José Meseguer

As for automata and Petri nets, we call a concurrent computation specification in combinatory logic a labeled arrow:

$$u \xrightarrow{\alpha} v$$

that is provable according to the inference system $\mathcal{L}(CL)$, i.e. we can build a proof tree for it as the root using the inference rules of $\mathcal{L}(CL)$. This gives as a labeled directed graph:

$$Spec(CL) = (CL, ProofTerms_{CL}, \xrightarrow[Spec(CL)]{})$$

where CL are the CL expressions, ProofTerms are the α such that there are $u, v \in CL$ such that $u \xrightarrow{\alpha} v$ is a concurrent computation specification, and $u \xrightarrow[Spec(CL)]{\alpha} v$ iff $u \xrightarrow{\alpha} v$ conc. comp. spec.

Note that arrows in this graph enjoy two operations:

1. Application: $(u \xrightarrow{\alpha} u', v \xrightarrow{\beta} v') \mapsto uv \xrightarrow{\alpha\beta} u'v'$
2. Composition: $(u \xrightarrow{\alpha} v, v \xrightarrow{\beta} w) \mapsto u \xrightarrow{\alpha;\beta} w$

These two operations exist and are well defined precisely because of the <u>Congruence</u> and <u>Transitivity</u> inference rules. Note that $CL \subseteq ProofTerms_{CL}$, and that $-;-$ is defined on both, and both operations <u>coincide</u> on CL (CL is a <u>subalgebra</u>)

As before, the million-dollar question is:

<u>When are two CL computation specs the same?</u>

or, what is equivalent:

<u>what is a natural notion of CL computation?</u>

By "engineering induction" based on our previous experience with automata and Petri nets, we can safely gues that:

1. <u>Associativity of $-;-$</u>: Given $u \xrightarrow{\alpha} v \xrightarrow{\beta} w \xrightarrow{\gamma} q$

$$u \xrightarrow{(\alpha;\beta)\gamma} q \equiv u \xrightarrow{\alpha;(\beta;\gamma)} q$$

2. <u>Identity</u>:

$$u \xrightarrow{u\alpha} v \equiv u \xrightarrow{\alpha} v \equiv u \xrightarrow{\alpha v} v$$

3. <u>Functoriality</u>: Given also $u' \xrightarrow{\alpha'} v' \xrightarrow{\beta'} w'$;

$$uu' \xrightarrow{(\alpha\alpha');(\beta\beta')} ww' \equiv uu' \xrightarrow{(\alpha;\beta)(\alpha';\beta')} ww'$$

are all natural identities that hold between concurrent CL computations. What else? In CL we have encountered a new phenomenon that did not arise either for automata or for Petri nets, namely, the phenomenon of parametric transitions such as:

$$\text{k-red}: (K\,x)\,y \longrightarrow x$$

which are parametric on x, y. That is, for any $u, v \in CL$ we get a concrete basic transition

$$\text{k-red}(u,v): (K\,u)\,v \longrightarrow u$$

So, the natural question to ask is:

Are there natural identities between CL computations associated to the parametric transitions k-red, S-red, I-red?

Looking at a concrete example may give us some hints. Consider the computation specification:

$$((S\,((K\,u)\,v))(I\,u))(I\,v) \xrightarrow{\text{S-red}(\text{k-red}(u,v),\text{I-red}(u),\text{I-red}(v))} (u\,v)(u\,v)$$

which can be naturally decomposed as follows:

in page 5 of Lecture 7a

$$\begin{tikzcd}
((S((Ku)v))(Iu))(Iv) \arrow[rr, "S\text{-red}((Ku)v,\, Iu,\, Iv)"] \arrow[d, "(S(K\text{-red}(u,v))\,I\text{red}(u))\,I\text{red}(v)"'] \arrow[drr, "S\text{-red}(K\text{-red}(u,v),\, I\text{-red}(u),\, I\text{red}(v))"] & & (((Ku)v)(Iv))((Iu)(Iv)) \arrow[d, "(K\text{-red}(u,v)\,I\text{red}(v))(I\text{red}(u)\,I\text{red}(v))"] \\
((Su)u)v \arrow[rr, "S\text{-red}(u,u,v)"] & & (uv)(uv)
\end{tikzcd}$$

That is, we can decompose the original computation by either:

Upper Decompos^on^ (1) applying S-red and then (2) Applying K-red, Ired(u) and I-red v) in parallel (Congruence)

or

Lower Decomp^on^ (1) applying K-red, Ired(u), Ired(v) and then (2) applying S-red in parallel (congruence)

Of course, the way in which K-red, Ired(u) and I-red(v) are applied in parallel (Congruence) in the upper and lower cases is different, i.e., according to the terms $((Sx)y)z$ and $(xz)(yz)$ in

$$S\text{-red}: ((Sx)y)z \longrightarrow (xz)(yz)$$

So, what does this suggest? The following:

