Determinism

Recall that a process P is deterministic iff

- $divergences(P) = \{\}$, and
- $s\hat{}\langle a\rangle \in traces_\perp(P) \Rightarrow (s, \{a\}) \notin failures_\perp(P)$

We need the full power of the failures/divergences model to decide this, since, for example

$$P \quad \text{and} \quad P \sqcap \mathsf{div}$$

always have the same value in $\mathcal{F}$ and the right-hand-side isn't deterministic when P is.

However, if P is known to be divergence-free, you *can* decide it in $\mathcal{F}$ (hence the failures model determinism check in FDR).

Let $\mathcal{D}$ denote the deterministic members of $\mathcal{N}$.

Maximality I

All members of $\mathcal{D}$ are maximal (under $\sqsubseteq$) in $\mathcal{N}$.

If not, we would have $(F, \{\}) \in \mathcal{D}$ and
$(F, \{\}) \sqsubset (F', \{\}) \in \mathcal{N}$.

$traces(F) = traces(F')$ because if not there is minimal $t\hat{}\langle a \rangle \in traces(F) \setminus traces(F')$. But then $(t, \{\}) \in F'$ and $(t, \{a\}) \notin F \sqsubseteq F'$, contradicting F3.

Each failure of F is forced by F3 relative to the given trace-set, so F' has them all, contradicting our assumption.

Correspondence with $\mathcal{T}$

There is at most one deterministic process for each set of traces, because the refusal sets are deducible from traces.

If a member of $\mathcal{T}$ has both $s\hat{}\langle\checkmark\rangle$ and $s\hat{}\langle a\rangle$ for $a \neq \checkmark$ then there is no corresponding deterministic processes since

$$(s, \{a\}) \in \mathit{failures}_\perp(P) \quad \text{by F4 and F2.}$$

Let $\mathcal{T}^d$ be the subset of $\mathcal{T}$ with no alternatives to $\checkmark$ after any s. By the above

$\mathit{traces}(P) \in \mathcal{T}^d$ for all $P \in \mathcal{D}$.

And if $P \in \mathcal{T}^d$ then $\mathit{det}(Q) = (F, \{\})$, where

$$F = \{(s, X) \mid s \in Q \wedge X \subseteq \Sigma^{\checkmark} \setminus \{a \mid s\hat{}\langle a\rangle \in Q\}\}$$

is deterministic.

Thus there is a 1–1 correspondence between $\mathcal{D}$ and $\mathcal{T}^d$.

Maximality II

Any process P has a deterministic refinement: $det(T)$, where

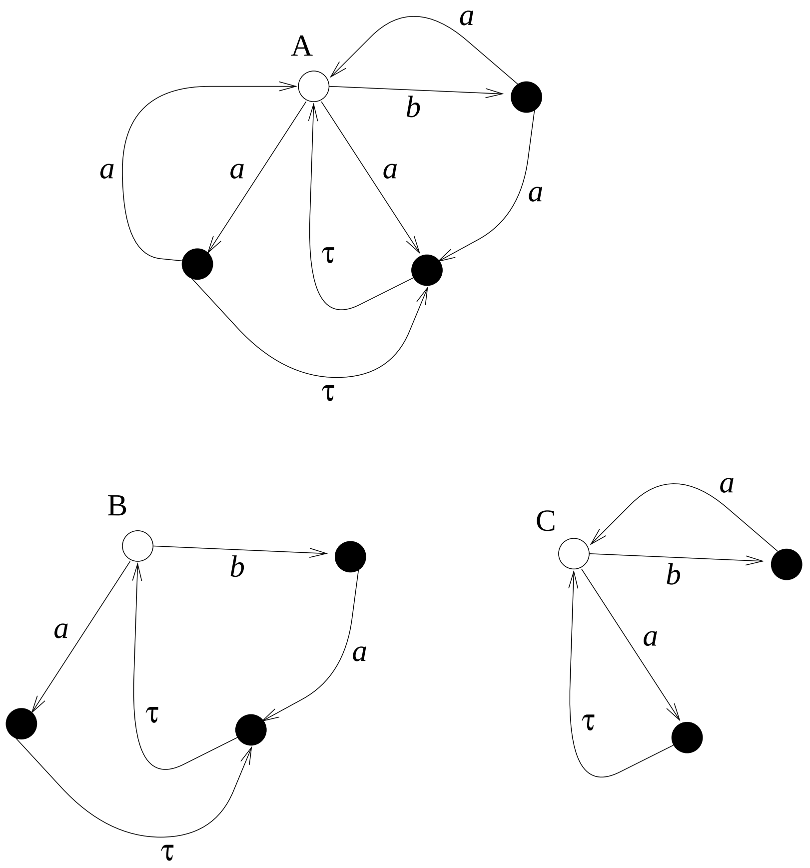
$$T = traces_{\perp}(P) \setminus \{s\hat{}t \mid s\hat{}\langle\checkmark\rangle \in traces_{\perp}(P) \wedge t \in \Sigma^*\}$$

