Global Synchronization in a Mobile Context
Progress Report

Andrew Hughes

March 29, 2006

Contents

Chapter 1

Activity Summary

Below is a synopsis of activities taking place over the last six months, commencing with registration as a postgraduate student in the Department of Computer Science back in October 2005.

1.1 Research

The aim of this research is the construction of a process calculus which combines the notions of discrete time and mobility. Earlier work on an undergraduate project developing semantics for the Cashews[1][42] language involved the use of the CaSE[2] process calculus (and later, the conservative extension Cashew-Nuts). It became clear during the project that it would be interesting to further extend this calculus with the notion of mobility, and this postgraduate work focuses on this.

At the time, mobility was taken to refer to the kind of mobility exhibited by the π calculus (this being the only form considered at that point), but further research over the past six months has shown that this idea can be realised in two different ways, outlined below.

The focus thus far has largely concerned the reading of existing papers, theses and other publications which relate to the chosen area. A bibliography of these is included. From the initially rather vague ideas of six months ago, this subsequent research has resulted in the following three main concepts presenting themselves as central to the proposed calculus:

1. Timing / Global synchronization as in TPL[27], PMC[5], CSA[20] and CaSE[43]

2. Localities [18]

[1] Cashews is a language for web service composition, initially based on OWL-S.

[2] A conservative extension of CCS [34], which adds the notion of multiple discrete time clocks

3. Mobility in the form of migration between localities and maybe also in the form of scope mobility as in the π calculus [39].

We now consider these in more detail.

1.1.1 Timing

TPL is the first in a series of calculi which extends CCS with discrete time. This is not time as we may naively consider it; as a value which can be divided into units such as minutes and seconds. Instead, a discrete time calculus is used for synchronization. TPL exhibits a phenomenon known as *maximal progress*; a clock ticks after all internal actions (τs) have been completed. Thus, a clock's tick is essentially a marker which identifies when all work is complete.

These clock ticks also enable timeouts to be added to the calculus. For example, take $[E]\sigma(F)$ where E and F are both processes and σ is a clock. There are two possible derivations. Firstly, E may act. However, if E can not act before the clock σ ticks, then F will act. In short, F acts if E times out on the clock σ.

These features are advantageous for multi-party synchronization. In CCS, synchronization works well for two processes communicating via corresponding input and output channels. However, this communication doesn't scale well. With more than two processes, a sequence of multiple synchronizations must be enacted. More importantly, if the number of processes changes, this series must be recreated. Thus, multi-party synchronization under CCS is *non-compositional*. In discrete timed calculi, the clock can be used as an external agent to synchronise an arbitrary number of processes. The clock will be stalled by the internal synchronisations taking place, so when it does tick, this indicates that all such actions have taken place.

The later calculi (PMC, CSA and CaSE) extend these notions further. Primarily, all have multiple clocks, whereas TPL has only the single clock, σ. However, only CaSE allows the use of these clocks to be segmented via *clock hiding*. Essentially, a system of processes can be subdivided into components, each of which can contain clocks. These are then hidden from other components (appearing merely as silent actions) to create a hierarchy of clocks. For instance, in $((E|F)/\sigma)|G$, E, F and G are processes and σ is again the clock. However, the ticks of σ can only be observed by E and F. They are hidden from R by the hiding operator, $/$, and only silent actions are observed. This idea of encapsulation is one of the central concepts of CaSE.

1.1.2 Localisation

One concept within the area of algebraic process calculi is the idea of *localisation*. This effectively adds another level of grouping to the calculus, one above that of the process, allowing a set of composed processes to be contained within one *locality*. One use of this is the modelling of *distribution* in the calculus. Localities make it possible to distinguish behaviour based on the number of

processes involved, by assigning each process to a locality. This distinction is used in relating processes via *location bisimulation* [10].

This idea of grouping, which can be taken to its logical extent by forming a hierarchy of localities, has echos of the notion of *clock hiding* within CaSE described above. Our first idea is to combine these two hierarchical concepts, so that a locality also acts as a method of hiding clocks. As a result, the clock hiding operator from CaSE disappears and instead we add in localities, the bounds of which define both a new group and the hiding of the clocks within that group.

