The need for infinite traces

We have seen:

- that the failures/divergences model $\mathcal{N}$ is the only one seen so far that gives a reasonably complete picture of processes, but
- that it breaks down (by failing to be a congurence) for unboundedly nondeterministic CSP.

The *way* it broke down was that we were no longer able to *infer* what infinite traces a process has from the finite ones. This information was needed in order to calculate which divergences are introduced via hiding.

The solution to this problem is to add a component to the model of a process: it becomes

$$(\mathit{failures}_\perp(P), \mathit{divergences}(P), \mathit{infinites}(P))$$

where $\mathit{infinites}(P)$ are the (divergence-closed) infinite traces of P.

Always bear in mind that the extra component will only convey extra information about unboundedly nondeterministic processes (for which we cannot guarantee that $\mathit{Traces}(P)$ is closed).

Factoring the problem

Ultimately we want to axiomatise which triples (F, D, I) are allowed and to analyse the resulting model. But this turns out to be relatively tricky......

So it is interesting to discover that you don't need all these details to calculate

$$(\mathit{failures}_\perp(P), \mathit{divergences}(P), \mathit{infinites}(P))$$

via denotational semantics: you can factor the problem into two.

The factorisation

This is because:

- The stable failures $failures(P)$ are accurately calculated, even for infinitely nondeterministic CSP, by the semantics over the stable failures model $\mathcal{F}$.
- Stable failures have no influence (in the semantics of any CSP operator) on finite or infinite traces, or divergences.

We can therefore calculate the divergences and infinite traces via the simpler model

$$(traces_{\perp}(P), divergences(P), infinites(P))$$

and combine this information with $failures(P)$ from $\mathcal{F}$ to build $failures_{\perp}(P)$.

The infinite traces/divergences model

This model, written $\mathcal{I}$, consists of the triples (T, D, I), where $T, D \subseteq \Sigma^{*\checkmark}$ and $I \subseteq \Sigma^{\omega}$, meeting the following conditions:

T1. T is nonempty and prefix closed.

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

D2′. $D \subseteq T$

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

D4. $s \in D \cap \Sigma^* \wedge u \in \Sigma^{\omega} \Rightarrow s\hat{}u \in I$

I1′. $u \in I \wedge s < u \Rightarrow s \in T$

Partial order properties of $\mathcal{I}$

$\mathcal{I}$ is a complete lattice under both $\sqsubseteq_{\mathcal{I}}$ (reverse inclusion on all three components) and the subset order.

$\top_{\mathcal{I}} = (\{\langle\rangle\}, \{\}, \{\})$ (the representation of $STOP$) is the greatest $\sqsubseteq_{\mathcal{I}}$-lower bound of the empty set, and the greatest lower bound of any nonempty set is given, as usual, by componentwise union: $\bigsqcap S = (T^*, D^*, I^*)$, where

$$\begin{array}{rcl} T^* & = & \bigcup\{T \mid (T, D, I) \in S\} \\ D^* & = & \bigcup\{D \mid (T, D, I) \in S\} \\ I^* & = & \bigcup\{I \mid (T, D, I) \in S\} \end{array}$$

The least refined element is $\perp_{\mathcal{I}} = (\Sigma^{*\checkmark}, \Sigma^{*\checkmark}, \Sigma^{\omega})$, representing any process that can diverge immediately.

Calculating $\mathit{infinites}(P)$

It is easy to work this out for each CSP operator: for example

$$
\begin{array}{rcl}
\mathit{infinites}(STOP) & = & \{\} \\
\mathit{infinites}(a \to P) & = & \{\langle a\rangle\hat{\ }u \mid u \in \mathit{infinites}(P)\} \\
\mathit{infinites}(P \mathrel{\Box} Q) & = & \mathit{infinites}(P) \cup \mathit{infinites}(Q) \\
\mathit{infinites}(P \mathbin{\underset{X}{\|}} Q) & = & \{u \in \Sigma^\omega \mid \exists\, s \in \mathit{Traces}(P), \\
 & & \exists\, t \in \mathit{Traces}(Q).u \in s \mathbin{\underset{X}{\|}} t\} \cup \\
 & & \{s\hat{\ }u \mid s \in \mathit{divergences}(P \mathbin{\underset{X}{\|}} Q) \cap \Sigma^* \\
 & & \qquad \wedge\, u \in \Sigma^\omega\} \\
\mathit{infinites}(P[\![R]\!]) & = & \{u' \mid \exists\, u \in \mathit{infinites}(P).u\; R\; u'\} \cup \\
 & & \{s\hat{\ }u \mid s \in \mathit{divergences}(P[\![R]\!]) \cap \Sigma^* \\
 & & \qquad \wedge\, u \in \Sigma^\omega\} \\
\mathit{infinites}(P \setminus X) & = & \{u' \in \Sigma^\omega \mid \exists\, u \in \mathit{infinites}(P). \\
 & & \qquad u \setminus X = u'\} \\
 & & \cup\{s\hat{\ }u \mid s \in \mathit{divergences}(P \setminus X) \cap \Sigma^* \\
 & & \qquad \wedge\, u \in \Sigma^\omega\}
\end{array}
$$