It follows that the deterministic processes are the *only* maximal processes.

FDR decides whether a process P is deterministic by extracting deterministic $P' \sqsupseteq P$ (by manipulation of its LTS) and then discovering if $P' \sqsubseteq P$.

This is true iff P is deterministic, because of the maximality of P'.

Extracting a deterministic sub-LTS



The LTS is searched from its root, selecting one τ or ✓ if any exist at a given node, or alternatively one representative from each visible action.

This there may be many implementations producible in this way (the above shows 2 possible results of applying the procedure to the top LTS) and might well only cover a proper subset of the LTS.

Preserving determinism

We can say an operator *introduces nondeterminism* if it can produce a nondeterministic process when applied to members of $\mathcal{D}$.

The operators that *preserve* determinism are

- prefix choice $?x : A \to P(x)$
- sequencing $P;\ Q$
- $1{-}1$ renaming
- alphabetised parallel $P\ {}_X\|_Y\ Q$
- $P \mathbin{\Box} Q$ when $initials(P) \cap initials(Q) = \{\}$

Recursion preserves determinism

If P is nondeterministic then there is n such that

$$P \downarrow n = Q \downarrow n \Rightarrow Q \notin \mathcal{D}$$

because the nondeterminism shows up on a given trace.

It follows that $\mathcal{D}$ is a *closed* subset of the metric space $\mathcal{N}$, because if $\langle P_n \mid n \in \mathbb{N} \rangle$ is a sequence of deterministic processes converging to P then $P \in \mathcal{D}$.

A closed nonempty subset of a complete metric space is itself a complete metric space. Thus constructive

$$F : \mathcal{D} \to \mathcal{D}$$

has a unique fixed point.

Thus any constructive function built out of operators that preserve determinism must have its unique (in the whole of $\mathcal{N}$) fixed point in $\mathcal{D}$.

In this sense we can add recursion to the list of operators preserving determinism.

Analysing fixed points

The argument on the last slide is an example of the following principle:

RECURSION INDUCTION If $\mu\, p.F(p)$ is a recursion and R is a property (of processes) such that $R(P) \Rightarrow R(F(P))$ then infer $R(\mu\, p.F(p))$.

(which can also be formulated for mutual recursions).

This principle is only true provided appropriate conditions hold of $F(\cdot)$ and R. One such (for any of $\mathcal{T}$, $\mathcal{F}$ and $\mathcal{N}$) is that $F(\cdot)$ is constructive and $\{P \mid R(P)\}$ is closed (i.e., R is *continuous*) and nonempty (i.e., R is *satisfiable*).

This *metric* version deals with just about every practical need, but it is interesting to investigate alternatives ....

Behavioural specifications

The most common type of specification is ones that can be written

$$R(P) \equiv Q \sqsubseteq P$$

R is a lifted condition on behaviours (traces, etc.)

(e.g., any written as **sat** $S(tr)$)

Any behavioural specification over $\mathcal{T}$, $\mathcal{F}$ or $\mathcal{N}$ is continuous, because any process P that fails it does so in a single, *finite* behaviour. For any n greater than its length

$$P \downarrow n = Q \downarrow n$$

implies Q has the behaviour too.

If $R(P) \equiv Q \sqsubseteq P$ then

$$\forall\, P.R(P) \Rightarrow R(F(P))$$

is equivalent, by monotonicity, to

$$Q \sqsubseteq F(Q)$$

Partial order arguments

Suppose we are working in $\mathcal{T}$ or $\mathcal{F}$ and that $Q \sqsubseteq F(Q)$.

