The Timed, Compensable Conversation Calculus

Hugo A. López
PLS, IT University of Copenhagen

Jorge A. Pérez
CITI - DI, FCT - New University of Lisbon

Abstract

We are interested in the interplay of timing issues and exceptional behavior in the realm of structured communications. As an initial step, we have defined C^3, a variant of the Conversation Calculus in which conversation contexts are enriched with a time duration, a compensating activity, and a designated signal for conversation abortion. In this note, we present C^3 and its basic semantics, and discuss ongoing work for equipping the language with logic-based reasoning techniques.

1 Introduction

This paper investigates the rôle of *time* in languages for structured communication. In particular, we are interested in how time and constructs for *exceptional behavior* can be jointly captured in models of structured communications and lead to more effective reasoning techniques.

Time plays an increasingly crucial rôle in practical scenarios of structured communication. Consider, for instance, a web banking application: interaction between a user and her bank generally takes place by means of a secure session, which is meant to expire after a certain period of inactivity of the user. When that occurs, the user must exhibit again her authentication credentials, so as to initiate another session or to continue with the expired session. In some cases, the session (or parts of it) have a predetermined duration and so interactions may also be bounded in time. The user must then reinitiate the session if, for instance, she is taking too long in providing some input or if the network connection is too slow. Crucially, the different incarnations of time in interactions (session durations, timeouts, delays) can be seen to be closely related to the behavior of the whole system in exceptional circumstances. In the previous example: a specification of the web banking application would appear incomplete unless one specifies how the system should behave when, e.g., the session has expired and the user attempts to reinitiate it, or when an interaction within the session is taking longer than anticipated.

It then appears that in the specification of practical instances of structured communications timed behavior goes hand in hand with exceptional behavior. This observation acquires more relevance in the realm of *healthcare scenarios* [14]—a central source of motivation for our work. In healthcare applications, structured communications often involve strict *time bounds*, such as, e.g., "monitor the patient's signs every two hours, for the following 48 hours". They may also include behavioral patterns which may be defined as both a *default behavior* and an *alternative behavior* to be executed in case of unexpected conditions, including time-related ones: "contact the substitute doctor if the responsible doctor cannot be reached within 15 minutes". Also, structured communications involve critical tasks that may be indefinitely *suspended* or *aborted*, such as, e.g.,"stop the administering the medicine if the patient reacts badly to it". We find that this kind of requirements are all hard to express appropriately in known formalisms for structured communication.

Here we report on our initial steps towards a language for structured communications with a explicit treatment of time and exceptional behavior. It arises as an extension of the Conversation Calculus (CC) [16, 15] in which conversations have durations and are sensible to compensations. The CC is an interesting base language for our study. First, it is a *simple* model: the CC corresponds to a π-calculus extended with *conversation contexts*. Hence, we may draw inspiration from previous work on variants of the π-calculus with time and constructs for exceptional behavior. Second, the CC allows for the compact specification of *multiparty interactions*, which are ubiquitous in healthcare scenarios. Third, the CC counts with a number of *reasoning techniques* to build upon, which include a behavioral theory and so-called *conversation types* [3] which, by means of suitable projections, allow for a unified treatment of the global and local views of a system (also known as *choreographies* and *orchestrations*, respectively).

We illustrate a few basic notions of the CC by means of an example—the *purchasing scenario* proposed in, e.g., [7, 15]. This scenario describes the interaction between a buyer and a seller for buying a certain product; the seller later involves a shipper who is in charge of delivering the product. In the CC, this scenario can be given as follows:

$$\begin{array}{l} \mathit{Buyer} \blacktriangleleft [\ \mathbf{new}\, \mathit{Seller} \cdot \mathtt{BuyService} \Leftarrow \mathtt{buy}^{\downarrow}!(\mathit{prod}).\mathtt{price}^{\downarrow}?(p).\mathtt{details}^{\downarrow}?(d)\ ] \\ \mid\ \mathit{Seller} \blacktriangleleft [\ \mathit{PriceDB} \mid\ \mathbf{def}\, \mathtt{BuyService} \Rightarrow \mathtt{buy}^{\downarrow}?(\mathit{prod}).\mathtt{askPrice}^{\uparrow}!(\mathit{prod}). \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathtt{priceVal}^{\uparrow}?(p).\mathtt{price}^{\downarrow}!(p). \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathbf{join}\, \mathit{Shipper} \cdot \mathtt{DeliveryService} \Leftarrow \mathtt{product}^{\uparrow}!(\mathit{prod})\ ] \\ \mid\ \mathit{Shipper} \blacktriangleleft [\ \mathbf{def}\, \mathtt{DeliveryService} \Rightarrow \mathtt{product}^{\downarrow}?(p).\mathtt{details}^{\downarrow}!(\mathit{data})\ ] \end{array}$$

In the CC, a *conversation context* represents a distributed communication medium where partners may interact. We write $n \blacktriangleleft [\ P\ ]$ for the conversation context with behavior P and identity n; process P may seamlessly interact with processes inside any other conversation contexts with identity n. The model above considers three participants: *Buyer*, *Seller*, and *Shipper*. *Buyer* invokes a new instance of the `BuyService` service, which is defined by *Seller*. As a result, a *conversation* on a fresh name is established between them; this name can then be used to exchange information on the desired product and its price (the latter is retrieved by *Seller* from *PriceDB*, which represents a database). When the transaction has been agreed, *Shipper* joins in the conversation, and receives product information from *Seller* and delivery details from *Buyer*. Notice how **def**, **new**, and **join** —the three *idioms* available in the CC— are used to define services, creating a new instance of a predefined service, and joining a running service instance, respectively. Remarkably, these idioms can be derived from the basic syntax of the CC:

$$\begin{array}{lcllcl} \mathbf{def}\, s \Rightarrow P & \triangleq & \mathtt{s}^{\downarrow}?(x).x \blacktriangleleft [\ P\ ] & \mathbf{new}\, n \cdot s \Leftarrow Q & \triangleq & (\nu c)(n \blacktriangleleft [\ \mathtt{s}^{\downarrow}!(c)\ ] \mid c \blacktriangleleft [\ Q\ ]) \\ \mathbf{join}\, n \cdot s \Rightarrow Q & \triangleq & \mathbf{this}(x).(n \blacktriangleleft [\ s^{\downarrow}!(x)\ ] \mid Q) & \star\mathbf{def}\, s \Rightarrow P & \triangleq & \mu X.\, \mathtt{s}^{\downarrow}?(x).(x \blacktriangleleft [\ P\ ] \mid X) \end{array}$$

where $\mathtt{s}^{\downarrow}?(x)$ and $\mathtt{s}^{\downarrow}!(c)$ are input and output prefixes (as in the π-calculus), and $\mathbf{this}(x)$ is a prefix that binds the enclosing conversation context to name x. $\star\mathbf{def}\, s \Rightarrow$ is the *persistent* version of $\mathbf{def}\, s \Rightarrow$.

We would like simple yet general constructs capturing both timing issues and exceptional behavior, and the relationship between the two. Constructs such as try/catch appear inadequate for our purposes. In fact, the kind of relationship between time and exceptional behavior we would like to address is more a *specification* issue than a *runtime* issue: we find that the "runtime character" of constructs such as try/catch cannot be lifted to specifications in a natural way. In order to include both time and exceptional behavior into specifications, we have opted to include these issues *directly* into conversation contexts. We propose C^3, a variant of the CC in which the *simple* conversation contexts $n \blacktriangleleft [\ P\ ]$ are replaced by *timed*, *compensable* conversation contexts, denoted as $n \blacktriangleleft [P; Q]^t_\kappa$. As before, n is the identity of the conversation context. Process P describes the *default* behavior for n, which is performed while the *duration* t is greater than 0. Observable actions from P witness the time passage in n; as soon as $t = 0$, the default behavior is dynamically replaced by Q, the *compensating* behavior. The duration t can also be ∞. Name κ represents an *abort* mechanism: the interaction of $n \blacktriangleleft [P; Q]^t_\kappa$ with the *kill signal* $\kappa^\dagger$ immediately sets t to 0. Notice that the simple conversation context $n \blacktriangleleft [\ P\ ]$ in the CC would correspond in C^3 to the extended context $n \blacktriangleleft [P; Q]^\infty_\kappa$, for some κ unknown to P and to the rest of the system.