The other components

For a denotational semantics over $\mathcal{I}$ we also need clauses for $traces_\bot(P)$ and $divergences(P)$.

In all but one case, the clauses for $divergences(P)$ are the same as over $\mathcal{N}$.

The other case is the interesting one:

$$\begin{array}{rcl} divergences(P \setminus X) & = & \{(u \setminus X)\hat{\ }t \mid u \in infinites(P) \land \\ & & \quad t \in \Sigma^{*\checkmark} \land u \setminus X \text{ is finite}\} \\ & & \cup\{(s \setminus X)\hat{\ }t \mid s \in divergences(P) \\ & & \quad \cap \Sigma^* \land t \in \Sigma^{*\checkmark}\} \end{array}$$

This is the only place where infinite traces directly affect other behaviours.

The trace-clauses are the same as the effects the clauses from $\mathcal{N}$ have on trace. These differ from the clauses for the finite traces model $\mathcal{T}$ where closure under $D1/D2$ is needed. For example:

$$\begin{array}{rcl} traces_\bot(P \setminus X) & = & \{s \setminus X \mid s \in traces_\bot(P)\} \\ & & \cup\{s\hat{\ }t \mid s \in divergences(P \setminus X) \cap \Sigma^* \\ & & \quad \land t \in \Sigma^{*\checkmark}\} \end{array}$$

Remarks

All the resulting semantic clauses are

- universally accurate
- with relational representations (often infinitary)

Thus all standard operators are fully distributive (over both $\mathcal{I}$ and $\mathcal{F}$) but many are discontinuous over $(\mathcal{I}, \sqsubseteq)$.
Notice how

$$A^* \quad \text{and} \quad A^* \sqcap AS$$

and hence

$$A^* \setminus a \quad \text{and} \quad (A^* \sqcap AS) \setminus a$$

are distinguished appropriately.

Recursion and fixed points in $\mathcal{I}$

As this model is a complete lattice we have a choice of either order for fixed points.

We have established a rule of thumb that infinitary behaviours in a model implies that $\sqsubseteq$ should be used.

The choice is more important, since it turns out that even well-behaved recursions have multiple fixed points.

$\mu\, p.a \to p$ has two: the $\subseteq$-least

$$(\{\langle a\rangle^n \mid n \in \mathbb{N}\}, \{\}, \{\})$$

and the $\sqsubseteq$-least (and correct) one

$$(\{\langle a\rangle^n \mid n \in \mathbb{N}\}, \{\}, \{\langle a\rangle^\omega\})$$

In slightly more complex cases such as

$$\mu\, p.(a \to p) \sqcap (b \to p)$$

there can be infinitely many fixed points. It is clear that the correct answer is again the $\sqsubseteq$-least one which contains all infinite traces made up of a's and b's.

Convergence beyond ω

In the presence of unbounded nondeterminism, recursions have frequently not reached their fixed points by

$$F^{\omega}(\perp_{\mathcal{I}}) = \bigsqcup_{i=0}^{\infty} F^{n}(\perp_{\mathcal{I}})$$

For example consider

$$P = (a \rightarrow P) \underset{\{a\}}{\parallel} A^{*}$$

where $F^{\omega}(\perp_{\mathcal{I}})$ is

$$(\{\langle a \rangle^{n} \mid n \in \mathbb{N}\}, \{\}, \{\langle a \rangle^{\omega}\})$$

This is not a fixed point, since applying F once more gives

$$(\{\langle a \rangle^{n} \mid n \in \mathbb{N}\}, \{\}, \{\})$$

which is the least fixed point.

