Electronic Communications of the EASST
Volume 45 (2011)

Proceedings of the Fourth International Workshop on Formal Methods for Interactive Systems (FMIS 2011)

A Process Algebraic Description of a Temporal Wireless Network Protocol

Colm Bhandal, Melanie Bouroche and Arthur Hughes

17 pages

Guest Editors: Judy Bowen, Steve Reeves
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

A Process Algebraic Description of a Temporal Wireless Network Protocol

Colm Bhandal[1]*, **Melanie Bouroche**[1] **and Arthur Hughes**[1]

[1] School of Computer Science & Statistics
Trinity College, Dublin, Ireland
bhandalc@tcd.ie
http://www.scss.tcd.ie/

Abstract: The problem of coordination is central to research in robotics, automatically guided vehicles, autonomous cars, unmanned aerial vehicles, and any other areas in which autonomous agents of any kind operate concurrently. This paper focuses on one particular model of coordination, namely Comhordú. The contribution of this work is a formalisation of the existing model in precise mathematical terms. This formalisation extends our understanding of the model and provides a basis for future work such as the formal verification of model properties, e.g. system safety.

Keywords: Process calculus, Temporal, Protocol, Model, Wireless, Mobile agent, System, Ad-hoc network.

1 Introduction

The problem of coordination is central to research in robotics [16, 26], automatically guided vehicles [8], autonomous cars [4], unmanned aerial vehicles [2], and any other areas in which autonomous agents of any kind operate concurrently. Of interest here are systems of mobile autonomous agents, mobility here referring to physical movement in space. Coordination in this setting is defined in [6] as "the management of interactions both amongst entities, and between entities and their environment, towards the production of a result."

The Comhordú model is a coordination model for the class of systems comprised of mobile entities communicating over an unreliable wireless network. The model was developed in [6] and [7] as an approach to coordination in a system of mobile agents that differed from the traditional consensus-based approaches [21, 14, 11, 9] and other approaches [12, 27]. Within the model is the notion of a safety constraint, which is specified for a particular system and denotes a property that should never be violated by that system. Also included in the model is a description of how entities should act to maintain this safety constraint, i.e. a protocol entities must follow.

This paper strives towards a formalisation of the Comhordú model. It is the first such attempt: all work on Comhordú e.g. [6, 7] prior to this has been informal. The contribution of such a formalisation if manifold. As stated in [28], a motivation for using formal specification is "to add precision, to aid understanding, and to reason about properties of a design." Precision & understanding go hand in hand. An imprecise model or design of some system is difficult to comprehend and may have many interpretations due to its inherent ambiguities. This lack of

* This author is sponsored by the Irish Research Council for Science, Engineering and Technology, IRCSET.

understanding will most likely lead to the design being incorrectly implemented, perhaps even to the point of system failure. Further to increasing our understanding of a problem, formalisation allows existing mathematical tools such as model checkers, e.g. [5], to be employed towards the assertion of certain system properties.

Our present understanding of Comhordú is certainly enhanced as a direct result of this formalisation. The process of building a formal model of Comhordú has uncovered within the informal model ambiguous concepts, hidden assumptions and under specified behaviours, none of which were obvious prior to this work. These issues are resolved here via the introduction of modifications and extensions to the model. Furthermore, while pre-existing model descriptions are predominantly English-based, this description employs mathematical functions and a process calculus [23] with formally defined rules. This adds an air of precision to the model. Finally, a process algebraic description such as this one offers the future possibility of the specification of system properties in a formal property logic. It is also likely that this formal model can be translated, somewhat approximately, into the language of a model checker which can machine verify to an extent its correctness with respect to some sensible properties, such as the satisfaction of the safety constraint.

The paper is organised as follows. Section 2 summarises the presentation of the protocol found in [6] and [7]. Section 3 presents a semi formal Comhordú model. Extensions and modifications to the original model are proposed therein, and the groundwork is laid for the subsequent formal model. In Section 4, a process language is presented which is used in Section 5 to formally specify the Comhordú protocol. A brief overview of related work is given in Section 6. Finally in Section 7, this work is discussed and compared to related work.

2 Informal Description of the Comhordú System

Comhordú is a coordination model which was developed in [6] and [7]. The model is a means of reasoning about certain systems of autonomous mobile entities which communicate over a wireless network. The model may also be used as an aid in the design of such systems. Summarised in this section are the main aspects of the model. The description here is deliberately informal; a formalised version of the same model will follow in Section 3 and Section 5.

In [6], it is noted that approaches to the problem of achieving coordination which were proposed prior to Comhordú are primarily consensus based, with the exception of a few. While these approaches are discussed in detail in [6], it is sufficient here to highlight that the cornerstone of the Comhordú approach is that it departs from the traditional consensus based approach. The consensus based approach requires that entities should have access to reliable communication and that the number of entities in the system is known a priori. The approach taken by Comhordú is that, since neither of these conditions is guaranteed in a wireless ad-hoc network, a consensus-based approach is not suitable in general. Within the Comhordú framework lies a solution to coordination based on responsibility, an alternative concept to consensus.

2.1 Properties of the Comhordú Model

A Comhordú system[1] incorporates a collection of entities. An entity is some mobile vehicle e.g. a robot. Every entity has a type. The behaviour of an entity is governed partly by its type. At all times, every entity will be in some state or another. The state of an entity contains information about what it is doing, where it is, how fast it is travelling etc. A mode, or mode of operation, is an abstraction of a state. For example, a state might tell us that a car is currently at coordinates $(154,23)$ travelling east at $9.6km/h$ while the mode of that same car might only indicate that the car is travelling at an absolute speed in the range $[0,10]km/h$. A mode may be envisaged as an equivalence class on states. We will assume here that the nature of abstraction from state to mode yields a countable number of modes e.g. it may be that sates are points in some real space while modes are cells in that space.

