Nomadic Time
(Extended Abstract)

Andrew Hughes[1]

Department of Computer Science, University of Sheffield,
Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK.
e-mail: a.hughes@dcs.shef.ac.uk

1 Introduction

CCS [1] is commonly used for modelling synchronous communication between two processes, where one sends a signal and the other receives it at the same time (a concept referred to as *local synchronization*). However, it cannot directly represent systems involving synchronization of a sender with an arbitrary number of recipient processes (known as *global synchronization*) in a *compositional* manner. Crucially, the semantics of a broadcast agent cannot suitably be represented using CCS. If the agent is defined as transmitting a signal to each of the recipients sequentially, through multiple local synchronizations, then its semantics will become non-compositional, because such behaviour depends upon the number of recipients. Each time a new recipient is introduced, or one of the existing ones is removed, the semantics will have to be changed.

A solution to this deficiency lies in providing a way of determining when all possible synchronizations have taken place. With this facility available, the broadcast agent can recurse, transmitting signals, until this condition holds. The family of abstract timed process calculi (including TPL[2] and CaSE[3]) allow this by extending CCS with *abstract clocks*. These don't represent real time, with units such as minutes and seconds, but are instead used to form synchronous cycles of internal actions followed by clock ticks. A concept known as *maximal progress* enforces the precedence of internal actions over clock ticks, allowing the possible synchronizations to be monitored. When a synchronization takes place, it appears to the system as an internal action. Thus, with maximal progress, synchronizations prevent the clock from ticking, and a result, the occurrence of a clock tick also indicates that there are no possible synchronizations.

However, the timed calculi mentioned above lack any notion of distribution or mobility. Thus, while they can adequately represent large static systems, involving both local and global synchronization, they fail to model more mobile systems, where the location of a process can change during execution. In contrast, the ambient calculus [4] includes both distribution (via structures known as *ambients*) and mobility (by allowing these structures to be moved, along with their constituent processes, during execution). But, it suffers from similar deficiencies to CCS when modelling global synchronization.

This extended abstract presents the calculus of *Typed Nomadic Time* (TNT), which combines the abstract timed calculus, CaSE, with notions of distribution and mobility from the ambient calculus and its variants ([5,6]). This allows

the creation of a compositional semantics for mobile component-based systems, which utilise the notion of communication between arbitrary numbers of processes within a mobile framework. To extend the example of a broadcast agent given above, this extension allow broadcasts to be localised to a particular group of processes, which can change during execution. Section 2 provides a simple example, illustrating the use of the calculus, while section 3 concludes with a discussion of future work.

2 A Simple Example

Consider the familiar children's game of musical chairs. The conduct of the game can be divided into the following stages:

1. The players begin the game standing. The number of players is initially equal to the number of chairs.
2. The music starts.
3. A chair is removed from the game.
4. The music stops.
5. Each player attempts to obtain a chair.
6. Players that fail to obtain a chair are out of the game.
7. The music restarts. Any players who are still in the game leave their chairs and the next round begins (from stage three).

The winner is the last player left in the game. A model of this game can be created using the TNT process calculus.

The game environment is represented using named locations (commonly known as *localities* in the literature). These localities can be nested within each other and form a forest structure (due to the possibility of multiple localities occurring at the top level). In the musical chairs scenario, each chair is represented by a locality, as is the 'sin bin', to which players are moved when they are no longer in the game. These localities are all nested inside a further locality which represents the room itself. This is not a necessity, but makes for a cleaner solution; it allows multiple instances of the system to be nested inside some larger system, each performing its own internal interactions and entering into the synchronization cycle of the larger system.

The locality structure is represented in the calculus by the expression shown below. The *room* locality contains multiple *chair* localities, each of which contains $\mathbf{0}$, a process with no explicit behaviour[1]. The $|$ operator connecting the chair localities denotes parallel composition; each locality and its constituent processes runs concurrently. CB and the σ and ω symbols will be explained shortly.

$$room[chair[\mathbf{0}]^{CB}_{\emptyset} \mid chair[\mathbf{0}]^{CB}_{\emptyset}]^{\omega}_{\{\sigma\}}. \tag{1}$$

[1] It does exhibit contextual behaviour, due to transitions created by clock ticks.