Recall the example of clock hiding above. We can turn $((E|F)/\sigma)|G$ into $(l[E|F]_{\{\sigma\}})|G$[3] by a change in syntax. Instead of having an operator, $/$, with scope over a set of processes for clock hiding, we assume that the clock σ is hidden outside the locality, l. Our first task will be to formalise this idea further, but our naive intuition is that the only change is the creation of an association between a name (that of the locality) and a particular instance of clock hiding.

With this in place, we can obtain the set of visible clocks in a particular locality by taking the union of the set of clocks associated with that locality and the sets of the parent localities. For example, in $n[E|m[F|l[G]_{\{\sigma\}}]_{\{\rho\}}]_{\{\}}$, G in locality l can observe the ticks of both σ and ρ. F can only see ρ's ticks, while E can only observe silent actions resulting from the two hidden clocks. Taking this further, we can assume that the clock context, the set of clocks within the system, is equal to the union of the sets of clocks associated with each locality.

1.1.3 Migration

One common use of localities, beyond equivalence theory, is as a form of mobility. As localities effectively group processes (and in our case, clocks as well), moving them (usually called migration) results in their contents also moving to a new location.

One well-known calculus that exhibits this is the ambient calculus [13]. Within this, localities are known as *ambients* and processes can perform one of three *capabilities*: *in* n, *out* n and *open* n. The first two act subjectively, moving the ambient containing the process inside or outside the specified ambient, n, respectively. The *open* capability dissolves n and is performed by a process in the parent ambient. This allows the contents of the ambient to enter the parent and interact with its existing occupants.

The next logical step will be to add something like this to the combination of CaSE with localities. This allows migration to take place without having to introduce a form of value passing and its additional complications (see 1.1.4); we can instead retain the simplicity of the signals used by CaSE.

Communication within the ambient calculus is provided via *open*, which seems implicitly a dangerous and ugly way of doing so and has been criticised in later papers [11]. With the ideas from CCS and CaSE present in the hypothetical

[3]Note that we add names to our localities here. This is not really necessary at this stage; they provide nothing more than a way to refer to localities in talking about a system. However, they are necessary for providing migration as discussed in 1.1.3

calculus we have described so far, we already have a form of communication that has been extensively considered by other researchers. Instead of using *open*, we can thus extend our existing communication primitives in the style of the Seal calculus [67], allowing communication to be directed to channels in parent or child localities as well as to siblings in the same locality. This is the form also adopted by the Boxed Ambients calculus, which arose directly through criticism of the *open* directive previously mentioned.

At this point, we already have an outline for an interesting new calculus with the original notions we desired and presumably also conservative of both the ideas in CaSE (simply a system sans mobility primitives) and CCS (the same, but also with an empty clock context). This seems to be a reasonable initial goal for this research.

However, adding the notion of value passing (and its generalisation to names, as in the π calculus) could provide a wider range of functionality.

1.1.4 Value Passing

To add value passing, we can simply take the same route as CCS and the π calculus, with inputs binding variables, just as restriction does. We can either just add in values, or we can generalise and pass names[4] as in the π calculus, which allows the scope extrusion/intrusion form of mobility. With this, as results for the π calculus show [37], we can send and receive tuples of values, as the polyadic variant is encodable by the monadic. We can also go as far as to pass processes, as seen in higher-order variants of the π calculus [50, 60]. Neither polyadic nor higher-order forms of the calculus add further computational power.

Value passing adds an extra layer of complexity to reasoning over the calculus. For example, equivalence theories have to take into account the binding of variables, so one of the forms associated with the π calculus (such as barbed bisimulation [50]) would be required. In contrast, it seems a reasonable presumption that temporal observation equivalence, as used in CaSE, could be extended to the more minimal calculus described above. Value passing also makes proving termination more complex [53].

There are obviously advantages to adding this, otherwise it would seem pointless to suggest such a direction for the calculus. In the lighter calculus, processes and clocks can only be moved as part of the larger locality grouping. With value passing, values (and, by extension, processes, clocks and localities) could also be passed between processes, allowing a greater degree of granularity to the mobility. The idea of values and scoping also links the calculus more to the familiar concepts of the λ calculus and programming languages, which exhibit these.

This also raises the interesting notion that the clock signal could be used to broadcast a value. This could be further extended to allow processes to submit values to the clock for broadcast.