Every Comhordú system will have associated with it some notion of safe operation e.g. "No vehicles are on a collision course". A constraint on the system, called the safety constraint (C_s) is a condition on the states of all the entities that embodies the safety requirements of the system. This constraint may be evaluated at any time in the system. If it is true, then the system is safe at the time of evaluation. A desirable property of any system is that the safety constraint *always* holds true as long as entities within the system obey the system protocol i.e. as long as their behaviours comply with certain rules of operation set out in the description of the system. A key concept of the Comhordú system is that when entities are sufficiently far apart, they cannot cause an incompatibility. This issue will be addressed later and used to reformulate the safety constraint in simpler terms. Currently, a Comhordú safety constraint is given in terms of an incompatibility that must never occur, where the following grammar describes incompatibilities.

$$
\begin{aligned}
incompatibility \stackrel{\text{def}}{=} & \ (incompatibility, \text{“}\wedge\text{”}, incompatibility) \\
& \mid (incompatibility, \text{“}\vee\text{”}, incompatibility) \\
& \mid (elementType, \text{“.”}, stateVariable, relOperator, value) \\
& \mid (elementType, \text{“.”}, stateVariable, relOperator, elementType, \\
& \text{“.”}, stateVariable) \\
& \mid (\text{“}distance\text{(”}, position, \text{“,”}, position, \text{“)”}, relOperatorr, value) \\
& \mid (\text{“|”}, entityType\text{“.”}, stateVariable, \text{“|”}, relOperator, value) \\
relOperator \stackrel{\text{def}}{=} & \text{“}<\text{”} \mid \text{“}\leq\text{”} \mid \text{“}>\text{”} \mid \text{“}\geq\text{”} \mid \text{“}=\text{”} \mid \text{“}\neq\text{”}
\end{aligned}
$$

The Comhordú model applies to systems communicating over a wireless network. A sub-model of Comhordú is the space elastic model, a wireless communication model upon which Comhordú is built. While the space elastic model concerns itself with problems at low level communication layers, such as collisions at the physical layer, Comhordú deals at a higher level of abstraction, relying on an interface provided by the space elastic model. This interface guarantees time bounded notification of coverage to all entities in the network. Assume an entity begins sending a message at time t. The actual coverage (C_a) of this message is the area to which it is

[1] An instance of the model.

delivered by time $t + msgLatency$, where $msgLatency$ is a fixed constant of the system. Provided by the space elastic model is a notification of C_a to the sender within a time bound $adaptNotif$. In other words, by time $t + msgLatency + adaptNotif$, an entity that has sent a message at time t will be notified of the area to which that message was delivered by time $t + msgLatency$.

2.2 The Comhordú Protocol

A Comhordú system must contain a number of distinguished modes called fail safe modes. In these modes, an entity must not be capable of contributing towards an incompatibility in the system. Hence, if all entities remain in fail safe modes all the time, the system remains trivially safe. However, if an entity wishes to progress towards one of its goals, it will most likely need to transition to or remain in some mode that may result in an incompatibility. For example, if a robot needs to get from point A to point B, is must move to do so, but moving introduces the possibility of a collision. Entering such a mode in an arbitrary way would pose a threat to the safety of the system. Hence, a protocol is imposed upon entities such that safety remains intact.

The protocol dictates that an entity wishing to act in some mode begins periodically sending messages, to entities in its environment, containing its position and the desired mode of operation. The sender waits a pre-determined amount of time, while remaining to periodically broadcast messages, before entering this desired mode and progressing towards its goal. If at any time the sender is notified of a degradation in coverage, or if it receives a message from another entity with which it may become incompatible, it will immediately begin to transition to a different mode that will ensure system safety. A conservative action for entities either receiving messages from other possibly incompatible entities or experiencing coverage difficulties is to immediately transition to a fail safe mode to maintain system safety. Henceforth, we shall refer to an entity in some fail safe mode as being in *the* mode FS, i.e. we will cease to distinguish between different fail safe modes. This is possible as the primary concern of the protocol is ensuring a lack of incompatibilities, and all fail safe modes are equivalent in terms of the inability of any entity in such a mode to cause an incompatibility.

In Figure 1, a sending entity *sender* sends a message to entities in its environment (*receivers*). However, a degradation in coverage occurs and the message is not sent to a sufficient set of receivers. The sender is notified of this degradation by the space elastic model at or before time $t' = t + msgLatency + adaptNotif$, where t is the time the message sending was initiated. Since an entity may not know the result of its message broadcast until time t', it is necessary for every entity to wait for a time of $t' - t = msgLatency + adaptNotif$ before it can act. However, the question still remains as to whether this wait time is sufficient i.e. can an entity begin acting once this time has elapsed? A consideration of this question in Section 3 leads to the conclusion that this wait time alone is not always sufficient.

3 Towards a Formal Comhordú Model

In this section, semi-formal modified definitions of the features discussed in Section 2 are provided. These definitions form a bridge between the informal descriptions of Section 2 and the strictly formal process-algebraic model of Section 5. In addition, new features are proposed here

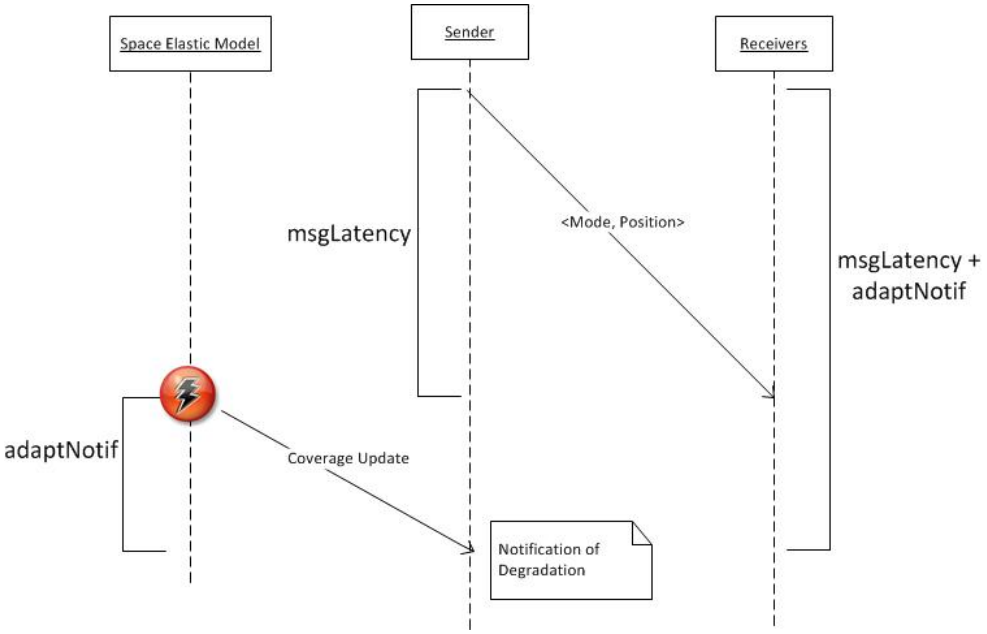

Figure 1: Message sending with coverage degradation

that were absent in the original model. The need to define such features became apparent through the process of formalising the model. The need to modify existing definitions also emerged as a result of the formalisation process. This array of semi-formal definitions, be they restatements of existing definitions or entirely new ones, lays the groundwork for the process-algebraic model.