Congruence theorem

With $\sqsubseteq$ fixed points, one can prove the natural congruence theorem between $\mathcal{I}$ and the operational semantics.

The lemmas for the ordinary operators are much the same as over earlier models,

and the proof that the image of operational fixed point *is* a fixed point is identical.

The proof that it is always the *least* fixed point is much harder, because we lose the finitary arguments that König's Lemma and continuity make possible in simpler models. It seems to require sophisticated and complex transfinite (ordinal constructions).

The important thing, however, is *that it is true*.

The demise of metrics

By adding infinite traces we have been able to give a complete picture of all processes, even unboundedly nondeterministic ones.

However we lose one of our main tools in proving things about processes:

- processes are no longer determined by all their finite restrictions (as infinite traces are never reached);
- therefore we do not get a metric on $\mathcal{I}$ as we did no earlier models;
- as we have seen already, constructive functions need not have unique fixed points.

Incorporating failures in the model

While we can now compute

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

via a pair of semantics, it is interesting to study which triples (F, D, I) represent processes. We will then be able to address subjects like full abstraction.

We want to build the failures/divergences/infinite traces model $\mathcal{U}$ consisting of these sets of triples (F, D, I).

The only problem in constructing an appropriate set of axioms is understanding the relationships between refusal sets and infinite traces. All the other relationships have been brought out in the axioms of $\mathcal{I}$ and $\mathcal{F}$.

Forcing infinite traces

There is such a relationship because sometimes you can tell that an infinite trace must be forcible by looking at its refusals. Consider the failure sets

$$\begin{array}{llll} F_a & = \mathit{failures}_\perp(R_a) & = & \{(\langle a\rangle^n, X) \mid a \notin X\} \\ F_{ab} & = \mathit{failures}_\perp(R_{ab}) & = & \{(s, X) \mid s \in \{a,b\}^* \wedge \{a,b\} \not\subseteq X\} \end{array}$$

If you offer either of them Σ for ever, an infinite trace is certain to occur. Therefore neither is consistent with I being empty ....

With F_a, this infinite trace is certain to be $\langle a, a, a \ldots\rangle$.

So $\mu\, p.a \rightarrow p$ actually only has one fixed point over $\mathcal{U}$.

With F_{ab} it could be any from $\{a,b\}^\omega$ (though there must be an extension of every finite trace).

$\mu\, p.(a \rightarrow p) \sqcap (b \rightarrow p)$ has infinitely (uncountably) many fixed points over $\mathcal{U}$.

Deterministic processes over $\mathcal{U}$

If a process has a deterministic failure set:

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

then you can force it to perform any of its traces by simply offering it the events one at a time.

The same goes for any infinite trace in $\overline{\mathit{traces}(P)}$: we can conclude that

$$\mathit{Traces}(P) = \overline{\mathit{traces}(P)}$$

We can thus tell if a process is deterministic by looking at its failures, and it must then have a *closed* set of traces.

This is hardly surprising, since we know that a finitely nondeterministic LTS gives a closed $\mathit{Traces}(P)$.

Philosophy of nondeterminism

Imagine that you are about to experiment on an unboundedly nondeterministic process P.

You can't tell whether the nondeterminism is decided during your experiment or has all been pre-resolved before you start.

In the latter case you are certainly experimenting on a closed process which refines P: but you don't know which one!

It follows that

$$P = \bigsqcap\{P' \mid P \sqsubseteq P' \wedge P' \text{ is closed}\}$$

Since whenever $(F, D) \in \mathcal{N}$ we must have

$$(F, D, \overline{traces(F)} \setminus traces(F)) \in \mathcal{U}$$

the above identity completely determines the members of $\mathcal{U}$.

Axioms of $\mathcal{U}$

We can thus define $\mathcal{U}$ to consist of those triples (F, D, I) satisfying F1–F4, D1–D4, the following

I1 $u \in I \wedge s < u$ (for $s \in \Sigma^*$) implies $s \in traces(F)$

and one more property, I2 capturing the relationship between refusals and I.

This is necessarily in a different style, since it must relate infinite sets of behaviours rather than small finite sets. It can be formulated in a variety of mathematical languages: e.g., infinitary logic and game theory.

