Beyond Mobility: What Next after CSP/π? *(Invited Talk)*

Michael GOLDSMITH
WMG Digital Laboratory, University of Warwick, Coventry CV4 7AL, UK
m.h.goldsmith@warwick.ac.uk

Abstract. Process algebras like CSP and CCS inspired the original occam model of communication and process encapsulation. Later the π-calculus and various treatments handling mobility in CSP added support for mobility, as realised in practical programming systems such as occam-π, JCSP, CHP and Sufrin's CSO, which allow a rather abstract notion of motion of processes and channel ends between parents or owners. Milner's *Space and Motion of Communicating Agents* on the other hand describes the bigraph framework, which makes location more of a first-class citizen of the calculus and evolves through reaction rules which rewrite both place and link graphs of matching sections of a system state, allowing more dramatic dynamic reconfigurations of a system than simple process spawning or migration. I consider the tractability of the notation, and to what extent the additional flexibility reflects or elicits desirable programming paradigms.

Keywords. bigraphs, formal modelling, refinement, verification.

Introduction

The Communicating Process Architecture approach to programming can be traced back to the occam of the 1980s [1], which in turn was the result of grafting together a rather simple (and hence amenable to analysis) imperative language with the interaction model of the two competing process algebras, CCS [2] and CSP [3,4]; occam lies squarely within the sphere where the two can agree, although both allow more dynamic process evolution (such as recursion through parallel operators in the case of CSP), while occam deliberately enforces a rather static process structure, to simplify memory allocation and usage checking. The π-calculus [5,6] extends CCS with the ability dynamically to create and communicate (event/channel) names, which can then be used for further communication by their recipients. These notions have been realised in practical imperative programming systems such as *Communicating Sequential Processes for Java* (JCSP) [7], occam-π [8] and *Communicating Scala Objects* [9], while *Communicating Haskell Processes* (CHP) [10] embeds them into a functional-programming setting.

We have grown accustomed to thinking of the ability to transfer channel ends and (running) processes as "mobility", but often this is little more than a metaphor: movement is between logical owners and perhaps processors, but (at least within these languages) there is no notion of geographical location or physical proximity, such as would be needed to treat adequately of pervasive systems or location-aware services. For considerations of this kind, something like the *Ambient Calculus* [11] might be more appropriate, but this concentrates rather exclusively on entry to and exit from places, rather than communication and calculation per se.

Milner's bigraphs [12] provide a framework within which all these formalisms can be represented and perhaps fruitfully cohabit. In this paper, I give a brief overview of the notions and some musings on the potential and challenges they bring with them.

1. Bigraphs

A *bigraph* is a graph with two distinct sets of edges, or, if you prefer, a pair of graphs with a (largely) common set of vertices. One is the *place graph*, a forest of vertices whose leaves (if any – there may equally be nodes with out-degree 0) are *sites*, placeholders into which another place graph might be grafted; a signature is a set of *control types* (with which the vertices of the place graph are annotated) and the *arity* (or number of *ports*) associated with each. The *link graph* on the other hand is a hypergraph relating zero or more ports belonging to nodes of the place graph together with (possibly) names constituting the inner and outer *faces* of the bigraph. Names on the inner face (conventionally drawn below the diagrammatic representation) provide a connection to the link graph of a bigraph plugging into the sites of the place graph; those on the outer face link into any context into which the bigraph itself may be plugged.

Thus a given bigraph can be viewed as an adaptor mediating the fit of a bigraph with as many *regions* (roots of its place-graph forest) as this one has sites, and with the set of names on its outer face equal to those on the inner face of the one in question, into a context with as many sites as this one has regions and inner name-set equal to our outer. For those so inclined, it can be regarded as an arrow from its inner interface[1] to its outer interface[2] in a suitable category, and composition of such arrows constitutes such a plug-in operation. Both places and links may be required to obey a sorting discipline, restricting the control types which may be immediate children of one another, or for instance restricting each link to joining at most one "driving" port; such well-formedness rules will restrict the number of compositions that are possible, in comparison with amongst unsorted bigraphs.

