(P), t \in \mathit{traces}_\perp(Q). \\
\qquad\quad u \in (s \underset{X}{\|} t) \cap \Sigma^* \wedge \\
\qquad\quad (s \in \mathit{divergences}(P) \vee t \in \mathit{divergences}(Q))\}
\end{array}
$$

$$
\begin{array}{l}
\mathit{failures}_\perp(P \underset{X}{\|} Q) = \\
\qquad \{(u, Y \cup Z) \mid\ Y \setminus (X \cup \{\checkmark\}) = Z \setminus (X \cup \{\checkmark\}) \\
\qquad \wedge\ \exists\, s, t.(s, Y) \in \mathit{failures}_\perp(P) \\
\qquad\quad \wedge\ (t, Z) \in \mathit{failures}_\perp(Q) \\
\qquad\quad \wedge\ u \in s \underset{X}{\|} t\} \\
\qquad \cup \{(u, Y) \mid u \in \mathit{divergences}(P \underset{X}{\|} Q)\}
\end{array}
$$

Other parallel operators (${}_X\|_Y$, $|||$) are definable in terms of this one.

Hiding

$P \setminus X$ can only refuse a set of events when P could have refused X too: otherwise it would not have been stable.

$$\begin{array}{l} \mathit{failures}_\perp(P \setminus X) = \\ \qquad \{(s \setminus X, Y) \mid (s, Y \cup X) \in \mathit{failures}_\perp(P)\} \\ \qquad \cup\{(s, X) \mid s \in \mathit{divergences}(P \setminus X)\} \end{array}$$

Everything we have seen so far can be done via relational representations (as discussed for $\mathcal{T}$) in that each result behaviour comes from at most one from each argument.

This breaks down, however, for $\mathit{divergences}(P \setminus X)$.

$P \setminus X$ can diverge when P performs an infinite sequence of events in X, and all we can do is to infer this from an infinity of finite traces:

$$\begin{array}{l} \mathit{divergences}(P \setminus X) = \\ \qquad \{(s \setminus X)\hat{\ }t \mid s \in \mathit{divergences}(P)\} \\ \qquad \cup\{(u \setminus X)\hat{\ }t \mid u \in \Sigma^\infty \wedge (u \setminus X) \,\text{finite} \\ \qquad\quad \wedge \forall\, s < u . s \in \mathit{traces}_\perp(P)\} \end{array}$$

Need for finite nondeterminism

If $\mathit{Traces}(P)$ is not closed, this may well be inaccurate:

$$\begin{array}{rcl} A^* & = & \bigsqcap\{A_n \mid n \in \mathbb{N}\} \quad \text{and} \\ A^\infty & = & AS \sqcap \bigsqcap\{A_n \mid n \in \mathbb{N}\} \end{array}$$

where $A_0 = STOP$, $A_{n+1} = a \rightarrow A_n$ and $AS = a \rightarrow AS$. Identical in $\mathcal{N}$, but only one of them can actually perform an infinite sequence of a's and hence diverge when we hide $\{a\}$.

Thus $\mathcal{N}$ fails to be a *congruence* for the full language. You must restrict yourself to finitely nondeterministic CSP to use $\mathcal{N}$ properly.

Fortunately, there is a solution to this problem, namely adding infinite traces. See later!

Two more operators

$$
\begin{aligned}
& \mathit{divergences}(P[\![R]\!]) = \\
& \qquad \{s'\hat{\ }t \mid \exists\, s \in \mathit{divergences}(P) \cap \Sigma^*. s\; R\; s'\}
\end{aligned}
$$

$$
\begin{aligned}
& \mathit{failures}_\perp(P[\![R]\!]) = \\
& \qquad \{(s', X) \mid \exists\, s. s\; R\; s' \wedge (s, R^{-1}(X)) \in \mathit{failures}_\perp(P)\} \\
& \qquad \cup \{(s, X) \mid s \in \mathit{divergences}(P[\![R]\!])\}
\end{aligned}
$$

Here, $R^{-1}(X) = \{a \mid \exists\, a' \in X.(a, a') \in X\}$

In sequential composition, the first ✓ is treated like a hidden event:

$$
\begin{aligned}
& \mathit{divergences}(P;\; Q) = \\
& \qquad \mathit{divergences}(P) \cup \\
& \qquad \{s\hat{\ }t \mid s\hat{\ }\langle \checkmark \rangle \in \mathit{traces}_\perp(P) \wedge t \in \mathit{divergences}(Q)\}
\end{aligned}
$$

$$
\begin{aligned}
& \mathit{failures}_\perp(P;\; Q) = \\
& \qquad \{(s, X) \mid s \in \Sigma^* \wedge (s, X \cup \{\checkmark\}) \in \mathit{failures}_\perp(P)\} \\
& \qquad \cup \{(s\hat{\ }t, X) \mid s\hat{\ }\langle \checkmark \rangle \in \mathit{traces}_\perp(P) \\
& \qquad\qquad \wedge (t, X) \in \mathit{failures}_\perp(Q)\} \\
& \qquad \cup \{(s, X) \mid s \in \mathit{divergences}(P;\; Q)\}
\end{aligned}
$$

Relational representations

As over $\mathcal{T}$, the basic properties of these operators are largely deducible from the existence of relational representations.