The following is perhaps the simplest, and is nearly just a restatement of the principle on the previous slide:

I2 $$s \in traces(F) \;\Rightarrow\; \begin{array}{l} \exists\, T \in \mathcal{T}^d.(F, D)/s \sqsubseteq_{FD} det(T) \\ \wedge \{s\hat{\ }u \mid u \in \overline{T}\} \subseteq I \end{array}$$

Example

With the failure-set $\Sigma^{*\checkmark} \times \mathbb{P}(\Sigma^{\checkmark})$ and divergence-set $\{\}$, any set of infinite traces is permissible. In particular, you can have the empty set of infinite traces, creating the interesting process $FINITE_{\Sigma}$ which can be written

$$\bigsqcap \{\, T_s \mid s \in \Sigma^{*\checkmark} \}$$

with T_s, as before, being the process which simply communicates the trace s.

This has the same failures and divergences as $Chaos$, but the opposite set of infinite traces.

You should always remember that if P is a finitely nondeterministic term (or, indeed, a node in a finitely nondeterministic LTS) then its value in $\mathcal{U}$ is always simply the closure of its value in $\mathcal{N}$.

Full abstraction and $\mathcal{U}$

Using $\sqcap$ (now safe!) the construction used to show every element of $\mathcal{N}$ for finite Σ now works for arbitrary Σ.

It always produces a closed process: thus all closed processes are expressible.

Since every process is the nondeterministic composition of closed processes, it follows that all are expressible.

That $\mathcal{U}$ is a natural extension of $\mathcal{N}$ for unboundedly nondeterministic CSP is emphasised by the following:

- The $\mathcal{U}$ semantics for CSP is fully abstract with respect to $\mathbf{T}_3$ (immediate deadlock or immediate divergence).

This is the same result as $\mathcal{N}$, only applying to the full language.

The key to this result is finding a context $C_u[\cdot]$ that makes P fail $\mathbf{T}_3$ exactly when $u \in \mathit{infinites}(P)$:

$$(P \underset{\Sigma}{\|} T_u) \setminus \Sigma$$

Partial order properties of $\mathcal{U}$

The refinement order on $\mathcal{U}$ is exactly what we would expect:

$$(F, D, I) \sqsubseteq_{\mathcal{U}} (F', D', I') \equiv F \supseteq F' \wedge D \supseteq D' \wedge I \supseteq I'$$

The bottom is any immediately divergent process:

$$\bot_{\mathcal{U}} = (\Sigma^{*\checkmark} \times \mathbb{P}(\Sigma^{\checkmark}), \Sigma^{*\checkmark}, \Sigma^{\omega})$$

Its maximal elements are, like $\mathcal{N}$, the deterministic processes: $\{\overline{det(T)} \mid T \in \mathcal{T}^d\}$

The greatest lower bound of any nonempty set is computed by nondeterministic choice (componentwise union).

However, it is not complete, even for finite Σ:

$$A^* \sqsubseteq_{\mathcal{U}} a \to A^* \sqsubseteq_{\mathcal{U}} a \to a \to A^* \sqsubseteq_{\mathcal{U}} \ldots$$

The failures of an upper bound would be a subset of

$$F_a = \{(\langle a \rangle^n, X) \mid n \in \mathbb{N} \wedge a \notin X\}$$

F3 implies they must equal this set, and I2 implies that $\langle a, a, a \ldots \rangle \in I$. This is impossible since the limit refines A^*.

Other orders?

Over $\mathcal{N}$ with infinite Σ we got round incompleteness with a stronger order.

You can still define $\leq$:

$$(F, D, I) \leq_{\mathcal{U}} (F', D', I') \equiv (F, D) \leq_{\mathcal{N}} (F', D') \wedge I \supseteq I'$$

CSP operators are monotone with respect to this order, but it is not complete.

There are sequences $I_1 \supset I_2 \supset \ldots$ such that

$$(F_{ab}, \{\}, I_n) \in \mathcal{U}, \quad \text{but} \quad \bigcap_{n=1}^{\infty} I_n = \{\}$$

One can construct a sequence of processes that are ordered in any partial order on $\mathcal{U}$ which could give CSP the right values for recursions, but which can have no upper bound.

Thus the incompleteness is *fundamental*.

Should we worry about whether recursions have least fixed points?