The set of processes

$$\{P \mid P \sqsupseteq Q\} = \{P \mid P \subseteq Q\}$$

is a (sub)-complete lattice of $\mathcal{T}$ or $\mathcal{F}$ which is preserved by F. It thus (by Tarski's theorem) contains a fixed point of F.

As it is downwards-closed under $\subseteq$ (the order generating fixed points for recursions) it contains the least fixed point. Hence

$$Q \sqsubseteq \mu\, p.F(p)$$

Provided $\mu\, p.F(p)$ is divergence-free we can use the strong order to establish a similar result for $\mathcal{N}$:

Easy when Σ is finite: F must have a fixed point above Q (since $\{Q \mid P \sqsupseteq Q\}$ is a sub-cpo), but F has a unique fixed point as divergence-free processes are maximal under $\leq$.

Infinite Σ requires a topological argument.

Full Abstraction

A semantic model is said to be *fully abstract* if it meets two conditions:

- The semantics $\mathcal{S}[\![\cdot]\!]$ should distinguish two programs P and Q if, and only if, they are naturally distinguished. This is usually via one of $C[P]$ and $C[Q]$ passing, and the other failing, some simple test.
- The model $\mathcal{M}$ being used should not contain redundant elements. Usually we want all elements to be limits of ones expressible via programs.

In fact, for CSP (thanks to its infinitary constructs) *all* are expressible. Note that this will show that the healthiness conditions of the models are *complete* in the strongest possible sense.

We need separate analyses of each model. We will give details only for $\mathcal{T}$. (See the book for $\mathcal{F}$ and $\mathcal{N}$.)

Expressibility for $\mathcal{T}$

If $s \in \Sigma^{*\checkmark}$, T_s has trace s and as few others as possible: either

$$a_1 \to a_2 \to \ldots \to a_k \to STOP \qquad \text{or}$$

$$a_1 \to a_2 \to \ldots \to a_{k-1} \to SKIP$$

depending on whether s has last member $\checkmark$.

It follows that if $S \in \mathcal{T}$, then $S = traces(\bigsqcap\{\, T_s \mid s \in S\})$ So *every* member of $\mathcal{T}$ is the image of an infinitary and rather useless CSP process.

A similar but more involved construction works for $\mathcal{F}$.

For $\mathcal{N}$ with finite Σ, the construction produces a much "better" process, because the model doesn't leave large gaps in what is recorded. *See the book!*

Distinguishing processes in $\mathcal{T}$

Say a process fails the test $\mathbf{T}_1$ if it has the trace $\langle fail \rangle$.

If $C[\cdot]$ is any CSP context and P is a process, we can clearly tell whether $C[P]$ fails $\mathbf{T}_1$ from the $\mathcal{T}$ semantics. Thus if

$$traces(P) = traces(Q)$$

then for any context, $C[P]$ and $C[Q]$ will both either pass or fail $\mathbf{T}_1$.

If $traces(P) \neq traces(Q)$, we need a context that will distinguish them via this test. We can do this by finding $CT_s[\cdot]$ such that $CT_s[P]$ fails $\mathbf{T}_1$ precisely when $s \in traces(P)$.

Building a trace context

When $s = s'\hat{}\langle\checkmark\rangle$, we can set

$$CT_s[V] = (V \underset{\Sigma}{\parallel} T_s) \setminus \Sigma;\ (\mathit{fail} \to STOP)$$

For $s \in \Sigma^*$, we can use $CT_s[V] =$

$$(V \underset{\Sigma}{\parallel} T_s)[\![a/x \mid x \in \Sigma]\!] \underset{\{a\}}{\parallel} FAIL(\#s)) \setminus \{a\}$$

where $a \neq \mathit{fail}$ is a member of Σ and $FAIL(m)$ is defined

$$\begin{array}{rcl} FAIL(m) & = & (\mathit{fail} \to STOP) \\ & & \not< m = 0 \not> \\ & & (a \to FAIL(m-1)) \end{array}$$

Both allow V to communicate only s, and find a way of hiding these actions and communicating fail when s is finished.

The renaming is needed in case V uses fail.

Thus $\mathcal{T}$ is fully abstract with respect to $\mathbf{T}_1$.

Full abstraction for $\mathcal{F}$ and $\mathcal{N}$

Any test for $\mathcal{F}$ must include something about refusal sets.

Define a process to fail $\mathbf{T}_2$ if *either* it deadlocks immediately *or* has the trace $\langle fail \rangle$.