[4]These names may be just a generalisation of values and channels, as in the π calculus, but they may also include clocks, localities and processes

A final step, from the theoretical side, would be to also type the calculus; having a varied range of notions, including names, processes, clocks and localities, would seem to make this a sensible idea. At the simplest level, it could disallow any non-sensical constructions which are allowed by the syntax alone. Certainly, type systems have been used in other distributed calculi [49, 12, 16, 33]

1.1.5 Using The Calculus

Some obvious case studies present themselves for the calculus we outline above. One is web service composition, as we have already been involved in research in this area. The advantages of simply encoding our existing semantics[5] using this new calculus would be twofold; firstly, it would provide evidence that it still has the same usability for this scenario as CaSE, and secondly, the new features could be tested within this arena.

Biology also seems a likely candidate, as this is a common use for the ambient calculus. In this area, P systems [45] appear interesting as they have a lot of similarities with the ambient calculus and their notion of operation (global synchronous rule execution with mobility between cell membranes) would seem to suggest that giving a semantics within our calculus would not be unreasonable.

Another area of interest is the programming domain. It seems common, in the literature, for distributed calculi to evolve into a language. For example, the join calculus also has an associated programming language [25], and Pict[63] evolved from the π calculus (and further into Nomadic Pict [68]). There is also a recent development called PiDuce [14], which combines the π calculus with the web service domain. Doing something in this vein could prove interesting.

As both the ambient calculus and the π calculus are Turing complete, it seems reasonable to suggest that the resulting calculus would also be, and thus a language would be an interesting further development (and typing, as mentioned above, would further fit with this scenario).

1.1.6 Conclusions

Over the course of this discussion, we have developed a number of ideas for our proposed calculus:

- Adding localities and merging them with clock hiding
- Allowing localities to be migrated as in the Ambient calculus
- Moving from signal-based communication to value-passing
- Typing the calculus which results from the addition of the above.

[5]Note that their use of Cashew Nuts may present a problem here. There are two possible solutions to this; either a variant of the existing semantics can be derived which don't use the Cashew Nuts extensions (possible as a large part of their development took place without these) or the same changes could be applied to our calculus as were applied to CaSE

We have also clearly identified a number of case studies where the functionality of the resulting calculus may be useful:

- Web service composition (specifically in the context of Cashews)
- Biology (specifically relating to P systems)
- Programming (where the calculus is used as the formal basis of a language)

These ideas are currently just that. The next stage of this research will involve sketching these out more clearly, and formally defining the syntax and semantics of these proposed extensions. This is detailed further in chapter 2.

1.2 RTP

Thirty credits of work has been undertaken, over three RTP modules, which are as follows:

1. **GSC6100 – Library and Information Skills for Successful Research** This module involves the completion of a series of research tasks, including the development of a literature review and bibliography. Some of the material provided here will form the basis of the requirements for this module.
2. **GSC6310 – Personal and Professional Skills Development 2**: This module gives credit to a number of tasks performed as a postgraduate student which aren't directly related to the research being undertaken. This includes credits for participation in group activities, such as seminars, demonstrating work, paper publication and administrative duties such as personal website maintenance.
3. **COM6890 – Theory of Distributed Systems – Exemption**: An exemption from this module has been agreed with Mike Stannett, due to its successful completion as an undergraduate at an appropriate level for consideration here.

1.3 Talks

The following talks have been given over the last six months, copies of which are available at http://www.dcs.shef.ac.uk:

1. **Combining Timing, Localities and Migration in a Process Calculus**: Delivered at the VT group seminar on the 17th of February, 2006.
2. **Process Algebras With Localities**: Delivered to the Theory Special Interest Group on the 20th of January, 2006.

3. **Co-monads: Musings on 'Signals and Co-monads' by Tarmo Uustalu and Varmo Vene**: Delivered to the Theory SIG on the 28th of October, 2005.

The material presented here will form the basis of a talk delivered to the British Colloquium on Theoretical Computer Science (BCTCS))on the 5th of April (and to the Theory SIG beforehand, on the 31st of March).

1.4 External Activities

The following activities have taken place outside the department:

1. **19th of December, 2005**: Attendance at the Christmas seminar at Nottingham University, where a series of talks were presented.
2. **4th - 7th of April, 2006**: Attendance at BCTCS in Swansea.
3. **8th - 12th of April, 2006**: Attendance at the Midland Graduate School (MGS) in Leicester.
4. **18th - 21st of April, 2006**: Attendance at Types/Trends in Functional Programming 2006 in Nottingham.