A bigraph provides only a (potentially graphical) representation of a state of a system, like a term in a traditional process algebra; it does not in itself define any notion of evolution such as to give rise to an operational semantics. This is presented in the form of a number of reaction rules, where a bigraphical *redex* is associated with a *reactum* into which any subgraph matching the redex may be transformed. The precise meaning of "match" is defined in categorical terms, in that there must exist contexts for both the current bigraph and the redex, such that the two in their respective contexts are equal. The result of the reaction is the reactum placed in the context required for the redex. For any well-behaved reaction system this last composition will be well defined, and one might hope that the context of the source bigraph could be stripped off the result to reflect a self-contained rewrite, but this is not part of the definition.

Place graphs have to play a dual role: they may indeed represent physical location, and so juxtaposition (and parallel execution) of their children; but they are also the only structuring mechanism available, to represent possession of data or the continuation of a process beyond some guarding action. Thus it may be necessary to distinguish *active* and *inactive* control types, and to restrict matching of a redex to contexts where all its parents are active, lest the second term of a sequential composition, for example, should be reduced (and so effectively allowed to execute) before the first has successfully terminated.

Similarly links might be expected to be used to model events or channels, and indeed one can formulate process algebras within this framework in such a way that they do, but they can also be used to represent more abstract relations between nodes, such as joint possession of some "secret" data value which it would be inconvenient to model as a distinguished type of control which two agents would have to "possess". Note that links can relate multiple nodes, like events in CSP, and are not restricted to point-to-point connections. Thus they can also represent "barriers" in the occam-π sense.

[1] The pair of the number of sites and the inner name-set.
[2] The pair of the number of regions and the outer name-set.

1.1 Simple Examples

Figure 1 shows perhaps the simplest useful ambient-like reaction rule on place graphs: an atomic agent *a* who is adjacent to a room *r* might invoke this rule to move inside the room. Even here there is more than may meet the eye: we are in fact defining a parametric family of reaction rules, parameterised not only by the identities *a* and *r*, but also by the existing contents of the room (represented graphically by the pair of dots). There is also some abuse of notation involved, since strictly controls ought to have a fixed number of children (their *rank*), and here we have changed the number of nodes both inside and immediately outside the room; but there is some punning which lets us get away with such flexibility; note that the context that matches the redex against (a context of) the source bigraph must also be flexible in this way, since the children of the node where it plugs in reduce in number.

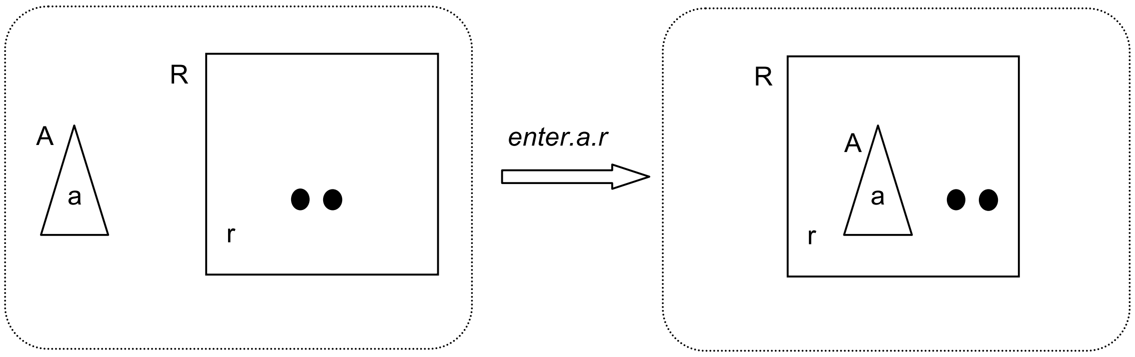

Figure 1

Figure 2 shows a similar transformation, but here a secure room *x* will only admit a user who carries a badge with the correct authorisation. In this version (and so at this level of abstraction) we are probably treating the badge as a passive token, and the *U* control should probably be defined to be inactive; but in a slightly richer model it might be that it had an expiry date and an appropriate timeout reaction would unlink it from the *unlocks.x* attribute (though not necessarily from other *unlocks* permissions), and in this case *U* should be active to allow this to proceed autonomously.