One can show *(see book)* that $\mathcal{F}$ is fully abstract with respect to $\mathbf{T}_2$ and with respect to the simpler test "cannot deadlock immediately" if we add an interrupt operator:

$$P \triangle_a Q$$

behaves like P until a occurs, then like Q.

Similarly $\mathcal{N}$ is fully abstract with respect to the test $\mathbf{T}_3$, failed if *either* P deadlocks immediately *or* diverges immediately.

Proving a model fully abstract with respect to a given test $\mathbf{T}$ establishes that it is the weakest congruence which distinguishes between two processes if one passes, and the other fails, $\mathbf{T}$.

For example, for CSP extended by $\triangle_a$, $\mathcal{F}$ is the weakest congruence that allows us to tell if a process can deadlock immediately.

Reconciling viewpoints

We have two distinct ways of working out the values in $\mathcal{T}$, $\mathcal{F}$ and $\mathcal{N}$ of any CSP process:

- directly (denotational semantics)
- extracting sets of behaviours from operational semantics.

We have not distinguished between these notationally (e.g., $traces(P)$ could have come from either), but it is by no means obvious that the two approaches always give the same answer.

That they do is to say that the operational and denotational semantics are *congruent*.

We will now outline the proofs that this is true.

Notation

For this discussion we have to me more careful about distinctions between syntactic CSP terms, and the various ways of computing the semantics.

Let $\mathbf{CSP}$ be the set of syntactic terms,
and let $\overline{\mathbf{CSP}}$ be the set of closed terms (ones with no free identifiers).

For each node σ of an LTS V

$$\begin{array}{rcl}
\Phi_{\mathcal{T}}(\sigma) & = & traces(\sigma) \\
\Phi_{\mathcal{F}}(\sigma) & = & (traces(\sigma), failures(\sigma)) \\
\Phi_{\mathcal{N}}(\sigma) & = & (failures_{\perp}(\sigma), divergences(\sigma))
\end{array}$$

These can, of course, be used to compute one set of interpretations of CSP from the LTS (of closed terms) generated by the operational semantics.

Let $\mathcal{S}_{\mathcal{T}}[\![\cdot]\!]$, $\mathcal{S}_{\mathcal{F}}[\![\cdot]\!]$ and $\mathcal{S}_{\mathcal{N}}[\![\cdot]\!]$ be the three denotational semantic functions.

Proof overview

We are actually proving three congruence theorems, of course.

Each goes by structural induction on a term $P \in \mathbf{CSP}$.

This might have free process identifiers, so the statements of the results have to accommodate this: for $\mathcal{M} \in \{\mathcal{T}, \mathcal{F}, \mathcal{N}\}$ define

$$Env_{\mathcal{M}} = Ide \to \mathcal{M}$$

to be the set of $\mathcal{M}$-environments, so that the type of the semantic function becomes

$$\mathcal{S}_{\mathcal{M}} : \mathbf{CSP} \to Env_{\mathcal{M}} \to \mathcal{M}$$

and, for a process term P and environment $\rho \in Env_{\mathcal{M}}$, $\mathcal{S}_{\mathcal{M}}[\![P]\!]\rho$ is the value of P when its free identifiers have the values assigned by ρ.

Proof overview II

The operational analogue of $\rho \in Env_{\mathcal{M}}$ is a substitution ξ that maps each identifier to a closed CSP term:

$$\xi \in Subst : Ide \to \overline{\mathbf{CSP}}$$

ξ then converts any CSP term P into a closed one $subst(P, \xi)$ in the obvious way.

The statement of the congruence theorem for $\mathcal{M}$ is then

$$\Phi_{\mathcal{M}}(subst(P, \xi)) = \mathcal{S}_{\mathcal{M}}[\![P]\!]\overline{\xi}^{\mathcal{M}}$$

where $\overline{\xi}^{\mathcal{M}} \in Env_{\mathcal{M}}$ is defined

$$\overline{\xi}^{\mathcal{M}}(p) = \Phi_{\mathcal{M}}(\xi(p))$$

It is this result we prove by structural induction over P (for all ξ simultaneously).

The claim is trivially true for all 'base' cases of the induction: P being either a process identifier p or one of the constant processes $STOP$ and $SKIP$.

An inductive step

The inductive proof then follows the same pattern for each non-recursive operator.