Actually defining the relations is rather dull, since we now have to worry about the interplay between two different types of behaviour as well as the combinations of different arguments.

For example, consider renaming $P[\![R]\!]$, where we get a relation from failures to failures

$$\{((s, R^{-1}(X)), (s', X)) \mid s \, R \, s'\}$$

one from divergences to divergences

$$\{(s, s'\hat{\ }t) \mid s \in \Sigma^* \wedge s \, R \, s'\}$$

and one from divergences to failures

$$\{(s, (s'\hat{\ }t, X)) \mid s \, R \, s' \wedge (s \in \Sigma^* \vee t = \langle\rangle)\}$$

These exist for every operator except hiding, since instead of a behaviour of $P \setminus X$ coming from a single one of P, sometimes it takes infinitely many.

Distributivity

Thus, each of the operators other than $\setminus X$ is fully distributive (and hence monotonic) over $\mathcal{N}$ in every argument.

$\setminus X$ is not fully distributive (for example,

$$(\bigsqcap\{A_n \mid n \in \mathbb{N}\} \setminus a \neq \bigsqcap\{A_n \mid n \in \mathbb{N}\}$$

but does satisfy $(P \sqcap Q) \setminus X = (P \setminus X) \sqcap (Q \setminus X)$ (which also implies monotonicity).

This is because if $P \sqcap Q$ has all finite prefixes of some infinite trace then one of P and Q must too, by F1.

Full (infinite) distributivity is a dubious concept in a model that can't handle infinite nondeterminism properly. The problem here with hiding is just another symptom of that.

Recursion

We again need to decide how to extract fixed points.

The status of the metric theory (of constructive recursions) is exactly the same as over $\mathcal{T}$, namely that it works for almost all well-behaved ones, and when it does says that there was not really a choice at all.

There is still the question of how to handle under-defined recursions.

This time, we only have one choice, namely least fixed points with $\sqsubseteq$. This is odd, since we chose the opposite order for $\mathcal{T}$.

It is bound to give us a different set of traces ($\Sigma^{*\checkmark}$) for $\mu\, p.p$ than we got in $\mathcal{T}$.

That is OK since it simply reflects the difference between $traces(P)$ and $traces_\perp(P)$.

An ill-defined recursion is one where the unwinding of the recursion creates a divergence, which means that the right value for it (after the given trace) is $\perp_\mathcal{N}$. This is entirely consistent with choosing the $\sqsubseteq$-least fixed point.

Positive and negative reasoning

The $\sqsubseteq$-least fixed point contains all those behaviours that cannot be proved to be absent from any fixed point.

We can characterise this as *negative* reasoning.

A $\subseteq$-least fixed point (where possible) contains all those behaviours for which there is a direct, well-founded proof that they must be in *every* fixed point.

This is *positive* reasoning.

Positive reasoning is correct if all behaviours are finite with finite proofs (e.g., finite traces in $\mathcal{T}$), but cannot prove the presence of infinite behaviours like divergences. Consider $\mu\, p.p$: no proofs of anything!

One of the main roles of D1 and D2 is ensuring that negative reasoning for fixed points works:

It is likely to allow many other behaviours into the representation of a process than the "real" (non-extension-closed) divergences. But these, fortunately, only extend divergences, and so are things D1 and D2 say we want.

Continuity

Analysing continuity is greatly simplified when Σ is finite, since this means that there are only finitely many failures with any given trace.

Thus any relational representation of an operator that is finitary on traces is *automatically* finitary on failures.

Other than $\setminus X$, this essentially means that all finitely nondeterministic operators inherit their $\sqsubseteq$-continuity from $\mathcal{T}$.

However, unlike over $\mathcal{T}$, finite hiding is now continuous (using a quite separate König's Lemma argument).

When discontinuity arises over $\mathcal{T}$, it gets hidden behind divergence in $\mathcal{N}$.

Continuity is, in any case, a dubious question over infinite Σ with $\sqsubseteq$ since $\mathcal{N}$ is not a cpo.

Cutting down $\mathcal{N}$

After a trace a process must either diverge or become stable.

We can cut produce "cut-down" versions which only record one or other of these possibilities, but must appreciate that neither is, by itself, strong enough to address liveness specifications.

The combination of divergences and finite traces works as a congruence (with D1 and modified D2, and $\sqsubseteq$-fixed points), and has a theory in most ways like $\mathcal{N}$.

More interesting is the model in which we record only traces and 'stable refusals':

$$(traces(P), failures(P))$$

This gives the *stable failures* model $\mathcal{F}$.