Fixed points in $\mathcal{U}$

We know we can calculate the $\mathcal{U}$ semantics of any CSP process via $\mathcal{F}$ and $\mathcal{I}$.

$$(T, F), (T', D, I) \mapsto (F \cup \{(s, X) \mid s \in D\}, D, I)$$

But how do we know the result *is in* $\mathcal{U}$?

How do we know we haven't built objects that fail (e.g.) I2?

Given what we already know, the simple way is via the congruence with operational semantics: we know that the representations in $\mathcal{F}$ and $\mathcal{I}$ of a process P are the correct observations of the same LTS. The combination of these values is then the observation of the same LTS in $\mathcal{U}$, and therefore satisfies the axioms.

One can readily show that the congruence theorems for $\mathcal{I}$ and $\mathcal{F}$ imply the corresponding result for $\mathcal{U}$.

But it would be nice if we could prove the existence of fixed points without all this machinery......

Barrett's proof

A process $(F, D, I) \in \mathcal{U}$ is *pre-deterministic* if it behaves deterministically whenever it does not diverge:

$$s \in traces(F) \setminus D \wedge s\hat{\ }\langle a\rangle \in traces(F) \Rightarrow (s, \{a\}) \notin F$$

The set $\mathcal{P}$ of all predeterministic processes is a cpo under $\sqsubseteq$ and $\leq$ (the orders coincide on $\mathcal{P}$), and every member of $\mathcal{U}$ has one of them above it.

In any partial order (e.g. $\mathcal{U}$) in which nonempty sets have greatest lower bounds, any monotonic function with a *post fixed point* ($f(x) \leq x$) has a least fixed point

$$\bigsqcap\{x \mid f(x) \leq x\}$$

(see the proof of Tarski's theorem).

So it will suffice to show that each CSP definable f has a post-fixed point.

Barrett's proof II

For each CSP definable function f, there is a function f' from $\mathcal{P}$ *to itself* which

- is monotone
- $f(P) \sqsubseteq_{\mathcal{U}} f'(P)$ for all $P \in \mathcal{P}$.

It follows that f' has a fixed point in $\mathcal{P}$, such that

$$f(\mu f') \sqsubseteq_{\mathcal{U}} f'(\mu f') = \mu f'$$

This is the post-fixed point we wanted.

f' is constructed by replacing each operator $\oplus$ in the defintion of f by a version $\oplus'$ which resolves all the nondeterminism the original can create. This has to be done systematically to keep monotonicity.

For example, we might define

$$P \sqcap' Q = P$$

Using infinite traces

It is debatable whether a real unboundedly nondeterminisitic process can be built.

However there can be no doubt that unbounded nondeterminism is useful in formulating specifications: whenever we want to express the idea that there is some finite bound on a behaviour without saying what the bound is:

For example, a finite buffer: $divergences(B) = \{\}$ and

(i) $s \in traces(B) \Rightarrow s \in \{| \, left, right \, |\}^* \wedge$
$s \downarrow right \leq s \downarrow left$

(ii) $(s, X) \in failures(B) \wedge s \downarrow right = s \downarrow left$
$\Rightarrow X \cap \{| \, left \, |\} = \{\}$

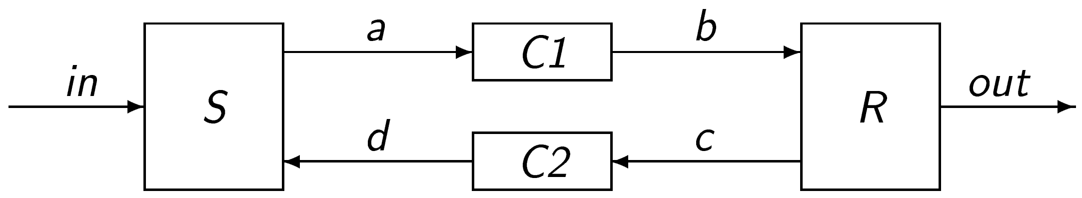
(iii) $(s, X) \in failures(B) \wedge s \downarrow right < s \downarrow left$
$\Rightarrow \{| \, right \, |\} \not\subseteq X$

(iv) $u \in infinites(B) \Rightarrow u \setminus \{| \, left \, |\}$ is infinite

Just as with other specifications, it can be useful to insert this sort as a component of a process:

If $C[FinBUFF_{\langle\rangle}] \sqsupseteq Spec$, where $FinBUFF_{\langle\rangle}$ is the characteristic process of the above, then $C[B] \sqsupseteq Spec$ for all B satisfying it.