3.1 Space Elastic Model

Recall the space elastic model from Section 2. A formal definition of the interface provided by this model is given here. The definition is broken into a definition of coverage and a definition of coverage-update. Let us say we observe a Comhordú system evolving from time t, and that an entity e within the system begins sending a message to its environment at this time. Then at time $t' = t + msgLatency$, the actual coverage C_a of e for this observation is the maximum distance d such that if there was an entity e' within this distance of e at time t', e' would have received the message sent by e at t. Now, let us assume the system further evolves and we observe it at time $t'' = t' + adaptNotif$. Then by this time, e is guaranteed by the space elastic model to know C_a i.e. e will be notified of the coverage at time t' within a time bound of $adaptNotif$.

3.2 Safety Constraint

We reformulate the safety constraint in terms of a function $minDistComp$. The function applied to modes i and j, $minDistComp(i, j)$ yields the minimum distance by which entities in those modes must be separated such that the entities are guaranteed to be compatible. The safety constraint then reduces to the assertion that at all times, all pairs of entities in modes i, j are

separated by at least $minDistComp(i,j)$. The function is symmetric. Since entities in fail safe mode cannot cause incompatibilities, the function evaluates to 0 when mode FS is one of its arguments, as per Equation 1.

$$\forall m : \mathbb{M} \bullet minDistComp(m, FS) = 0 \tag{1}$$

3.3 Reasoning about the Coverage

$$\begin{aligned}
Ignore(e,i,s,e',j,s') &\stackrel{\text{def}}{=} i = FS \vee j = FS \vee (e = e') \vee (dist(s,s') \\
rangeOK(i,r) &\stackrel{\text{def}}{=} i = FS \vee r > d_{max}(i) + s_{max}.(trans(i) + period(i) \\
&\quad + max(trans, adaptNotif)) \\
&\quad > minDistComp(i,j) + s_{max}.(trans(j) + period(j) \\
&\quad + max(trans, adaptNotif) + 0.5 msgLatency))c \\
trans &\stackrel{\text{def}}{=} setMax(\{trans(i) | i : \mathbb{M}\}) \\
d_{max}(i) &\stackrel{\text{def}}{=} setMax(\{minDistComp(i,j) | j : \mathbb{M}\})
\end{aligned}$$

We have reasoned about various aspects of the coverage and arrived at the above constant and function definitions that will be needed in the formal model. $Ignore(e,i,s,e',j,s')$ asserts whether a listener e in mode i and position s can ignore a message sent by e' in mode j and position s'. $rangeOK(i,r)$ asserts whether the range r of a message sent by an entity in mode i is sufficient. We assume an upper bound on the time it takes an entity in mode i to transition to mode FS. We call this time $trans(i)$. We then define $trans$ as the maximum such time for any mode. The function $d_{max}(i)$ is the maximum of all the minimum distances of compatibility for all mode pairs including i.

3.4 Mode Transitions

A relation $\rightsquigarrow$ is defined over mode pairs such that $i \rightsquigarrow j$ denotes the fact that an entity in mode i can transition to be in mode j. We have as a feature of the system that $i \rightsquigarrow FS$ for all non fail safe modes. This condition is needed to ensure safety via the protocol. While transitioning between modes, an entity will broadcast as if it were two entities in parallel, one broadcasting i messages, the other broadcasting j messages. This idea is taken from the idea of "soft handovers" [15] in the field of mobile cellular networks. In networks employing this idea, cell coverage overlaps and users crossing the overlapping area between cells remain in the coverage of their original cell while they connect to the new cell. The physical time it takes to transition from mode i to mode j is given by $t_s(i,j)$, while the total wait time is given in Equation 2.

$$t_w(i,j) \stackrel{\text{def}}{=} max(msgLatency + max(adaptNotif, trans), t_s(i,j)) \tag{2}$$

4 The Modelling Language for Comhordú

In this section, the language TCBS' will be presented. The language is strongly based on the Timed Calculus of Broadcasting Systems (TCBS) [23] and has been influenced by the work in [13]. Expressions in this language which constitute a formal Comhordú model will appear in Section 5. The syntax of the language yields its set of terms, which are built inductively. A term belongs to the language iff it can be constructed using the language base terms & constructors. A process P is a closed term, i.e. one without any free variables. The structural operational semantics of the language is given via four relations over processes. The ignore relation asserts whether a process can ignore a value spoken by another process and essentially do nothing. The input relation captures the behaviour of a process that can consume and use some value, thus becoming a different process. The output relation applies to processes that can speak a value and then become some other process. Novel to the timed version of the calculus, the delay relation links processes such that one may delay and become the other. Given below is the inductively built syntax of the TCBS' language.

$$T ::= 0 \mid (\text{in } e)?T \mid (\text{out } e)!T \mid \text{Take}_{i\in I}T_i \mid \text{if } b \text{ then } T \mid A(\vec{e}) \mid T_{(f,g)} \mid (T \mid T) \mid \text{del}(d).T$$

The process 0 denotes the null process. It may delay indefinitely and ignores anything it hears. It cannot evolve to become another process and cannot output any value. An input-prefixed process $(\text{in } e)?T$ is one which listens for a value matching the pattern of e. Since this is a process, all free variables in T are bound by e. Once a value v is heard that matches e, all free variables in e, and hence in T are bound to the matched values. The output prefixed process $(\text{out } e)!P$ may broadcast a value v, to which the expression e evaluates. Notice that e and P are closed here. Here is a simple example to demonstrate the syntax so far. Let $P \stackrel{\text{def}}{=} (\text{in } [x,y,z])?(\text{out } (x+y+z))!0$. Assuming our language is defined over integers and lists of integers, we have that P listens for a list of three integers, outputs their sum, and finally terminates. Let's say P hears $[1,2,3]$. We represent this by the input action $P \xrightarrow{[1,2,3]?} P''$ where $P'' = (\text{out } (1+2+3))!0$. Since $1+2+3=6$, P'' then outputs 6 before terminating. This is represented by the output action $P'' \xrightarrow{6!} 0$.