1.5 Demonstrating

The following demonstrating duties have been performed within the department:

1. **COM1030: Requirements Engineering - Autumn Semester 2005**:
   - Helping students in tutorial classes in the Lewin Lab.
   - Marking stages 1 and 2 of the 'Crossover' project.
2. **COM2010: - Autumn Semester 2005**:
   - Marking Haskell-based assignments.
3. **COM2060: Database Systems - Autumn Semester 2005**:
   - Helping students in large group (lecture-hall size) tutorial classes.
4. **COM1040: Systems Analysis and Design - Spring Semester 2006**:
   - Helping students in tutorial classes in the Lewin Lab.
   - Marking stages 3, 4 and 5 of the 'Crossover' project.
5. **COM1090: Computer Architectures - Spring Semester 2006**:
   - Helping students in tutorial classes in the Lewin Lab.

6. **COM2020: Abstract Data Types - Spring Semester 2006**:
    - Marking student assignments.
7. **COM3170: Distributed Systems - Spring Semester 2006**:
    - Answering student questions using the WebCT system
    - Invigilating the quizzes taken by the students
    - Marking assignments.

1.6 Administrative Duties

The following administrative duties have been performed within the department:

1. **VT Research Lab Management – November 2005 on**: This has involved ensuring the provision of sufficient resources within the lab for new and existing students, in addition to general problem solving as issues arise.
2. **Organising and Chairing the Theory SIG meetings every Friday – January 2006 on**: This has involved timetabling various speakers and other activities for the session, as well as (attempting to) co-ordinate the meeting itself. Lately, this has also included the shameless promotion of the session via posters and a regular slot on the plasma screen.
3. **Instigation and Organisation of the Concurrency Reading Group every Tuesday – March 2006 on**: This new group was formed to stimulate interest in the subject of concurrency within the department, and to hopefully cover a wider variety of research material than is possible within the scope of a single postgraduate study. Again, this is also promoted via the use of posters and the plasma screen.

Chapter 2

Future Plans

The previous section outlined a series of ideas for creating the proposed calculus, in roughly the intended order in which these tasks will be attempted. Here, we simply expound on these a little more, giving more detail as to the plans for this research over the next two and a half years. Note that this plan will vary over time, and there is no guarantee that all tasks will be completed within the scope of this research.

2.1 Short Term

- Complete the literature review (see the draft provided) to such a stage that it can be submitted, along with the other requirements, for **GSC6100**.
- Further extend the literature review so that it forms part of a transfer report in time for the next panel meeting (late September 2006).
- Consider further the composition of clock hiding and localities and formulate a paper around this localised extension to CaSE. Formalise a syntax and semantics.
- Use this extension to represent the Cashew semantics in localised form.

2.2 Medium Term

- Take localised CaSE and extend it with mobility primitives, thus creating the mobile timed calculus originally prophesied. Attempt to achieve the same formalisations as for the merely localised variant.
- Again, apply this to Cashew, particularly so as to observe what new features can be modelled by mobility.
- Look at this new localised migratory calculus in the context of P systems. Can a reasonable semantics be provided?

2.3 Long Term

- If we reach this stage, consider either adding value passing or typing the calculus, depending on which seems most relevant. Both may also be an option.
- Formulate a programming language around the resulting calculus.

Bibliography

[1] Abadi, M., and Gordon, A. D. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the Fourth ACM Conference on Computer and Communications Security* (1997), ACM Press, pp. 36–47.

[2] Aceto, L., and Murphy, D. Timing and causality in process algebra. *Acta Informatica 33*, 4 (June 1996), 317–350.

[3] Amadio, R. An asynchronous model of locality, failure and process mobility. In *COORDINATION 97* (1997), no. 1282 in LNCS, Springer-Verlag.

[4] Amadio, R., and Prasad, S. Localities and failures. In *Foundation of Software Technology and Theoretical Computer Science: 14th Conference (FST-TCS '94)* (December 1994), no. 880 in LNCS, Springer-Verlag, pp. 205–216.