The players themselves are represented by *processes*. This allows them both to interact and to move between localities. A gamesmaster process is also introduced. This doesn't play an active role in the game itself, but is instead responsible for performing the administrative duties of removing chairs from the game and controlling player movement. The process definitions are summarised in Table 1, along with the derived syntax used in this example.

Table 1. Summary of Processes and Derived Syntax for Musical Chairs

$$\omega \stackrel{\text{def}}{=} \mu X.(\overline{in}.X + \overline{out}.X + \overline{open}.X) \tag{2}$$

$$\sigma.P \stackrel{\text{def}}{=} \lceil \mathbf{0} \rceil \sigma(P) \tag{3}$$

$$CB \stackrel{\text{def}}{=} \mu X.(\overline{in}.\overline{out}.X + \overline{open}) \tag{4}$$

$$SB \stackrel{\text{def}}{=} \mu X.\overline{in}.X \tag{5}$$

$$GM2 \stackrel{\text{def}}{=} \sigma.GM3 \tag{6}$$

$$GM3 \stackrel{\text{def}}{=} open\ chair.GM5 \tag{7}$$

$$GM5 \stackrel{\text{def}}{=} \mu X.(\lceil in\ chair\ sit.X \rceil \sigma(GM6)) \tag{8}$$

$$GM6 \stackrel{\text{def}}{=} \mu X.(\lceil in\ sinbin\ leave.X \rceil \sigma(GM2)) \tag{9}$$

$$Player \stackrel{\text{def}}{=} \lceil sit.PInChair \rceil \sigma(Loser) \tag{10}$$

$$PInChair \stackrel{\text{def}}{=} \sigma.(out\ chair\ stand.0 | stand.Player) \tag{11}$$

$$Loser \stackrel{\text{def}}{=} leave.0 \tag{12}$$

The presence of music is signified by the ticks of a clock σ. A tick from σ is also used to represent the implicit acknowledgement that everyone who can obtain a chair has done so, and that the remaining player left in the room has lost. σ appears as part of a set of clocks on the bottom right of the locality definition to signify that its ticks are visible within the locality (including any nested localities), but not outside. Instead, ticks appear as silent actions outside the location boundaries.

The top right of a locality is used to specify a further property of the locality, the *bouncer*. This is essentially a process with a very limited choice of available actions. It has no real behaviour of its own, but instead performs the job of managing the locality. It dictates whether processes or other localities may enter or exit the locality, and whether the locality may be destroyed by a process in the parent locality. Within the musical chairs model, such protection is irrelevant for the room itself (a bouncer, ω (2), is used which ensures that all possible movements are allowed), but is essential for the chairs (4) and the sin bin (5).

It is the chair bouncer that enforces the implicit predicate that only one player may inhabit a chair at any one time, while the sin bin bouncer prevents players leaving the sin bin once in there.

To model stage one of the game, n player processes and n chair locations are placed in the room. The advantage of using TNT for this model is that the actual number of players or chairs is irrelevant. They only have to be equal at the start to accurately model the game. The calculus allows the creation of a compositional semantics, as discussed in section 1, which work with any n.

For the purposes of demonstration, n is assumed to be two to give the following starting state:

$$room[chair[\mathbf{0}]^{CB}_{\emptyset} \mid chair[\mathbf{0}]^{CB}_{\emptyset} \mid \sigma.\sigma.Player \mid \sigma.\sigma.Player \mid GM2]^{\omega}_{\sigma}. \tag{13}$$

The room and chairs appear as shown earlier. The processes of the form $\sigma.\sigma.Player$ simply wait until two clock cycles have passed, the end of each being signalled by a tick from σ. The intermittent period between the ticks (the second clock cycle) represents the playing of the music. This syntactic form, denoted more generally by $\sigma.P$ (P being some arbitrary process), is derived from the core syntax of TNT as shown in (3). Like most of the model, it relies on the stable timeout operator, $\lceil E \rceil \sigma(F)$, where F acts if E times out on the clock, σ. In this case, E, being $\mathbf{0}$, will always time out as it has no actions to perform.