$\text{Take}_{i\in I}P_i$ denotes a choice over the set of process T_i indexed by elements of a set I. Roughly speaking, this process can choose to behave as any of its constituent processes T_i. For finite sums, this choice notation is often replaced by the infix operator $+$ e.g. in $P+Q$. if b then P behaves exactly as P does when b evaluates to true, and behaves as 0 otherwise. $A(\vec{e})$ is the parametrised process on the vector of closed expressions $\vec{e} = (e_1, e_2, ..., e_n)$. Each such process is given an accompanying definition $A(\vec{x}) \stackrel{\text{def}}{=} T_A$ where the free variables of T_A are contained in $\vec{x}$. $P_{(f,g)}$ is the process P with remap functions f and g applied to it. When P can speak v, this process can speak $f(v)$ and when this process hears u, the nested P hears $g(u)$. $\text{del}(d).P$ is a process that must delay by d before it may perform any of the actions available to P. $(P \mid Q)$ denotes two process operating in parallel.

Table 1 defines the structural operational semantics (SOS) of the TCBS' language. These are the laws which govern how processes, i.e. expressions in the language, evolve. In this table, $comp$ is the symmetric function such that $comp(w!,w?) = comp(w!,w;) = w!$, $comp(w?,w?) = comp(w?,w;) = w?$, $comp(w;,w;) = w;$, $comp(d,d) = d$ and otherwise $comp(\alpha,\beta) = \perp$. When

Table 1: Structural operational semantics of TCBS'

Ignore $(\xrightarrow{w:})$	*Input* $(\xrightarrow{v?})$	*Output* $(\xrightarrow{w!})$	*Delay* $(\xrightarrow{d})$
$0 \xrightarrow{w:} 0$			$0 \xrightarrow{d} 0$
$\dfrac{\forall \vec{v} \bullet w \neq e[\vec{v}/fv(e)]}{(\text{in } e)?T \xrightarrow{w:} (\text{in } e)?T}$	$\dfrac{v = e[\vec{v}/fv(e)]}{(\text{in } e)?T \xrightarrow{v?} T[\vec{v}/fv(e)]}$		$(\text{in } e)?T \xrightarrow{d} (\text{in } e)?T$
$(\text{out } e)!P \xrightarrow{w:} (\text{out } e)!P$		$\dfrac{[\![e]\!] = w}{(\text{out } e)!P \xrightarrow{w!} P}$	
$\dfrac{\forall i \in I \bullet P_i \xrightarrow{w:} P_i}{\text{Take}_{i\in I}P_i \xrightarrow{w:} \text{Take}_{i\in I}P_i}$	$\dfrac{\exists i \in I \bullet P_i \xrightarrow{v?} P'}{\text{Take}_{i\in I}P_i \xrightarrow{v?} P'}$	$\dfrac{\exists i \in I \bullet P_i \xrightarrow{w!} P'}{\text{Take}_{i\in I}P_i \xrightarrow{w!} P'}$	$\dfrac{\forall i \in I \bullet P_i \xrightarrow{d} P'_i}{\text{Take}_{i\in I}P_i \xrightarrow{d} \text{Take}_{i\in I}P'_i}$
$\dfrac{[\![b]\!] = \textbf{false}}{\text{if } b \text{ then } P \xrightarrow{w:} \text{if } b \text{ then } P}$			$\dfrac{[\![b]\!] = \textbf{false}}{\text{if } b \text{ then } P \xrightarrow{d} \text{if } b \text{ then } P}$
$\dfrac{P \xrightarrow{w:} P}{\text{if } b \text{ then } P \xrightarrow{w:} \text{if } b \text{ then } P}$	$\dfrac{P \xrightarrow{v?} P' \qquad [\![b]\!] = \textbf{true}}{\text{if } b \text{ then } P \xrightarrow{v?} P'}$	$\dfrac{P \xrightarrow{w!} P' \qquad [\![b]\!] = \textbf{true}}{\text{if } b \text{ then } P \xrightarrow{w!} P'}$	$\dfrac{P \xrightarrow{d} P' \qquad [\![b]\!] = \textbf{true}}{\text{if } b \text{ then } P \xrightarrow{d} P'}$
$\dfrac{T_A[\vec{e}/\vec{x}] \xrightarrow{\alpha} P'}{A(\vec{e}) \xrightarrow{\alpha} P'}$			
$\dfrac{P \xrightarrow{gw:} P}{P_{(f,g)} \xrightarrow{w:} P_{(f,g)}}$	$\dfrac{P \xrightarrow{gv?} P'}{P_{(f,g)} \xrightarrow{v?} P'_{(f,g)}}$	$\dfrac{P \xrightarrow{w!} P'}{P_{(f,g)} \xrightarrow{fw!} P'_{(f,g)}}$	$\dfrac{P \xrightarrow{d} P'}{P_{(f,g)} \xrightarrow{d} P'_{(f,g)}}$
$\dfrac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\beta} Q' \qquad comp(\alpha,\beta) \neq \perp}{(P \mid Q) \xrightarrow{comp(\alpha,\beta)} (P' \mid Q')}$			
$\dfrac{P \xrightarrow{\alpha} P'}{\text{del}(0).P \xrightarrow{\alpha} P'}$			
			$\dfrac{d' \leq d}{\text{del(d)}.P \xrightarrow{d'} \text{del}(d-d').P}$
$\dfrac{d > 0}{\text{del}(d).P \xrightarrow{w:} \text{del}(d).P}$			$\dfrac{P \xrightarrow{d'} P'}{\text{del}(d).P \xrightarrow{d+d'} P'}$

two concurrent process P and Q perform actions α and β concurrently, $comp(\alpha,\beta)$ denotes the composition of these actions i.e. the overall action of the process $(P \mid Q)$. This is the action seen by the external environment or equivalently an observer. There is a certain precedence present here. Whenever an output is performed by either process, the observed action is output. Otherwise, when there is at least one input, an input is observed to occur overall. Finally, when only ignore actions are performed, the resultant action is also an ignore.