For any $u \xrightarrow{\alpha} u'$, $v \xrightarrow{\beta} v'$, $w \xrightarrow{\gamma} w'$ CL comp. specs

we have:

$$
\begin{tikzcd}
((S\, u)\, v)\, w \arrow[rr, "\text{S-red}(u,v,w)"] \arrow[d, "(S(\alpha)\beta)\gamma"'] \arrow[drr, "\text{S-red}(\alpha,\beta,\gamma)"] & & (u\, w)(v\, w) \arrow[d, "(\alpha\gamma)(\beta\gamma)"] \\
((S\, u')\, v')\, w' \arrow[rr, "\text{S-red}(u',v',w')"'] & \equiv & (u'\, w')(v'\, w')
\end{tikzcd}
$$

$\equiv$

or, more briefly:

4. $\boxed{((S(\alpha)\beta)\gamma);\ \text{S-red}(u',v',w') \equiv \text{S-red}(\alpha,\beta,\gamma) \equiv \text{S-red}(u,v,w); ((\alpha\gamma)(\beta\gamma))}$

For the exact same reason for <u>K-red</u> we get:

5. $\boxed{((K\alpha)\beta);\ \text{K-red}(u',v') \equiv \text{K-red}(\alpha,\beta) \equiv \text{K-red}(u,v);\ \alpha}$

An for I-red we get:

6. $\boxed{(I\alpha);\ \text{I-red}(u') \equiv \text{I-red}(\alpha) \equiv \text{I-red}(u);\ \alpha}$

All the identifications (1)-(6) are completely <u>natural</u> and gives a natural notion of CL <u>parallel computation</u>:

<u>Definition</u> (Category $\mathcal{T}_{CL}$ of CL-computations).

The category $\mathcal{T}_{CL} = (CL, ProoTerms_{CL}/\equiv, \underset{\mathcal{T}_{CL}}{\rightarrow})$

defined by: (i) Identifying proof terms according to the smallest <u>congruence</u> generated by the equivalences (1)-(6) in the labeled graph Spec(CL) with the two arrow operations of application and composition, ~~is called the category of parallel CL computations.~~

~~That $\mathcal{T}_{CL}$ is a category~~ and (ii) defining:

$$u \xrightarrow[\mathcal{T}_{CL}]{[\alpha]_\equiv} v \quad \text{iff} \quad u \xrightarrow{\alpha} v \text{ CL spec. comp.}$$

is called the <u>category of parallel CL computations.</u>

That $\mathcal{T}_{CL}$ is a <u>category</u> follows immediately from the equivalences (1)-(2). But what other properties does $\mathcal{T}_{CL}$ have?

Since $\equiv$ denotes the smallest <u>congruence</u> generated by (1)-(6), this means that it <u>respects</u> application and composition, Since it respects composition we of course have that $\equiv$ $\mathcal{T}_{CL}$ is a category. But what does it mean that $\equiv$ respects application? It means that if

$$(u \xrightarrow{\alpha} u') \equiv (u \xrightarrow{\alpha'} u') \text{ and } (v \xrightarrow{\beta} v') \equiv (v \xrightarrow{\beta'} v')$$

then we must have: $(uv \xrightarrow{\alpha\beta} u'v') \equiv (uv \xrightarrow{\alpha'\beta'} u'v')$.

But this exactly means that in $\mathcal{T}_{CL}$ we also have, as in $Spec(CL)$ an *application operation on arrows*:

$$(u \xrightarrow{[\alpha]_\equiv} u', v \xrightarrow{[\beta]_\equiv} v') \mapsto uv \xrightarrow{[\alpha\beta]_\equiv} u'v'$$

What other properties does $\mathcal{T}_{CL}$ enjoy?

To answer this ~~property~~ question, we need to consider the concept of a *natural transformation*. What is that? To answer this second question we need to consider the concept of a *functor*. What is that? To answer this question we just need to first consider the hopefully familiar concept of (labeled) *graph homomorphism*.