For example $\Box$ over $\mathcal{T}$:

$$\begin{aligned} subst(P \Box Q, \xi) &= (subst(P, \xi)) \Box (subst(Q, \xi)), \quad \text{and} \\ \mathcal{S}_{\mathcal{T}}[\![P \Box Q]\!]\overline{\xi}^{\mathcal{T}} &= \mathcal{S}_{\mathcal{T}}[\![P]\!]\overline{\xi}^{\mathcal{T}} \Box_{\mathcal{T}} \mathcal{S}_{\mathcal{T}}[\![Q]\!]\overline{\xi}^{\mathcal{T}} \end{aligned}$$

where $\Box_{\mathcal{T}}$ is the representation of this operator over $\mathcal{T}$, namely $\cup$ (union).

Thus, by induction, it is sufficient to prove that for any $R, S \in \overline{\mathbf{CSP}}$,

$$\Phi_{\mathcal{T}}(R \Box S) = \Phi_{\mathcal{T}}(R) \Box_{\mathcal{T}} \Phi_{\mathcal{T}}(S)$$

This result (and the corresponding ones for other operators) is proved by examining operational behaviour.

Reminder

Recall that the operational semantics of $\Box$ are defined:

$$\frac{P \xrightarrow{\tau} P'}{P \Box Q \xrightarrow{\tau} P' \Box Q} \qquad \frac{Q \xrightarrow{\tau} Q'}{P \Box Q \xrightarrow{\tau} P \Box Q'}$$

Visible actions resolve the choice:

$$\frac{P \xrightarrow{a} P'}{P \Box Q \xrightarrow{a} P'}(a \neq \tau) \qquad \frac{Q \xrightarrow{a} Q'}{P \Box Q \xrightarrow{a} Q'}(a \neq \tau)$$

External choice analysed

It is easy to show that, for any R and S,

- If $R \stackrel{\langle\rangle}{\Longrightarrow} R'$ and $S \stackrel{\langle\rangle}{\Longrightarrow} S'$ then $R \,\Box\, S \stackrel{\langle\rangle}{\Longrightarrow} R' \,\Box\, S'$.
- If $s \neq \langle\rangle$ and either $R \stackrel{s}{\Longrightarrow} T$ or $S \stackrel{s}{\Longrightarrow} T$ then $R \,\Box\, S \stackrel{s}{\Longrightarrow} T$.
- The above are the only pairs (s, V) such that $R \,\Box\, S \stackrel{s}{\Longrightarrow} V$.

This, of course, establishes the correspondence we need for the inductive step $\Box$.

(All that is required for $\Box$ and $\mathcal{F}$, $\mathcal{N}$ is a refinement of the above lemma to record what each state can refuse and whether it is divergent.)

Recursion

Proving the congruence of recursion is usually the most challenging phase of one of these proofs.

This is perhaps because the operational (via rewriting) and denotational (via least fixed points) semantics are so different conceptually.

What we have to show (for $\mathcal{T}$) is that, for any ξ, $\Phi_{\mathcal{T}}(subst(\mu\, p.P, \xi))$ is the least fixed point of the function $\Upsilon(\alpha)$ mapping $\alpha \in \mathcal{T}$ to $\mathcal{S}_{\mathcal{T}}[\![P]\!](\overline{\xi}^{\mathcal{T}}[\alpha/p])$.

It is relatively easy (and model-independent) to show that it is a fixed point .....

Rewriting gives a fixed point

Let P' be the term in which all free identifiers of P other than p have been substituted by ξ, so that $subst(\mu\, p.P, \xi) = \mu\, p.P'$. Then

$$\begin{array}{rcll}
\Phi_{\mathcal{T}}(subst(\mu\, p.P, \xi)) & = & \Phi_{\mathcal{T}}(\mu\, p.P') & \\
& = & \Phi_{\mathcal{T}}(P'[\mu\, p.P'/p]) & \text{(a)} \\
& = & \Phi_{\mathcal{T}}(subst(P, \xi[\mu\, p.P'/p])) & \text{(b)} \\
& = & \mathcal{S}_{\mathcal{T}}[\![P]\!]\overline{\xi[\mu\, p.P'/p]}^{\mathcal{T}} & \text{(c)} \\
& = & \mathcal{S}_{\mathcal{T}}[\![P]\!](\overline{\xi}^{\mathcal{T}}[\Phi_{\mathcal{T}}(\mu\, p.P')/p]) & \text{(d)} \\
& = & \Upsilon(\Phi_{\mathcal{T}}(subst(\mu\, p.P, \xi))) &
\end{array}$$