We assume that all terms $A(\vec{x})$ are given associated definitions T_A where the free variables of T_A are a subset of the free variables of $\vec{x}$. Also, the bound variables of T_A are disjoint from the free variables of $\vec{x}$- this can always be achieved by alpha renaming. It is further assumed that all such terms T_A are guarded. A guarded term is one in which all parametrised process names of the form $A(\vec{x})$ are part of some sub-expression whose structure is either an input prefix, and output prefix, or a non-zero delay prefix [1]. This assumption prevents the occurrence of nonsensical definitions such as $A \stackrel{\text{def}}{=} A$. Notice that in the SOS, we only deal with processes i.e. closed terms.

5 Formal Description of the Model

The arguments of Section 3 and the language of Section 4, will now be amalgamated into one coherent formal model of Comhordú. We begin with a description of the system at the highest level. The system is initialised with three parameters. $n : \mathbb{N}$ represents the total number of entities; $S : \mathbb{R}^2[n]$ is a finite list of n points in the plane, which are the positions of the entities; $V : \mathbb{R}^2[n]$ are the velocities of the entities as two dimensional vectors. This system easily generalises to k dimensions. Below is the expression for the system. It is given as n entities running in parallel.

$$Sys(n,S,V) \stackrel{\text{def}}{=} \prod_{i=1}^{n} E(i,s_i,v_i)$$

Here, $\prod_{i=1}^{n} P_i$ is the fold of the binary parallel composition operator over the processes P_i indexed by the set $\{j \in \mathbb{N}_+ | j \leq n\}$. For the sake of formality, let us say that this fold is right associative. Also, we note that s_i, v_i are the elements of lists S, V respectively indexed by i.

The TCBS' language is defined relative to a value type and a boolean type, each with its own associated expression language. Here, the value type will be briefly introduced. The language for boolean expressions B_{exp} will not be presented here, but is assumed to be well behaved. In the following, u, v will range over values in the type Val, b over B_{exp}, c over $Chan$, s over $Base$ and e will be a process identifier of type $\mathbb{N}$.

$$\begin{array}{ll} Base & \stackrel{\text{def}}{=} \mathbb{R} \cup \mathbb{R}^2 \cup \mathbb{N} \cup \{A, W\} \\ Val & \stackrel{\text{def}}{=} s \mid c\langle \vec{s} \rangle \\ Chan & \stackrel{\text{def}}{=} adaptNotif \mid bc \mid coverage(e) \mid fromRelay \mid read \mid start(e) \\ & \quad \mid switch \mid toServer \end{array}$$

5.1 Top level Composition of an Entity

Let us now focus on the structure of the entity process. This process is divided into three logical sub-components. The protocol is a high level description of the message sending/receiving be-

haviour of the entity. The environment buffers outgoing messages and filters incoming messages based on the facts that messages take time to propagate through space and that not all messages reach every entity. Finally, the mobile sub-process records in its parameters the position and velocity of the entity, updating these periodically based on the mode of the entity. All entities are initialised to be in mode FS.

$$\begin{aligned} \mathrm{E}(e,s,v) &\stackrel{\text{def}}{=} (\mathrm{Protocol}(e) \mid \mathrm{Environment}(e) \mid \mathrm{Mobile}(0,s,v))[\emptyset](G)\backslash\{S\} \\ G &\stackrel{\text{def}}{=} \{start(e) \mapsto start, coverage(e) \mapsto coverage\} \\ S &\stackrel{\text{def}}{=} \{read, toServer, switch, fromRelay, adaptNotif\, start, coverage\} \end{aligned}$$

We have used some syntactic shortcuts in this specification. The relabelling shortcut generates a map from values to values given a finite map of channels in the form of a list. The idea is that if one channel is mapped to another in the finite map, then all values constructed with this channel are mapped to values constructed with the other channel. Formally, the syntactic sugar for the remap is defined as follows.

$$\begin{aligned} P[L_F](L_G) &\stackrel{\text{def}}{=} P_{(f,g)} \\ f(c\langle xs\rangle) &\stackrel{\text{def}}{=} F(c)\langle xs\rangle \\ g(c\langle xs\rangle) &\stackrel{\text{def}}{=} G(c)\langle xs\rangle \end{aligned}$$

Here, F and G are functions from channel names to channel names. They are specified by the finite lists of pairs L_F and L_G respectively. Pairs in each list are of the form $c \mapsto c'$ such that no c occurs as the first element of more than one pair in a list. Channels c not explicitly mapped by these lists are assumed to map to themselves. The definition for F is given below. G follows an analogous definition in terms of L_G.

$$F(c) = \begin{cases} c' & \text{if } c \mapsto c' \in L_F; \\ c & \text{if } \forall c' \bullet c \mapsto c' \notin L_F. \end{cases}$$

We have also defined a restriction shortcut for channels. Here, it is desired that a process P cannot send/receive messages to/from its environment over any channel in the set of channels S. Since channels are modelled in this language by value constructors, this is the same as saying that P cannot send to or receive from its environment any values constructed using the value constructors in S. More formally:

$$\begin{aligned} P\backslash\{C\} &\stackrel{\text{def}}{=} P[C'](C') \\ C' &\stackrel{\text{def}}{=} \{c \mapsto \tau \mid c \in C\} \end{aligned}$$

5.2 Protocol Agents

Zoning in on the protocol process, which constitutes the bulk of the system, we are faced with a parallel composition of six components. These six components can themselves be grouped into two sets of 3: the active set, and the dormant set. In the active set are three processes. Active(i)

executes the task of periodically sending messages to its environment containing mode and position information, as per the protocol description. LisAct(e,i) receives incoming messages from the environment. If any of these messages cannot be ignored, then this process will initiate a transition of the entity to fail safe mode via an internal broadcast to all protocol sub-components. ANCheckA(i) performs a similar duty to the message listener but it instead listens for degradations in coverage. The dormant set contains three processes also. The dormant process is so called because it idles for a time until the entity decides to change mode. This decision is modelled via input on the channel *start*. The process then evolves to a waiting process, which like the active process periodically broadcasts both mode and position information.

