Additional Remarks on the Linking Combinator

Luca Aceto Anna Ingolfsdottir Kim G. Larsen Jiří Srba

October 23, 2007

In Exercise 2.13 on page 30 of the textbook, we asked you to draw an initial fragment of the transition graph for a value-passing CCS process whose specification used the derived *linking combinator*.

Intuitively, the linking combinator $\frown$ is used to "connect" complementary ports of two processes so that those ports can only be used by the two processes to communicate one with the other. For example, those amongst you who solved Exercise 2.12 in the book will readily see that one can define a two-place FIFO queue by "linking together" two copies of the one-place buffer Cell defined in Exercise 2.12 as follows:

$$\text{Cell}\frown\text{Cell} \stackrel{\text{def}}{=} (\text{Cell}[a/\text{out}] \mid \text{Cell}[a/\text{in}]) \setminus \{a\} \ .$$

The effect of the linking combinator in the above example is to force the synchronization between the output actions along port 'out' performed by the first copy of the Cell process with the input actions along port 'in' performed by the second copy of the Cell process. This is achieved by renaming both ports to a port name that is *not* used by the Cell process, and by restricting that new port name. (Can you see why it is important to choose a *fresh* port name in renaming the ports that we wish to link?)

In relation to Exercise 2.13, consider now the process

$$\text{C}(0)\frown\text{B} \ .$$

Recall that $\text{C}(0)\frown\text{B}$ is defined in that exercise as follows:

$$\text{C}(0)\frown\text{B} = (\text{C}(0)[p'/p, e'/e, o'/o] \mid \text{B}[p'/\text{push}, e'/\text{empty}, o'/\text{pop}]) \setminus \{p', o', e'\} \ .$$

You may now ask yourself:

> "What role do the relabellings
>
> $$[p'/p, e'/e, o'/o] \text{ and } [p'/\text{push}, e'/\text{empty}, o'/\text{pop}]$$
>
> play when applied to processes $\text{C}(0)$ and B, respectively?"

This is a most reasonable question since a brief look at the specification of a process of the form $\mathrm{C}(x)$ immediately indicates that such a process cannot initially perform actions involving any of the relabelled ports. Similarly, it is clear that process B cannot initially communicate on port 'pop'. So, what is the use of those relabellings?

As usual, the best way to answer this question is to get your hands dirty and to try and simulate the process $\mathrm{C}(0)\frown\mathrm{B}$ using the operational semantics. In solving Exercise 2.13, you will realize that those relabellings ensure, for instance, that the process $\mathrm{C}(0)\frown\mathrm{B}$ cannot initially perform actions involving ports 'push' and 'empty' that are initially afforded by process B. However, there is more to the linking combinator than that!

One of the initial transitions of process $\mathrm{C}(0)\frown\mathrm{B}$ is

$$\mathrm{C}(0)\frown\mathrm{B} \stackrel{\mathrm{push}(1)}{\rightarrow} (\mathrm{C}(1)\frown\mathrm{C}(0))\frown\mathrm{B} \ .$$

A little thought should allow you to conclude that, in the target process of the above transition, the relabelling ensures that also the actions involving ports 'push' and 'pop' afforded by process $\mathrm{C}(0)$ cannot be performed, and that only process $\mathrm{C}(1)$ can execute actions involving ports 'push' and 'pop'. You should be able to convince yourself that

$$(\mathrm{C}(1)\frown\mathrm{C}(0))\frown\mathrm{B} \stackrel{\overline{\mathrm{pop}}(1)}{\rightarrow} (\mathrm{D}\frown\mathrm{C}(0))\frown\mathrm{B} \ .$$

We now encourage you to analyze the effect of the linking combinator on the possible initial transitions of the process

$$(\mathrm{D}\frown\mathrm{C}(0))\frown\mathrm{B} \ .$$

As a result of your analysis, you will find out that the only initial transition afforded by that process is

$$(\mathrm{D}\frown\mathrm{C}(0))\frown\mathrm{B} \stackrel{\tau}{\rightarrow} (\mathrm{C}(0)\frown\mathrm{D})\frown\mathrm{B} \ .$$

Let us conclude this discussion of linking applied to Exercise 2.13 by answering a further question that might have crossed your mind. In simulating the behaviour of process B, you might have noticed that the relabelling of port p with p' plays no role in the specific instances of linking we consider in the exercise. So, why is it there? This is because the linking combinator is generally meant to connect selected pairs of ports in the two processes that are being linked together. Therefore, for reasons of uniformity, we prefer to describe linking in terms of relabellings that change the name of the *same number* of ports in the two arguments of the linking combinator. We have applied this convention in this exercise, even though the relabelling of port p is irrelevant in our specific case.

You will find a wealth of applications of the linking combinator in [1, Chapter 6], a reference that we warmly recommend for further independent study.

Acknowledgment We thank Petr Jančar for suggesting that we clarify the definition of the linking combinator in Exercise 2.13 in our book, and for his punctual comments on a draft version of this note.

References

[1] Milner, R. *Communication and Concurrency*, Prentice-Hall International, Englewood Cliffs.