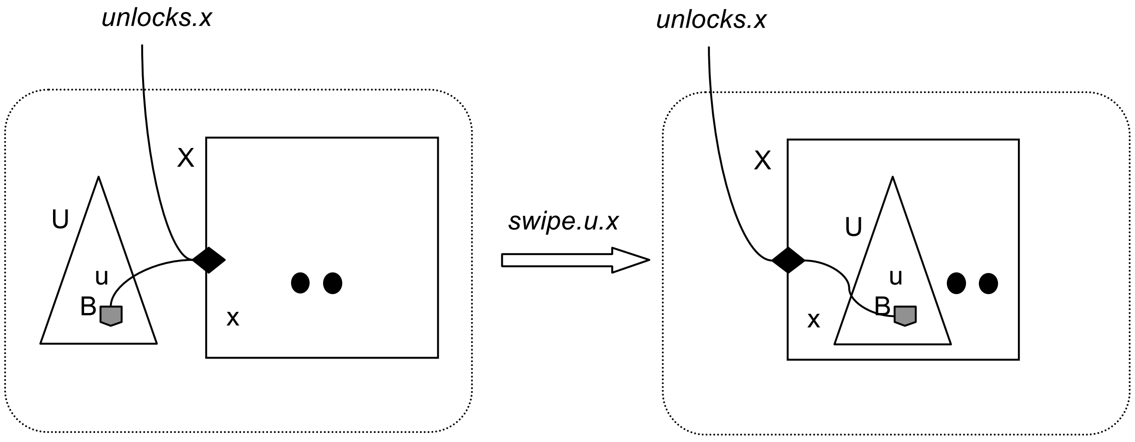

Figure 2

So far all the reactions have preserved the number and kind of nodes in the graph, and merely restructured the place graph. That certainly need not be the case.

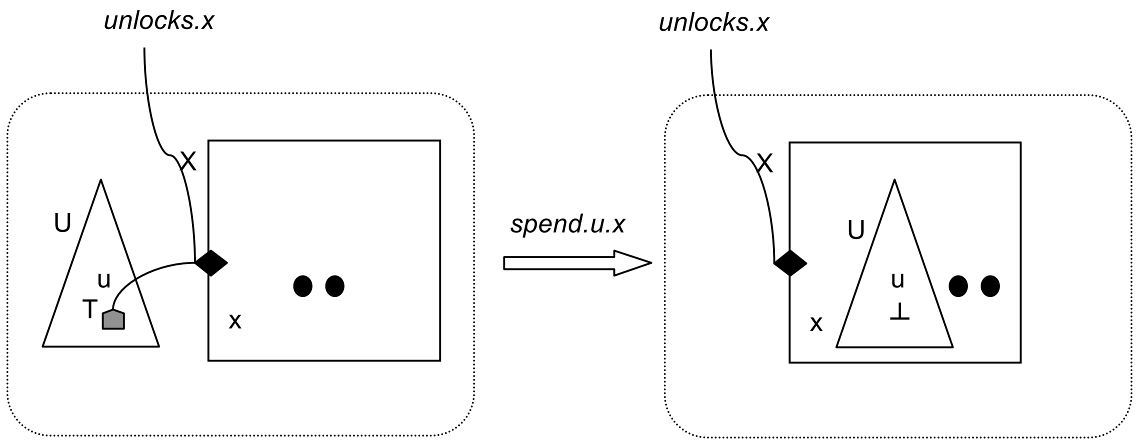

Figure 3

For example, in Figure 3, rather than a long-term badge, *u* holds a single-use token – at least until he uses it in order to enter *x*, at which point it simply vanishes, leaving him with an empty pocket.

As a final example, consider communication in a language like occam: two processes, possibly in widely separated parts of a graph, can interact over a channel to resolve an alternative construct where the input end is guarding one of the branches (Figure 4).