Summary of uses of $\mathcal{F}$

- We get the ability to see much of the detail of $\mathcal{N}$ past potential divergence *by ignoring divergence completely*
- It allows some fascinating specifications of processes P formed by putting it in a context $C[P]$ that might introduce divergence. Examples later.
- It gives a formal justification to the more efficient "failures" check of FDR.

Healthiness conditions

It is by no means inevitable that all traces have stable failures.

$$\mathbf{div} \; \Box \; a \rightarrow a \rightarrow \mathbf{div}$$

only has stable failures after $\langle a \rangle$ (neither prefix-closed nor having all traces as prefixes).

Thus we need to record traces explicitly: a pair (T, F) is in $\mathcal{F}$ iff

T1 T is nonempty and prefix-closed.

T2 $(s, X) \in F \Rightarrow s \in T$

plus conditions F2, F3, and F4 (with T replacing $traces_\perp(P)$)

The following condition replaces D3: it standardises post-✓ behaviour.

T3 $s\hat{\ }\langle \checkmark \rangle \in T \Rightarrow (s\hat{\ }\langle \checkmark \rangle, X) \in F$

The soundness and completeness issues arise as before.

Order properties

Any nonempty subset of $\mathcal{F}$ has a glb under $\sqsubseteq_F$:

$$\bigsqcap S = (\bigcup_{(T,F)\in S} T, \bigcup_{(T,F)\in S} F)$$

Also $\top_{\mathcal{F}} = \mathbf{div} = (\{\langle\rangle\}, \{\})$ is the glb of $\{\}$

Hence, $\mathcal{F}$ is a complete lattice. (Note that this holds whether Σ is finite or not.)

Thus any set of processes has a least common refinement, but these are not always particularly intuitive. That of

$$\{a \to a \to STOP, a \to b \to STOP\}$$

is $a \to \top_{\mathcal{F}} = (\{\langle\rangle, \langle a\rangle\}, \{(\langle\rangle, X) \mid a \notin X\})$
the inconsistent behaviour after $\langle a\rangle$ being resolved to $\top_{\mathcal{F}}$.

This is further evidence that $\mathcal{F}$ does not give a complete picture of processes!

Calculating semantics

Since a process is represented by

$$(traces(P), failures(P))$$

it follows that the first component of the $\mathcal{F}$ semantics is identical to the $\mathcal{T}$ semantics.

And also that we should use a positive ($\subseteq$) least fixed point for recursion. That is consistent both with the fact that $\mu\, p.p = \mathbf{div}$ is identified with $\top_{\mathcal{F}}$ (the $\perp$ under $\subseteq$) and the fact that all its behaviours are finitary.

The clauses for calculating the failures component are essentially the same as those over $\mathcal{N}$ except that there is no need to close under axioms D1 and D2.

All operators have relational representations, and hence are fully distributive, and continuous with respect to $\subseteq$.

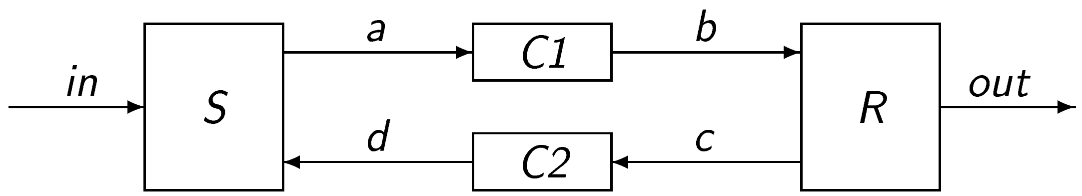
Observations

- You must always remember that $\mathcal{F}$ tells you *nothing* about whether a process can diverge; however
- If a process is known to be divergence-free, its $\mathcal{F}$ representation is effectively the same as its $\mathcal{N}$ one.
- There is now no restriction to finitely nondeterministic CSP: the awkward cases that arose over $\mathcal{N}$ are avoided.
- $\mathcal{F}$ represents a proper refinement of $\mathcal{T}$, but it is neither a refinement nor an anti-refinement of $\mathcal{N}$. It lets us see detail past divergence.

Stepwise refinement

Where it is difficult to eliminate divergence, we may choose to develop a system in $\mathcal{F}$ before finally making a refinement which eliminates divergence, bringing the two models together:

For example, alternating bit protocol



diverges with arbitrary loss/duplication

$$
\begin{array}{rcl}
C(in, out) & = & in?x \rightarrow C'(in, out, x) \\
C'(in, out, x) & = & out!x \rightarrow C'(in, out, x) \\
 & & \sqcap\ in?y \rightarrow C'(in, out, y)
\end{array}
$$

but refines $COPY$ in $\mathcal{F}$. Therefore any refinement that is divergence-free refines $COPY$ in $\mathcal{N}$.

Deadlock checking

It can be more elegant and/or efficient to analyse a divergence-free process by putting it into a context where it might diverge.

The most general case of this is *abstraction*: see later.

An easy and powerful case is deadlock checking: P is deadlock-free iff

$$SKIP \sqsubseteq_F P \setminus \Sigma$$

This is the key to efficient checking on FDR, since it enables effective use of compression functions.