[5] Andersen, H. R., and Mendler, M. An asynchronous process algebra with multiple clocks. In *Proceedings of the 5th European Symposium on Programming (ESOP '94)* (April 1994), no. 788 in LNCS, Springer-Verlag, pp. 58–73.

[6] Beaten, J. C. M., and Middelburg, C. A. Process algebra with timing: Real time and discrete time. In *Handbook of Process Algebra*. North-Holland, 2001, ch. 10, pp. 627–684.

[7] Bergstra, J. A., and Klop, J. W. Process algebra for synchronous communication. *Information and Control 60* (1984), 109–137.

[8] Berry, G., and Boudol, G. The chemical abstract machine. *Theoretical Computer Science 96* (1992), 217–248.

[9] Boudol, G., Castellani, I., Germain, F., and Lacoste, M. Models of distribution and mobility: State of the art. MIKADO Global Computing Project Deliverable D1.1.1, 2002. INRIA and France Telecom Research and Development.

[10] Boudol, G., Castellani, I., Hennessy, M., and Kiehn, A. Observing localities. *Theoretical Computer Science 114* (1993), 31–61.

[11] Bugliesli, M., Castagna, G., and Crafa, S. Boxed ambients. In *Theoretical Aspects of Computer Software: 4th International Symposium (TACS '01)* (2001), N. Kobayashi and B.C.Pierce, Eds., vol. 2215 of *LNCS*, Springer-Verlag, pp. 38–63.

[12] Cardelli, L., Ghelli, G., and Gordon, A. D. Mobility types for mobile ambients. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming (ICALP '99)* (1999), no. 1644 in LNCS, Springer-Verlag, pp. 230–239.

[13] Cardelli, L., and Gordon, A. D. Mobile ambients. In *Proceedings of the 1st International Conference on Foundations of Software Science and Computation Structures (FoSSaCS '98)* (1998), vol. 1378 of *LNCS*, Springer-Verlag.

[14] Carpineti, S., Laneve, C., and Padovani, L. Piduce – a project for experimenting web service technologies. Submitted to Logic and Algebraic Programming, January 2006.

[15] Carriero, N., and Gelernter, D. Linda in context. *Communications of the ACM 32*, 4 (April 1989), 444–458.

[16] Castagna, G., Ghelli, G., and Nardelli, F. Z. Typing mobility in the seal calculus. In *Proceedings of the 12th International Conference on Concurreny Theory (CONCUR '01)* (2001), K. Larsen and M. Nielsen, Eds., no. 2154 in LNCS, Springer-Verlag, pp. 82–101.

[17] Castagna, G., and Nardelli, F. Z. The seal calculus revisited: Contextual equivalence and bisimilarity. In *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS '02)* (2002), no. 2556 in LNCS, Springer-Verlag, pp. 85–96.

[18] Castellani, I. Process algebras with localities. In *Handbook of Process Algebra*. North-Holland, 2001, ch. 15, pp. 945–1046.

[19] Church, A. The calculi of lambda-conversion. *Annals of Mathematical Studies 6* (1941).

[20] Cleaveland, R., Lüttgen, G., and Mendler, M. An algebraic theory of multiple clocks. In *Proceedings of the 8th International Conference on Concurreny Theory (CONCUR '97)* (1997), no. 1243 in LNCS, Springer-Verlag, pp. 166–180.

[21] Coppo, M., Dezani-Ciancaglini, M., Giovannetti, E., and Salvo, I. m^3: Mobility types for mobile processes in mobile ambients. In *Proceedings of Computing: the Australasian Theory Symposium (CATS '03)* (April 2003), vol. 78 of *Electronic Notes in Theoretical Computer Science*, Elsevier, pp. 1–34.

[22] Dijkstra, E. W. Hierarchical ordering of sequential processes. *Acta Informatica 1*, 2 (June 1971), 115–138.

[23] Fournet, C., and Gonthier, G. The reflexive chemical abstract machine and the join-calculus. In *Proceedings of the 23rd. Annual Symposium on Principles of Programming Languages (POPL '96)* (1996), pp. 372–385.

[24] Fournet, C., Gonthier, G., Lévy, J.-J., Maranget, L., and D.Rémy. A calculus of mobile agents. In *Proceedings of the 7th International Conference on Concurreny Theory (CONCUR '96)* (1996), no. 1119 in LNCS, Springer-Verlag.