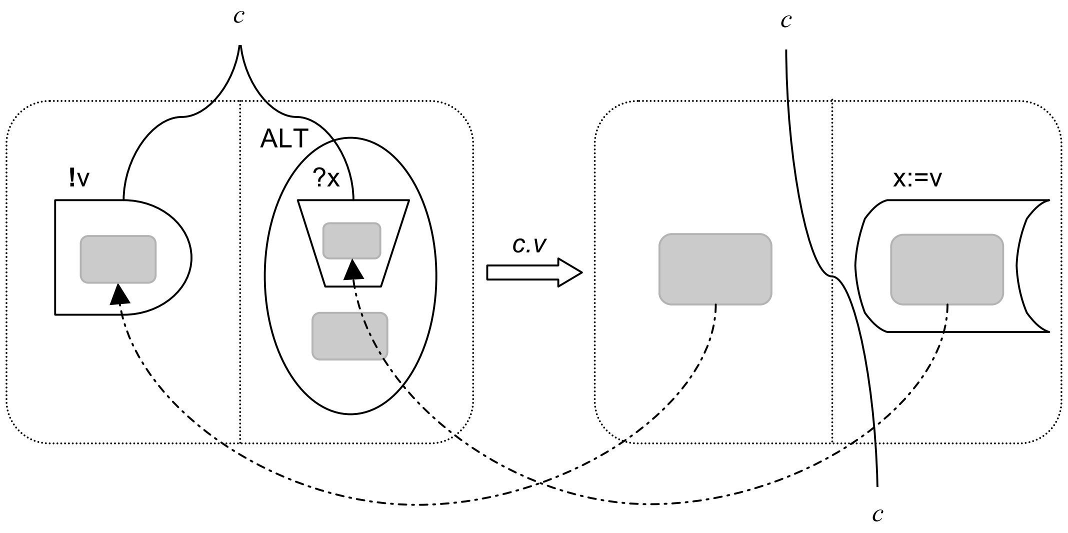

Figure 4

Note that the other alternatives are discarded from the *ALT*, that there remains an obligation to complete the data transfer on the right-hand side, and that the channel *c* remains available for connection of either or both of the components with each other or the other's (sequential) context.

2. Challenges

2.1 Semantics

The bigraph framework is extremely flexible and powerfully expressive; but with great power comes great responsibility! The category-theoretical semantics is quite rebarbative, at least to a CSP-ist, and much of the theory seems aimed at finding restrictions on a given bigraphical reactive system to ensure that the resulting transition system can be trimmed to make it tractable, while preserving its discrimination power (up to bisimulation).

The labelled transition system naturally derived from a set of reaction rules over some set of ground bigraphs has, as its labels, the context into which the source bigraph must be placed in order to be unified with a (unspecified) redex itself in some (unspecified) context. This results in a system where both the labels and the results of the transition are infinite in number and arbitrarily complex in structure; this is clearly undesirable for any practical purpose. One set of conditions is designed to ensure that the essentials of this transition system are captured when attention is restricted to the *minimal* contexts enabling the rule to fire; another allows attention to be restricted further to *prime engaged* transitions (see [12]).

But even here I find something unsatisfactory: the label describes only what needs to be done to the source bigraph to allow it to engage in some reaction. There is no immediate way to determine by which reaction rule, nor where in that composed process, the rewrite occurs. For instance, any reductions that can take place internally to the current state give rise to a transition labelled with the null (identity) context. In some ways this is quite natural, by analogy with the internal τ-actions of the operational semantics of CCS or CSP, but my instinct is that it will often be of crucial interest which reactions gave rise to those essentially invisible events. In many cases it can be contrived (by subtle use of link graphs) that the minimal or prime engaged systems do agree with a more intuitive labelling scheme (such as might arise from the reaction names decorating the arrows in the above examples). But it remains far from clear to me whether there is any systematic way of attaching an intuitive scheme to (some subclass of) reactive systems or of contriving a reactive system whose natural semantics are equivalent to one conceived from a given intuitive labelling.

Of course, given my background, it is not really bisimilarity that interests me – I want to be able to encode interesting specifications and check for refinement using FDR [13]. At least the failures pre-order is known to be monotonic over well-behaved bigraphs (and failures equivalence is a congruence), so there is no fundamental problem here; but there remains a certain amount of research needed to attain fluency and to map onto FDR.

2.2 Applications

There are some areas where a tractable bigraphical treatment offers immediate benefit over the sometimes convoluted modelling needed in CSP, say, to achieve the same effect: the interface between physical and electronic security (of which the examples above are simplistic illustrations), pervasive adaptive systems, location-based and context-aware services, vehicle telematics, and so on. The security application has already received welcome attention in the form of Blackwell's Spygraphs [14,15], which adds cryptographic primitives in the style of the Spi Calculus [16].