(a) follows because of the operational semantics of recursion:

$$\mu\, p.P' \xrightarrow{\tau} P'[\mu\, p.P'/p]$$

and because if A is a node of an LTS whose only action is $A \xrightarrow{\tau} B$ then $\Phi_{\mathcal{T}}(A) = \Phi_{\mathcal{T}}(B)$. (b) follows by definition of P', (c) comes by induction on the term P which is simpler than $\mu\, p.P$, and (d) by definition of $\overline{\xi}^{\mathcal{T}}$.

Which fixed point?

For $\mathcal{T}$, we chose to identify recursions with the subset-least fixed points of the associated functions. This least fixed point is given by

$$\bigcup_{n=0}^{\infty} \Upsilon^n(\{\langle\rangle\})$$

since every CSP-definable function is continuous.

We know (by the argument on the last slide) that this is a subset of $\Phi_{\mathcal{T}}(subst(\mu\, p.P, \xi))$.

Thus all we need to do is show that each trace in the latter is in $\Upsilon^n(\{\langle\rangle\})$ for some n.

If s is such a trace then there must be some $s' \in (\Sigma^{\tau})^{*\checkmark}$ and Q such that $s' \setminus \tau = s$ and $subst(\mu\, p.P, \xi) \stackrel{s'}{\longmapsto} Q$.

If $N = \#s'$, then during the derivation of the behaviour s', the recursion cannot possibly be unfolded more than N times, since every time it is unfolded a τ action is generated which must be present in s'.

Completing the proof

We can now show $s \in \Upsilon^n(\{\langle\rangle\})$ via the following two mutual recursions

$$\begin{array}{rcll} p_n & = & P'[p_{n+1}/p] & n \in \mathbb{N} \end{array}$$

$$\begin{array}{rcll} q_n & = & P'[q_{n+1}/p] & n \in \{0, \ldots, N-1\} \\ q_N & = & STOP & \end{array}$$

where P' is as before.

p_0 behaves identically to $subst(\mu\, p.P, \xi)$: they have identical operational semantics except for identifier names. We therefore know that p_0 has the behaviour s'.

q_0 behaves exactly like p_0 until after the Nth unfolding, which means that it has the behaviour s'; this means that $s \in \Phi_{\mathcal{T}}(q_0)$.

However it is not hard to see that (by induction)

$$\Phi_{\mathcal{T}}(q_0) = \Upsilon^N(\{\langle\rangle\})$$

so $s \in \Upsilon^N(\{\langle\rangle\})$, which is what we wanted to show.

This completes our sketch of the proof of congruence between the operational semantics and the trace semantics.

Other models

The proof for $\mathcal{F}$ is almost exactly the same with a few more details.

The only real difference with $\mathcal{N}$ is in the handling of the recursion case, since we are now

- establishing the $\sqsubseteq$-least fixed point (i.e., the "opposite" one), and
- dealing with behaviours (divergences) which may take an infinite number of unfoldings of the recursion to derive.

The proof that the operational semantics gives *a* fixed point is identical to that over $\mathcal{T}$.

Justifying the $\sqsubseteq$-least fixed point

Under the assumption of finite nondeterminism (necessary for $\mathcal{N}$ to be accurate) you will find that for any trace s

- *either* the tree of nodes reachable by a process *en route* to, or with, trace s is finite
- *or*, by König's lemma, it contains an infinite path of τ's

In the first case you can use a similar argument to that over $\mathcal{T}$ to show that

$$initials(\Phi_{\mathcal{N}}(subst(\mu\, p.P, \xi))/s) = initials(\Upsilon^N(\bot_{\mathcal{N}})/s)$$

$$refusals(\Phi_{\mathcal{N}}(subst(\mu\, p.P, \xi))/s) = refusals(\Upsilon^N(\bot_{\mathcal{N}})/s)$$

for all N larger than the depth of the finite tree.

In the second axioms D1 and D2 mean there is nothing to consider about what happens on s: $\Phi_{\mathcal{N}}(subst(\mu\, p.P, \xi))$ certainly has all behaviours of the least fixed point on s.