The processes ANCheckD(i) and LisDorm(e,i) are analogous to Active(i) and LisAct(e,i) respectively, only instead of initiating fail safe transitions on reception of a coverage degradation or incompatible message, they initiate an *abort* broadcast which causes a rollback action of the waiting process to its dormant state. The rollback is possible as opposed to a full fail safe transition because the entity has not yet entered the waiting mode yet. The Protocol process is parametrised on the id of the entity to which it belongs. All its components are initialised to fail safe mode.

$$\begin{aligned}
\text{Protocol}(e) &\stackrel{\text{def}}{=} (\text{Active}(0) \mid \text{Dormant}(0) \mid \text{LisAct}(e,0) \mid \text{LisDorm}(e,0) \\
&\quad \mid \text{ANCheckA}(0) \mid \text{ANCheckD}(0)) \backslash \{trans, abort\}
\end{aligned}$$

$$\begin{aligned}
\text{Active}(i) &\stackrel{\text{def}}{=} (\text{out } read\langle\rangle)!((\text{in } read\langle s\rangle)?((\text{out } toServer\langle A,i,s\rangle)!(\text{del}(p(i)).\text{Active}(i) + \text{A}') \\
&\quad + \text{A}') + \text{A}') + \text{A}' \\
\text{A}' &\stackrel{\text{def}}{=} (\text{in } switch\langle j\rangle)?\text{Active}(j)
\end{aligned}$$

$$\begin{aligned}
\text{Dormant}(i) &\stackrel{\text{def}}{=} \text{Take}_{j\in\{k\mid i\rightsquigarrow k\}}(\text{in } start\langle j\rangle)?\text{Waiting}(i,j,t_w(i,j)) + \text{D}'(i) \\
\text{D}'(i) &\stackrel{\text{def}}{=} (\text{in } trans)?\text{Waiting}(i,0,trans(i)) + (\text{in } abort)?\text{Dormant}(i) \\
\text{Waiting}(i,j,t) &\stackrel{\text{def}}{=} (\text{out } read\langle\rangle)!(\text{in } read\langle s\rangle)?((\text{out } toServer\langle W,j,s\rangle)!\text{W}'(i,j,t) + \text{D}'(i)) + \text{D}'(i) \\
\text{W}'(i,j,t) &\stackrel{\text{def}}{=} \text{del}(p(j)).\text{Waiting}(i,j,t-p(j)) \\
&\quad + \text{del}(t).(\text{out } switch\langle j\rangle)!\text{Dormant}(j) + \text{D}'(i)
\end{aligned}$$

$$\begin{aligned}
\text{LisAct}(e,i) &\stackrel{\text{def}}{=} (\text{in } fromRelay\langle e',j,s'\rangle)?(\text{LisAct}(e,i) \\
&\quad \mid \text{MessHand}(e,i,e',j,s')) + \text{LA}'(e)
\end{aligned}$$

$$\begin{aligned}
\text{MessHand}(e,i,e',j,s') &\stackrel{\text{def}}{=} (\text{out } read\langle\rangle)!(\text{in } read\langle s\rangle)?\text{MH}'(e,i,s,e',j,s') \\
\text{LA}'(e) &\stackrel{\text{def}}{=} (\text{in } trans)?\text{LisAct}(e,0) + (\text{in } switch\langle j\rangle)?\text{LisAct}(e,j) \\
\text{MH}'(e,i,s,e',j,s') &\stackrel{\text{def}}{=} \text{if } \neg Ignore(e,i,s,e',j,s') \text{ then } (\text{out } trans)!0
\end{aligned}$$

$$\begin{aligned}
\text{LisDorm}(e,i) &\stackrel{\text{def}}{=} \text{Take}_{j\in\{k|i\rightsquigarrow k\}}(\text{in } start\langle j\rangle)?\text{LW}(e,i,j) + \text{LW}'(e,i) \\
\text{LW}(e,i,j) &\stackrel{\text{def}}{=} (\text{in } fromRelay\langle e',j',s'\rangle)?(\text{LW}(e,i,j) \\
&\quad \mid \text{MessHand}(e,j',e',j,s')_{(\{trans\mapsto abort\},\emptyset)}) \\
&\quad + \text{LW}'(e,i) \\
\text{LW}'(e,i) &\stackrel{\text{def}}{=} (\text{in } trans)?\text{LW}(e,i,0) \\
&\quad + (\text{in } switch\langle j\rangle)?\text{LisDorm}(e,j) + (\text{in } abort)?\text{LisDorm}(e,i)
\end{aligned}$$

$$\begin{aligned}
\text{ANCheckA}(i) &\stackrel{\text{def}}{=} (\text{in } adaptNotif\langle t,i,r\rangle)?(\text{ANCheckA}(i) \mid \text{ANC}'(i,r)) \\
&\quad + (\text{in } switch\langle j\rangle)?\text{ANCheckA}(j) \\
\text{ANC}'(i,r) &\stackrel{\text{def}}{=} \text{if } \neg rangeOK(i,r) \text{ then } (\text{out } trans)!0
\end{aligned}$$

$$\begin{aligned}
\text{ANCheckD}(i) &\stackrel{\text{def}}{=} \text{Take}_{j\in\{k|i\rightsquigarrow k\}}(\text{in } start\langle j\rangle)?\text{ANCheckW}(i,j) \\
\text{ANCheckW}(i,j) &\stackrel{\text{def}}{=} (\text{in } adaptNotif\langle W,j,r\rangle)?(\text{ANCheckW}(i,j) \\
&\quad \mid \text{ANC}'(j,r)_{(\{trans\mapsto abort\},\emptyset)}) + (\text{in } switch\langle j\rangle)?\text{ANCheckD}(j) \\
&\quad + (\text{in } trans)?\text{ANCheckW}(i,0) + (\text{in } abort)?\text{ANCheckD}(i)
\end{aligned}$$

5.3 Environment & Space Elastic Model

Here the focus is on processes which model the environment i.e. the propagation of messages through space and the filtering of these messages based on their coverage. Also modelled in this logical sub-component is the interface provided by the space elastic model, which provides any sender with a coverage notification $adaptNotif$ time units after message sending is initiated.