What others are there? Do they exhibit particular challenges or opportunities?

2.3 Language Features

The encryption and decryption schemas for symmetric and asymmetric cryptography lend themselves quite naturally to a bigraphical description with nodes representing data items –

see Figure 5. (Note however that this reaction rule is not *nice*, in the technical sense, because it duplicates the contents of the site in the redex into the reactum, and so is not *affine*.)

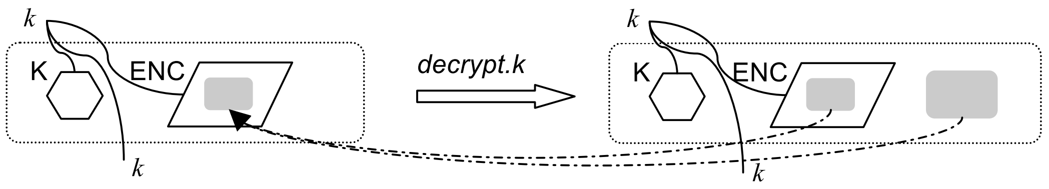

Figure 5

But are there any other data-processing features which could usefully be described in this way, possibly without the adjacency requirement? Perhaps external correlation attacks against putatively anonymised databases?

More generally, suppose I describe some pervasive-adaptive system, say, and establish that the design has desirable properties; some of the agents in the system will describe people and other parts of the environment, but others will be components which need to be implemented. What is the language most appropriate for programming such components, so that we can attain a reasonable assurance that they will behave as their models do in all important aspects? Is it something like occam-π, or do we need new language features or whole new paradigms to reflect the potentially quite drastic reconfigurations that a reaction can produce? A challenge for the language designers out there!

References

[1] Inmos Ltd, occam *Programming Manual*, Prentice Hall, 1984.
[2] R. Milner, *A Calculus of Communicating Systems*, Springer Lecture Notes in Computer Science **92**, 1980.
[3] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice Hall, 1985.
[4] A.W. Roscoe, *The Theory and Practice of Concurrency*, Prentice-Hall, 1998.
[5] R. Milner, *Communicating and Mobile Systems: the Pi-Calculus*, Cambridge University Press, 1999.
[6] D. Sangiorgi and D. Walker, *The π-calculus: A Theory of Mobile Processes*, Cambridge University Press, 2001.
[7] JCSP home page. **http://www.cs.kent.ac.uk/projects/ofa/jcsp/**
[8] P.H. Welch, An occam-pi Quick Reference, 1996-2008, **https://www.cs.kent.ac.uk/research/groups/sys/wiki/OccamPiReference**
[9] B. Sufrin, *Communicating Scala Objects*, Communicating Process Algebras 2008, pp. 35-54, IOS Press, ISBN 978-1-58603-907-3, 2008.
[10] N.C.C. Brown, *Communicating Haskell Processes: Composable Explicit Concurrency using Monads*, Communicating Process Architectures 2008, pp. 67-83, IOS Press, ISBN 978-1-58603-907-3, 2008.
[11] L. Cardelli and A.D. Gordon, *Mobile Ambients*, in Foundations of System Specification and Computational Structures, Springer Lecture Notes in Computer Science **1378**, pp.140-155, 2000.
[12] R. Milner, *The Space and Motion of Communicating Agents*, Cambridge University Press, 2009.
[13] Formal Systems (Europe) Ltd: *Failures-Divergence Refinement: the FDR2 manual*, **http://www.fsel.com/fdr2_manual.html**, 1997-2007.
[14] C. Blackwell: *Spygraphs: A Calculus For Security Modelling*, in Proceedings of the British Colloquium for Theoretical Computer Science – BCTCS, 2008.
[15] C. Blackwell: *A Security Architecture to Model Destructive Insider Attacks*, in Proceedings of the 8th European Conference on Information Warfare and Security, 2009.
[16] M. Abadi and A.D. Gordon: *A calculus for cryptographic protocols: The Spi Calculus*, in Proceedings of the Fourth ACM Conference on Computer and Communications Security, pp.36-47, 1997.