[25] Fournet, C., and Maranget, L. *The Join Calculus Language*, September 2000.

[26] Godskesen, J. C., Hildebrandt, T., and Sassone, V. A calculus of mobile resources. In *Proceedings of the 13th International Conference on Concurreny Theory (CONCUR '02)* (2002), no. 2421 in LNCS, Springer-Verlag, pp. 272–287.

[27] Hennessy, M., and Regan, T. A process algebra for timed systems. *Information and Computation 117*, 2 (March 1995), 221–239.

[28] Hewitt, C., Bishop, P., and Steiger, R. A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence* (1973), pp. 235–245.

[29] Hoare, C. A. R. Communicating sequential processes. *Communications of the ACM 21*, 8 (August 1978), 666–677.

[30] Lee, I., Philippou, A., and Sogolsky, O. A family of resource-bound real-time process algebras. In *Short Contributions from the Workshop on Algebraic Process Calculi: The First Twenty Five Years and Beyond* (2005), vol. NS-05-03 of *BRICS Notes*, pp. 151–154.

[31] Lee, J. Y., and Zic, J. On modeling real-time mobile processes. In *Proceedings of the twenty-fifth Australasian Conference on Computer science* (2002), vol. 17 of *Conferences in Research and Practice in Information Technology Series*, pp. 139–147.

[32] Mazurkiewicz, A. Concurrent program schemes and their interpretations. Tech. Rep. AIM PB-78, Computer Science Department, University of Aarhus, 1977.

[33] Merro, M., and Sassone, V. Typing and subtyping mobility in boxed ambients. In *Proceedings of the 13th International Conference on Concurreny Theory (CONCUR '02)* (2002), L. B. et. al., Ed., no. 2421 in LNCS, Springer-Verlag, pp. 304–320.

[34] Milner, R. *Communication and Concurrency*. Prentice-Hall, 1989.

[35] Milner, R. Functions as processes. *Mathematical Structures in Computer Science 2*, 2 (1992), 119–141.

[36] Milner, R. Elements of interaction: Turing award lecture. *Communications of the ACM 36*, 1 (January 1993), 78–89.

[37] Milner, R. The polyadic π-calculus: a tutorial. In *Logic and Algebra of Specification*, F. L. Bauer, W. Brauer, and H. Schwichtenberg, Eds. Springer-Verlag, 1993, pp. 203–246.

[38] Milner, R. What's in a name? In *Computer Systems: Theory, Technology and Applications*. Springer-Verlag, December 2003, ch. 28.

[39] Milner, R., Parrow, J., and Walker, D. A calculus of mobile processes, parts I and II. Tech. Rep. ECS-LFCS-89-86, University of Edinburgh, June 1989.

[40] Moller, F., and Tofts, C. A temporal calculus of communicating systems. Tech. Rep. ECS-LFCS-89-104, University of Edinburgh, December 1989.

[41] Nardelli, F. Z. *The Semantics of Higher Order Processes*. PhD thesis, The University of Paris, December 2003.

[42] Norton, B., Foster, S., and Hughes, A. A compositional operational semantics for OWL-S. In *Proceedings of the 2nd International Workshop on Web Services and Formal Methods (WS-FM 2005)* (September 2005), no. 3670 in LNCS, Springer-Verlag, pp. 303–317.

[43] Norton, B., Lüttgen, G., and Mendler, M. A compositional semantic theory for synchronous component-based design. In *Proceedings of the 14th International Conference on Concurreny Theory (CONCUR '03)* (2003), no. 2761 in LNCS, Springer-Verlag.

[44] Parrow, J. An introduction to the *pi* calculus. In *Handbook of Process Algebra*. North-Holland, 2001, ch. 8, pp. 479–543.

[45] Paun, G. Computing with membranes. Tech. Rep. 208, Institute of Mathematics of the Romanian Academy, November 1998.

[46] Paun, G. Computing with membranes. *Journal of Computer and System Sciences 61*, 1 (2000), 108–143.

[47] Petri, C. A. *Kommunikation mit Automaten*. PhD thesis, Institut für Instrumentelle Mathematik, 1962.

[48] Pierce, B. C., and Sangiorgi, D. Typing and sub-typing for mobile processes. *Mathematical Structures in Computer Science 6*, 5 (1996), 409–454.