Server(e) buffers outgoing messages. When an entity sends a message, it is caught by this server process and held in a freshly created buffer process for a time of *msgLatency*, after which it is then forwarded on to all other entities. RelayServ catches incoming messages and tests them based on their coverage field. If the coverage of the sent message is less than the distance between sender and receiver, then the message is discarded. Otherwise, it is passed through to the protocol process. Let it be noted that in an actual implementation of a Comhordú system, these environment processes would not exist. Instead, they would be replaced by the actual environment. Furthermore, message fields such as the exact position of the sender at the time of delivery of the message would not be included in the message, nor would the coverage- these would be emergent properties. However, they are necessary here to encode the assumed behaviour of the environment.

$$\text{Environment}(e) \stackrel{\text{def}}{=} \text{RelayServ} \mid \text{Server}(e)$$

$$\begin{aligned}
\text{Server}(e) &\stackrel{\text{def}}{=} (\text{in}\, \mathit{toServer}\langle t,i,s\rangle)?(\text{Server}(e) \mid \text{Buffer}(e,t,i,s)) \\
\text{Buffer}(e,t,i,s) &\stackrel{\text{def}}{=} \text{del}(\mathit{msgLatency}).(\text{in}\, \mathit{coverage}\langle r\rangle)?(\text{in}\, \mathit{read}\langle s'\rangle)?(\text{out}\, \mathit{bc}\langle r,e,i,s,s'\rangle)! \\
&\qquad \text{del}(\mathit{adaptNotif}).(\text{out}\, \mathit{adaptNotif}\langle t,i,r\rangle)!0
\end{aligned}$$

$$\begin{aligned}
\text{RelayServ} &\stackrel{\text{def}}{=} (\text{in}\, \mathit{bc}\langle r,e,i,s',s''\rangle)?(\text{RelayServ} \mid \text{Relay}(r,e,i,s',s'')) \\
\text{Relay}(r,e,i,s',s'') &\stackrel{\text{def}}{=} (\text{out}\, \mathit{read}\langle\rangle)!(\text{in}\, \mathit{read}\langle s\rangle)?\text{R}'(r,s,s'',e,i,s') \\
\text{R}'(r,s,s'',e,i,s') &\stackrel{\text{def}}{=} \text{if}\, \mathit{inRange}(r,s,s'')\, \text{then}\, (\text{out}\, \mathit{fromRelay}\langle e,i,s'\rangle)!0
\end{aligned}$$

The predicate *inRange* used in these expressions asserts whether or not a message, whose radius of delivery is specified, should be delivered to a given entity based on the position of this entity and the sending entity. Equation 3 defines this predicate.

$$\mathit{inRange}(r,s,s') \stackrel{\text{def}}{=} (\mathit{dist}(s,s') \leq r) \tag{3}$$

5.4 Mobile Agent

The mobile agent is a simple process modelling the physical state of an entity. $\text{Mobile}(i,s,v)$ records via its parameters the position s, velocity v and mode i of the entity to which is belongs. It periodically updates the position and velocity based on functions $s'(i,s,v)$ and $v'(i,s,v)$ respectively. It is assumed here that the mode i along with the current position and velocity is enough to determine the new position and velocity. However, this process, being independent of the rest of the system, can easily be altered to change the way position and velocity are updated e.g. by adding new parameters. The period of update is arbitrarily chosen and represents a discrete tick in time. Since this process calculus does not allow continuous actions, this discrete approximation must suffice. We now take notice of the functions s'' and v''. These make an approximate correction to the velocity and position when this process is interrupted by a mode change, which is necessary since it is impossible to know how much time has elapsed since the last update. The nature of such a correction will not be examined here, though it should be such that error is minimised in some sense.

$$\begin{aligned}
\text{Mobile}(i,s,v) &\stackrel{\text{def}}{=} \text{MobTrack}(i,s,v) \mid \text{MobServ}(s) \\
\text{MobTrack}(i,s,v) &\stackrel{\text{def}}{=} \text{del}(\mathit{update}).(\text{out}\, \mathit{read}\langle s\rangle)!\text{MobTrack}(i,s'(i,s,v),v'(i,s,v)) \\
&\qquad + (\text{in}\, \mathit{switch}\langle j\rangle)?\text{MobTrack}(i,s''(i,s,v),v''(i,s,v)) \\
\text{MobServ}(s) &\stackrel{\text{def}}{=} (\text{in}\, \mathit{read}\langle\rangle)?((\text{out}\, \mathit{read}\langle s\rangle)!\text{MobServ}(s) + \text{MS}) + \text{MS} \\
\text{MS} &\stackrel{\text{def}}{=} (\text{in}\, \mathit{read}\langle s'\rangle)?\text{MobServ}(s')
\end{aligned}$$

6 Related Work

In this section, we will briefly mention some related modelling languages to TCBS' and highlight their weaknesses, compared to TCBS', in terms of their applicability to the Comhordú problem. As mentioned earlier, TCBS' is a variation on the existing TCBS of [23], which itself is an extension of CBS [24]. The differences between TCBS' and TCBS are mostly influenced by the changes made to CBS in [13]. The advantage of TCBS' over TCBS is the clarity of notation achieved therein. Other formalisms were considered as alternatives to the timed broadcasting calculus. UPPAAL [5] is a model checker for finite timed automata [3]. The advantage of using UPPAAL to model a system is that certain system properties can be specified and machine-checked. However, the finiteness restriction of UPPAAL does not accommodate a fully general model and this has lead us towards the more powerful paradigm of process algebras. There are a myriad of process algebras currently in existence. We narrow our attention only to those most suitable.

Recently developed are a group of algebras for modelling wireless systems [18, 20, 17, 22, 25]. At first glance, these algebras seem ideal for the modelling task at hand. However, many of them focus on low level aspects of the wireless network such as collision detection. This would suit a formalised space elastic model but is too detailed for Comhordú. The languages also lack any notion of time. A similar calculus that does include time is given in [19], but the time modelled is in ticks rather than delays of real amounts and again the focus is on collisions. Long standing timed process algebras such as TCCS [29] and TCSP [10] were considered. However, the weakness of these languages is that they do not include broadcast as a communication primitive, whereas TCBS' does, thus making it more suitable to the task of modelling an inherently broadcast-based system than any of these other languages considered.

7 Conclusions & Future Work

In this paper a formalisation of the Comhordú model has been achieved. This formalisation provides us with a clearer understanding of the model, removing any ambiguity from pre-existing model descriptions. It paves the way for future verification work using model checkers. As a general framework, the model also serves as a guide to the construction of model instances i.e. formalisations of particular Comhordú systems. Finally, it has emerged through the process of formalising the Comhordú system that some new concepts were required at the heart of the model. In particular, there was the notion of "soft handover" from one mode to another, an idea taken from the telecoms community; the safety constraint language was reduced to the simpler notion of distance-based incompatibilities, which unlike expressions in the original language seem satisfiable in general by the protocol alone; coverage zones were re-evaluated and deduced from first principles, with some new constants and functions defined to yield expressions for sufficient coverage, wait time and ignorable messages.