Alternating bit protocol



Over $\mathcal{F}$ we showed how to argue that this protocol was correct except possibly for diverging.

Infinite traces allow us to make the most general assumptions for excluding divergence: we replace $C1$ and $C2$ by the most general refinements that never commit infinitely many successive errors.

In other words, neither performs infinitely many inputs without an output (which would be an infinite sequence of losses) or infinitely many outputs without an input (which would be an infinite sequence of duplications).

This is a *fairness* assumption.........

Fair arbiter

The most general fair arbiter between A and B is

$$\begin{array}{rcl}
\mathit{Fair}(A,B) & = & \mathit{LFair}(A,B) \sqcap \mathit{RFair}(A,B), \quad \text{where} \\
\mathit{LFair}(A,B) & = & \bigsqcap\{\mathit{NLeft}(n,A,B) \mid n \in \{1,2,\ldots\}\} \\
\mathit{RFair}(A,B) & = & \bigsqcap\{\mathit{NRight}(n,A,B) \mid n \in \{1,2,\ldots\}\} \\
\mathit{NLeft}(n,A,B) & = & \mathit{RFair}(A,B) \\
 & & \lessdot n = 0 \gtrdot ?x : A \rightarrow \mathit{NLeft}(n-1,A,B) \\
\mathit{NRight}(n,A,B) & = & \mathit{LFair}(A,B) \\
 & & \lessdot n = 0 \gtrdot ?x : B \rightarrow \mathit{NRight}(n-1,A,B)
\end{array}$$

After any finite trace of a's and b's this can choose either A or B, but any infinite trace must have infinitely many of each:

$$A^*(B\,B^*\,A\,A^*)^\omega$$

You can make a process fair by using one of these to replace an existing unfair choice mechanism (usually $\sqcap$).

Using a fair arbiter

$C_*(in, out)$ replaces the nondeterministic choices of $C(in, out)$ by external choices. This makes the choice mechanism available to the fair arbiter:

$$
\begin{aligned}
C_*(in, out) &= in?x \rightarrow C'_*(in, out, x) \\
C'_*(in, out, x) &= out!x \rightarrow C'_*(in, out, x) \\
&\quad \Box\ in?y \rightarrow C'_*(in, out, y) \\
C_{\mathit{fin}}(in, out) &= C_*(in, out) \underset{\Sigma}{\parallel} \mathit{LFair}(\{\mid in \mid\}, \{\mid out \mid\})
\end{aligned}
$$

$C_{\mathit{fin}}(in, out)$ has identical failures to $C(in, out)$, but has just the restriction on its infinite traces we want for inclusion into the ABP.

What we have managed to capture over $\mathcal{U}$ are the 'real' assumptions on the communication medium that are required to make the protocol work. Of course, having done this, any medium that refines the C_{fin} model is also guaranteed to work.

Fairness

Fairness is a difficult subject, both because it is inextricably linked with unbounded nondeterminism and because interpretations of what it means for a system to be 'fair' vary widely: it can be in terms of

- of one or more specific internal decisions inside the program, as was the one above;
- of the complete pattern of internal decisions, one example being 'angelic nondeterminism': if, whatever decisions are made, it always remains *possible* to attain some desirable state, then eventually that state *must* be reached;
- of the pattern of 'enabled actions' – if an action is enabled infinitely then it will occur; or
- of 'fair parallel': ensuring that each side of a parallel composition is treated fairly in terms of being allowed to make progress.

Fairness in CSP

In CSP one has to be careful about how to define fairness because of the basic model of communication: no action can occur unless the environment is willing to perform it.

For example, we cannot say

$$\mu\, p.(a \to p) \,\Box\, (b \to p)$$

is fair, because the environment can choose the trace $\langle a, a, a, \ldots \rangle$.

We will consider fairness that replaces nondeterministic choice, as with the channel on the previous slide.

Fair nondeterministic choice?

Let us conjecture an operator $\sqcap_F$ that behaves like $\sqcap$ except that it guarantees to make its choices fairly. We know we can use this as a shorthand for the construction used in the channel process, but does it make sense as a CSP operator?