[49] Riely, J., and Hennessy, M. A typed language for distributed mobile processes. In *Proceedings of Principles of Programming Languages (POPL "98)* (January 1998), ACM Press, pp. 378–390.

[50] Sangiorgi, D. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, The University of Edinburgh, 1993.

[51] Sangiorgi, D. Locality and true-concurrency in calculi for mobile processes. No. 789 in LNCS, Springer-Verlag, pp. 405–424.

[52] Sangiorgi, D. Locality and interleaving semantics in calculi for mobile processes. *Theoretical Computer Science 155*, 1 (1996), 39–89.

[53] Sangiorgi, D. Types, or: Where's the difference between ccs and π? In *Proceedings of the 13th International Conference on Concurreny Theory (CONCUR '02)* (2002), no. 2421 in LNCS, Springer-Verlag, pp. 76–97.

[54] Satoh, I. *Time and Asynchrony in Distributed Computing*. PhD thesis, Keio University, January 1996.

[55] Satoh, I., and Tokoro, M. A timed calculus for distributed objects with clocks. In *Proceedings of the 7th European Conference on Object-Oriented Programming (ECOOP '93)* (July 1993), no. 707 in LNCS, Springer-Verlag, pp. 326–345.

[56] Sewell, P. Applied *pi* – a brief tutorial. Tech. Rep. 498, University of Cambridge, 2000.

[57] Sewell, P., Leifer, J. J., Wansbrough, K., Nardelli, F. Z., Allen-Williams, M., Habouzit, P., and Vafeiadis, V. Acute: High-level programming language design for distributed computation. In *Proceedings of the 10th Annual ACM SIGPLAN International Conference on Functional Programming (ICFP '05)* (September 2005), ACM Press.

[58] Sewell, P., Wojciechowski, P. T., and Pierce, B. C. Location-independent communication for mobile agents: A two-level architecture. In *Internet Programming Languages: ICCL'98 Workshop* (1999), vol. 1686 of *LNCS*, Springer-Verlag, pp. 1–32.

[59] Stefani, J.-B. A calculus of kells. In *Proceedings of the 2nd European Association for Theoretical Computer Science International Workshop on Foundations of Global Computing (FGC '03)* (2003), vol. 85 of *Electronic Notes in Theoretical Computer Science*, Elsevier, pp. 40–60.

[60] Thomsen, B. A calculus of higher order communicating systems. In *Proceedings of the 16th. Annual Symposium on Principles of Programming Languages (POPL '89)* (January 1989), ACM Press.

[61] Thomsen, B., and Leth, L. Some facile chemistry. Tech. Rep. ECRC-92-14, European Computer-Industry Research Centre, Munich, May 1992.

[62] Turing, A. M. On computable numbers, with an application to the entscheidungsproblem. In *Proceedings of the London Mathematical Society* (1936), vol. 2, pp. 230–265.

[63] Turner, D. N. *The Polymorphic Pi-calculus: Theory and Implementation*. PhD thesis, The University of Edinburgh, 1996.

[64] Valencia, F. D. Concurrency, time and constraints. In *Proceedings of the 19th International Conference on Logic Programming (ICLP '03)* (2003), no. 2916 in LNCS, Springer-Verlag, pp. 72–101.

[65] van Glabbeek, R. J. The linear time-branching time spectrum i: The semantics of concrete, sequential processes. In *Handbook of Process Algebra*. North-Holland, 2001, ch. 8, pp. 3–99.

[66] Victor, B. *The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes*. PhD thesis, Uppsala University, June 1998.

[67] Vitek, J., and Castagna, G. Seal: A framework for secure mobile computations. In *Proceedings of the ICCL '98 Workshop on Internet Programming Languages* (1999), vol. 1686 of *LNCS*, Springer-Verlag, pp. 47–77.

[68] Wojciechowski, P. T. *Nomadic Pict: Language and Infrastructure Design for Mobile Computation*. PhD thesis, The University of Cambridge, March 2000.

[69] Zilio, S. D. Mobile processes: A commented bibliography. In *Workshop On Modelling and Verification of Parallel Processes* (2001), F. C. et. al., Ed., no. 2067 in LNCS, Springer-Verlag, pp. 206–222.