Let us now consider future explorations that may be carried out to extend this work. The ultimate goal is to prove the general safety of the model, i.e. prove that for any Comhordú system adhering to the Comhordú protocol, the safety constraint holds. As discussed earlier, to prove this safety condition, it will be enough to prove that adherence to the Comhordú protocol implies

sufficient spacing of the entities according to the distance function supplied. Other properties of the system also remain to be investigated such as those of liveness and deadlock. The intuition at the moment is that we will develop a property logic enabling us to express such conditions. After this, we can attempt proofs of certain properties by hand or we can develop some approximation of this system in a model checker like UPPAAL & machine check these properties. It is expected that both techniques will be employed. There have already been investigations performed on the behaviour of the system via the application of the TCBS' SOS laws to the formal system expressions of Section 5 to yield traces through the system. Due to spatial limitations however, these traces could not be included here.

Acknowledgements: We extend a special thanks to Matthew Hennessy for his insights and his advice relating to this work. We also express our gratitude towards the anonymous reviewers of this paper for their feedback and their time.

Bibliography

[1] L. Aceto, A. Ingólfsdóttir, K. G. Larsen, and J. Srba. *Reactive systems: modelling, specification and verification*. Cambridge Univ Pr, 2007.

[2] M. Alighanbari, Y. Kuwata, and J. How. Coordination and Control of Multiple UAVs with Timing Constraints and Loitering. In *Proceedings ofAmerican Control Conference*, 2003.

[3] R. Alur and D.L. Dill. A theory of timed automata* 1. *Theoretical computer science*, 126(2):183–235, 1994.

[4] J. Baber, J. Kolodko, T. Noel, M. Parent, and L. Vlacic. Cooperative autonomous driving: intelligent vehicles sharing city roads. *IEEE Robotics & Automation Magazine*, 12(1):44–49, 2005.

[5] Gerd Behrmann, Alexandre David, and Kim Larsen. A tutorial on uppaal. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *Lecture Notes in Computer Science*, pages 33–35. Springer Berlin / Heidelberg, 2004.

[6] M. Bouroche. *Real-Time Coordination of Mobile Autonomous Entities*. PhD thesis, University of Dublin, Trinity College, 2007.

[7] M. Bouroche and V. Cahill. We don't need to agree to coordinate. *Workshop on Dependable Network Computing and Mobile Systems (DNCMS'08)*, 1:47–51, October 2008.

[8] J. Cawkwell. A visually guided agv for use as passenger transport in urban areas. In *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pages 311–315. IEEE, 2000.

[9] C.M. Clark. *Dynamic robot networks: a coordination platform for multi-robot systems*. PhD thesis, Citeseer, 2004.

[10] J. Davies and S. Schneider. A brief history of timed csp. *Theoretical Computer Science*, 138(2):243–271, 1995.

[11] C.L. Fok, G.C. Roman, and G. Hackmann. A lightweight coordination middleware for mobile computing. In *Coordination Models and Languages*, pages 135–151. Springer, 2004.

[12] V. Gervasi and G. Prencipe. Coordination without communication: the case of the flocking problem. *Discrete Applied Mathematics*, 144(3):324–344, 2004.

[13] M. Hennessy and J. Rathke. Bisimulations for a calculus of broadcasting systems* 1. *Theoretical Computer Science*, 200(1-2):225–260, 1998.

[14] C. Julien and G.C. Roman. Active coordination in ad hoc networks. In *Coordination Models and Languages*, pages 199–215. Springer, 2004.

[15] R. Koodli and C.E. Perkins. Fast handovers and context transfers in mobile networks. *ACM SIGCOMM Computer Communication Review*, 31(5):37–47, 2001.

[16] N. Kubota, Y. Nojima, N. Baba, F. Kojima, and T. Fukuda. Evolving pet robot with emotional model. In *Proceedings of the 2000 Congress on Evolutionary Computation, 2000.*, volume 2, pages 1231–1237. IEEE, 2000.

[17] I. Lanese and D. Sangiorgi. An operational semantics for a calculus for wireless systems. *Theoretical Computer Science*, 411(19):1928–1948, 2010.

[18] M. Merro. An observational theory for mobile ad hoc networks. *Electronic Notes in Theoretical Computer Science*, 173:275–293, 2007.

[19] Massimo Merro and Eleonora Sibilio. A timed calculus for wireless systems. In Farhad Arbab and Marjan Sirjani, editors, *Fundamentals of Software Engineering*, volume 5961 of *Lecture Notes in Computer Science*, pages 228–243. Springer Berlin / Heidelberg, 2010.

[20] N. Mezzetti and D. Sangiorgi. Towards a calculus for wireless systems. *Electronic Notes in Theoretical Computer Science*, 158:331–353, 2006.

[21] A.L. Murphy, G.P. Picco, and G.C. Roman. Lime: A middleware for physical and logical mobility. In *International Conference On Distributed Computing Systems*, page 0524. Published by the IEEE Computer Society, 2001.

[22] S. Nanz and C. Hankin. A framework for security analysis of mobile wireless networks. *Theoretical Computer Science*, 367(1-2):203–227, 2006.

[23] K. Prasad. Broadcasting in time. In Paolo Ciancarini and Chris Hankin, editors, *Coordination Languages and Models*, volume 1061 of *Lecture Notes in Computer Science*, pages 321–338. Springer Berlin / Heidelberg, 1996.

[24] KVS Prasad. A calculus of broadcasting systems. *Science of Computer Programming*, 25(2-3):285–327, 1995.

[25] KVS Prasad. A prospectus for mobile broadcasting systems. *Electronic Notes in Theoretical Computer Science*, 162:295–300, 2006.

[26] R.D. Schraft. Mechatronics and robotics for service applications. *Robotics Automation Magazine, IEEE*, 1(4):31 –35, dec 1994.

[27] P. Verissimo and A. Casimiro. Event-driven support of real-time sentient objects. In *Object-Oriented Real-Time Dependable Systems, 2003. (WORDS 2003). Proceedings of the Eighth International Workshop on*, pages 2 – 9, jan. 2003.

[28] J. Woodcock and J. Davies. *Using Z: specification, refinement, and proof*, volume 39. Prentice Hall, 1996.

[29] Wang Yi. Real-time behaviour of asynchronous agents. In J. Baeten and J. Klop, editors, *CONCUR '90 Theories of Concurrency: Unification and Extension*, volume 458 of *Lecture Notes in Computer Science*, pages 502–520. Springer Berlin / Heidelberg, 1990.