<u>Definition</u>. Given labeled directed graphs $\mathcal{G} = (N, L, \underset{\mathcal{G}}{\rightarrow})$ and $\mathcal{G}' = (N', L', \underset{\mathcal{G}'}{\rightarrow})$, a (labeled) graph-homomorphism $F : \mathcal{G} \rightarrow \mathcal{G}'$ is a function $F : N \rightarrow N'$ on nodes and $F : L \rightarrow L'$ on labels (we use the same <u>notation</u> for both for simplicity) such that: [ ]

$$\boxed{\underline{\text{If}} \quad n \xrightarrow[\mathcal{G}]{\ell} n' \quad \underline{\text{then}} \quad F(n) \xrightarrow[\mathcal{G}']{F(\ell)} F(n')}$$

<u>Definition</u>. Given categories $\mathcal{A} = (O, A, \underset{\mathcal{A}}{\rightarrow})$ and $\mathcal{A}' = (O', A', \underset{\mathcal{A}'}{\rightarrow})$ a <u>functor</u> $F : \mathcal{A} \rightarrow \mathcal{A}'$ is a graph homomorphism $F : \mathcal{A} \rightarrow \mathcal{A}'$ such that:

1. <u>Preserves composition</u>, i.e., if $u \xrightarrow[\mathcal{A}]{\alpha} v \xrightarrow[\mathcal{A}]{\beta} w$,

then $\left(F(u) \xrightarrow{F(\alpha;\beta)} F(w) \right) = \left(F(u) \xrightarrow{F(\alpha);' F(\beta)} F(w) \right)$

2. <u>Preserves Identities</u>: $\left(F(u) \xrightarrow{F(id_u)} F(u) \right) = \left(F(u) \xrightarrow{id'_{F(u)}} F(u) \right)$

were $-;'-$ and id'_- denote compos. and ident. in $\mathcal{A}'$.

Let us some examples of functors:

1. $\mathcal{U}$: CommMon $\longrightarrow$ Set forgets about the monoid structure, i.e.:

$$\mathcal{U}\left((M,+,0) \xrightarrow{f} ((M',+',0')\right) = M \xrightarrow{f} M$$

2. The cartesian product of sets is also a functor:

$$\_\times\_ : \underline{Set} \times \underline{Set} \longrightarrow Set$$

$$(A \xrightarrow{f} A') \times (B \xrightarrow{g} B') = A \times B \xrightarrow{f\times g} A' \times B'$$

where we define: $f \times g =_{def} \lambda(a,b) \in A \times B.\,(f(a), g(b)) \in A' \times B'$

But what is the category $\underline{Set} \times \underline{Set}$?

Definition. Given categories $\mathcal{A} = (O, A, \to_{\mathcal{A}})$ and $\mathcal{A}' = (O', A', \to_{\mathcal{A}'})$
$\mathcal{A} \times \mathcal{A}'$ denotes the category: $\mathcal{A} \times \mathcal{A}' = (O \times O', A \times A', \to_{\mathcal{A}\times\mathcal{A}'})$

where: $(n,n') \xrightarrow[\mathcal{A}\times\mathcal{A}']{(f,f')} (m,m')$ iff $n \xrightarrow[\mathcal{A}]{f} m$ and $n' \xrightarrow[\mathcal{A}']{f'} m'$

- $id_{(n,n')} =_{def} (id_n, id'_{n'})$ for any $(n,n') \in O \times O'$
- If $(n,n') \xrightarrow[\mathcal{A}\times\mathcal{A}']{(f,f')} (m,m') \xrightarrow[\mathcal{A}\times\mathcal{A}]{(g,g')} (k,k')$, then

$$\left((n,n') \xrightarrow[\mathcal{A}\times\mathcal{A}]{(f,f');(g,g')} (k,k')\right) =_{def} \left((n,n') \xrightarrow[\mathcal{A}\times\mathcal{A}]{((f,g),(f';g'))} (k,k')\right)$$

3. Therefore, for any Petri net the fact that the <u>union operation</u> acts on the arrows of $\mathcal{T}_N$ is just the fact that $\_\,\_$ is a <u>functor</u>

$$\_\,\_ : \mathcal{T}_N \times \mathcal{T}_N \longrightarrow \mathcal{T}_N$$

$$\left((u,v) \xrightarrow{(\alpha,\beta)} (u',v')\right) \longmapsto \left(uu' \xrightarrow{\alpha\beta} vv'\right)$$

4. Likewise for CL logic <u>application</u> is a <u>functor</u>

$$\_\,\_ : \mathcal{T}_{CL} \times \mathcal{T}_{CL} \longrightarrow \mathcal{T}_{CL}$$

$$\left((u,v) \xrightarrow{(\alpha,\beta)} (u',v')\right) \mapsto \left(uu' \xrightarrow{\alpha\beta} vv'\right)$$

<u>Notation</u> For A set, $\mathcal{A}$ category, $A^n = A \times \overset{n}{\cdots} \times A$, and $\mathcal{A}^n = \mathcal{A} \times \cdots \times \mathcal{A}$.

5. For any n (assume $A^0 = \{*\}$ (one-point set, and $A^1 = A$)

we also have a <u>functor</u>

$$(\_)^n : Set \longrightarrow Set$$

$$\left(A \xrightarrow{f} B\right) \longmapsto \left(A^n \xrightarrow{f^n} B^n\right)$$