There are several reasons why not:

As a binary operator, $\sqcap$ and $\sqcap_F$ must have the same effect, but this must cease when we put the result in a recursive context:

$$(a \to P) \sqcap (b \to Q) \quad \text{and} \quad (a \to P) \sqcap_F (b \to Q)$$

are identical, but

$$\mu\, p.(a \to p) \sqcap (b \to p) \quad \text{and} \quad \mu(a \to p) \sqcap_F (b \to p)$$

are not.

Failure of distribution

Consider the 'identity'

$$(P \sqcap Q);\ (R \sqcap_F S) = (P;\ (R \sqcap_F S)) \sqcap (Q;\ (R \sqcap_F S))$$

which should be provable by the left distributive law of ; .

But the natural interpretation of $\sqcap_F$ distinguishes them: the left-hand side can perform

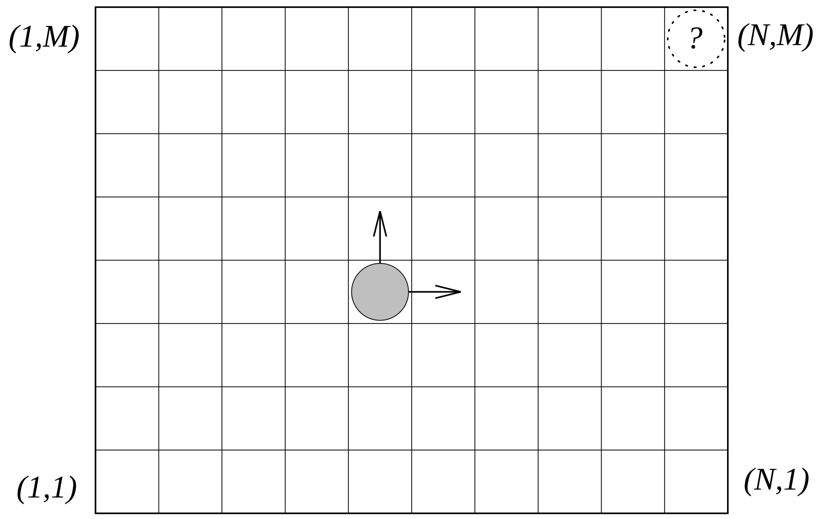
$$P;\ R;\ Q;\ S;\ P;\ R;\ Q;\ S;\ P;\ R;\ Q;\ S \ldots\ldots$$

but the right-hand side cannot: it has to be fair in how it follows P, and in how it follows Q!

We can conclude that fair choice has to be introduced via syntactic transformation, rather than be giving it a semantics of its own: see the book for details.

Fair counter

Fairness is generally used to ensure progress of some sort in a system. For example, imagine a counter sitting on a rectangular board



$$
\begin{array}{lll}
CTR(c,i,j) & = & procrastinate.c \rightarrow CTR(c,i,j) \\
& & \sqcap\ (req.i.j+1.c \rightarrow \\
& & \quad (ref.i.j+1.c \rightarrow CTR(c,i,j) \\
& & \quad \Box\ ok.i.j+1.c \rightarrow enter.i.j+1.c \rightarrow \\
& & \quad leave.i.j.c \rightarrow CTR(c,i,j+1))) \\
& & \sqcap\ (req.i+1.j.c \rightarrow \\
& & \quad (ref.i+1.j.c \rightarrow CTR(c,i,j) \\
& & \quad \Box\ ok.i+1.j.c \rightarrow enter.i+1.j.c \rightarrow \\
& & \quad leave.i.j.c \rightarrow CTR(c,i+1,j)))
\end{array}
$$

Eventuality

Without fairness the counter need never move, but with it it will eventually reach the square (N, M).

To specify that a system P must eventually (unless deadlocked by the environment) communicate a specific event a, all you have to specify is

$$a \rightarrow \mathbf{div} \sqsubseteq P \setminus (\Sigma \setminus \{a\})$$

over either $\mathcal{N}$ or $\mathcal{U}$ as appropriate to the constructs used.

This says that the process P can neither diverge nor engage in an infinite sequence of non-a actions before communicating a, and neither can it deadlock or terminate before doing so.

What do you think happens if there are two or more counters on the board, each acting (individually) fairly?