The gamesmaster ($GM2$ (6)) also waits for the first clock tick (the music starting), but then evolves to $GM3$ (7) and uses the second cycle, prior to the music stopping, to remove a chair from the game. Maximal progress, as explained in section 1, ensures that this occurs before the next clock tick, as the removal emits a silent action.

The most interesting part of the model lies in the interaction with the chairs, which forms part of stages five to seven. The aim of stage five is to get as many player processes as possible inside chair localities. This is handled by again relying on maximal progress to essentially perform a form of broadcast that centres on mobile actions. Rather than sending a signal to a number of recipients, a request to move into a chair (see (8) and (10)) is delivered instead.

If a chair is available, then a player process will enter it (the actual chair and player chosen is non-deterministic). This will cause an internal action to occur, as illustrated by (14), and this will take precedence over the clock tick. Thus, when the clock eventually does tick, it is clear that no more players can enter chairs. Using clocks in this manner makes the system *compositional*; in contrast to other models, players and chairs can be added without requiring changes to the process definitions.

$$\begin{aligned} & GM5 \mid Player \mid chair[\mathbf{0}]^{CB}_{\emptyset} \\ \xrightarrow{\tau} \; & GM5 \mid chair[\mathbf{0} \mid PInChair]^{\overline{out}.CB}_{\emptyset} \end{aligned} \tag{14}$$

Stages six and seven proceed in a similar way. Stage six sees essentially the same broadcasting behaviour applied to the losing players (see (9) and (12)).

The difference is that stage six demonstrates something which wouldn't be possible without mobility: the broadcast is limited to those player processes which remain in the room. Communication between processes in different localities is disallowed in TNT, causing an implicit scoping of the broadcast. The broadcast is again terminated by a tick from σ, which, in this case, also signifies the music starting up again. The remaining players leave their chairs (11), and the system essentially returns to stage three, with $n-1$ chairs and $n-1$ players.

3 Conclusions and Future Work

This extended abstract outlines a calculus which provides a novel combination of features, allowing arbitrary numbers of agents both to synchronize with other agents and move around a dynamic topology, constructed from nested localities. Current work on this calculus focuses on the formalisation of an operational semantics and the creation of a type system to allow additional validity and security checks to be performed. The existing equivalence theory for CaSE will also require extension in order to encompass the new features found in TNT. In the longer term, further case studies will be considered, which go beyond the simple example presented here. In particular, the modelling of quorum sensing bacteria is of interest.

Acknowledgements

This work is supported by a grant from the Engineering and Physical Sciences Research Council (EPSRC). I would also like to thank my supervisor, Mike Stannett, as well as Simon Foster and Georg Struth, for their insightful discussions and support.

References

1. Milner, R.: Communication and Concurrency. Prentice-Hall, London (1989)
2. Hennessy, M., Regan, T.: A process algebra for timed systems. Information and Computation **117**(2) (1995) 221–239
3. Norton, B., Lüttgen, G., Mendler, M.: A compositional semantic theory for synchronous component-based design. In: Proceedings of the 14th International Conference on Concurrency Theory (CONCUR '03). Number 2761 in Lecture Notes in Computer Science, Springer-Verlag (2003) 461–476
4. Cardelli, L., Gordon, A.D.: Mobile ambients. In: Proceedings of the 1st International Conference on Foundations of Software Science and Computation Structures (FoSSaCS '98). Volume 1378 of Lecture Notes in Computer Science., Springer-Verlag (1998) 140–155
5. Levi, F., Sangiorgi, D.: Mobile safe ambients. ACM Transactions on Programming Languages and Systems (TOPLAS) **25**(1) (2003) 1–69
6. Teller, D., Zimmer, P., Hirschkoff, D.: Using ambients to control resources. In Brim, L., Janar, P., Ketinsky, M., Kuera, A., eds.: Proceedings of the 13th International Conference on Concurrency Theory (CONCUR '02). Number 2421 in Lecture Notes in Computer Science, Springer-Verlag (2002) 288–303