An immediate and pleasant consequence of the timed, compensable conversation contexts in C^3 is that the signature of the service idioms can be extended too. In this way, service specifications in C^3 can express richer information on compensation signals, timeouts, and exceptional behavior. This achieves the above-mentioned "lifting" of timed and exceptional behavior to (service) specifications. Note that it suffices to extend **def** and **new**, the idioms representing service invocation and service definition:

$$\begin{array}{lcl} \mathbf{def}\, s\ \mathbf{with}\ (\kappa,t) \Rightarrow \{P; Q\} & \triangleq & \mathtt{s}^{\downarrow}?(y).\, y \blacktriangleleft [P; Q]^t_\kappa \\ \mathbf{new}\, n \cdot s\ \mathbf{with}\ (\kappa,t) \Leftarrow \{P; Q\} & \triangleq & (\nu c)\ (n \blacktriangleleft [\mathtt{s}^{\downarrow}!(c); \mathbf{0}]^\infty_\emptyset \mid c \blacktriangleleft [P; Q]^t_\kappa) \end{array}$$

where y and c are assumed to be fresh in P and Q, and different from κ, t, n. The definition of a service s with (main, default) protocol P is extended with a compensation signal κ, a timeout value t, and an alternative protocol definition Q, to be used in case of failure or unexpected behavior of P. Similarly, the extended idiom for invoking an new instance of service s residing at n considers a main protocol P, an alternative protocol Q, a duration, and a compensation signal. This way, for instance, processes

$$\mathit{Client} \blacktriangleleft [\, \textbf{new}\ \mathit{Provider} \cdot \texttt{Service}\ \textbf{with}\ (\kappa_c, t_c) \Leftarrow \{P; Q\}\,] \qquad \mathit{Provider} \blacktriangleleft [\, \textbf{def}\ \texttt{Service}\ \textbf{with}\ (\kappa_p, t_p) \Rightarrow \{R; T\}\,]$$

may interact and evolve into $(\nu s)\,(\mathit{Client} \blacktriangleleft [\, s \blacktriangleleft [P; Q]^{t_c}_{\kappa_c}\,] \mid \mathit{Provider} \blacktriangleleft [\, s \blacktriangleleft [R; T]^{t_p}_{\kappa_p}\,])$.

We illustrate the expressiveness of C^3 by extending the purchase scenario. Suppose a buyer who is willing to interact with a specific provider only for a finite amount of time. She engages in conversations with several providers at the same time, picks the one which presents the best offer, and finally abandons the conversations with the other providers. This refined scenario can be modeled in C^3 as follows:

$$\begin{array}{l}
\mathit{NewBuyer} \blacktriangleleft [\prod_{i\in\{1,2,3\}} \textbf{new}\ \mathit{Seller}_i \cdot \texttt{BuyService}\ \textbf{with}\ (e_i, v_i) \Leftarrow \{P_i; Q_i\} \mid \mathit{Control}; \mathit{CancelOrder}]^{tmax}_{x} \\
\quad \mid \prod_{i\in\{1,2,3\}} \mathit{Seller}_i \blacktriangleleft [\mathit{PriceDB} \mid \textbf{def}\ \texttt{BuyService}\ \textbf{with}\ (b_i, w_i) \Rightarrow \{\texttt{offer}^{\downarrow}?(\mathit{prod}).\texttt{askPrice}^{\uparrow}!(\mathit{prod}). \\
\qquad\qquad\qquad\qquad \texttt{priceVal}^{\uparrow}?(p).\,\texttt{price}^{\downarrow}!(p). \\
\qquad\qquad\qquad\qquad \textbf{join}\,\mathit{Shipper} \cdot \texttt{DeliveryService} \Leftarrow \texttt{product}^{\uparrow}!(\mathit{prod}); R_i\}; \\
\qquad\qquad\qquad \mathit{CancelSell}_i]^{t_i}_{x_i} \\
\quad \mid \mathit{Shipper} \blacktriangleleft [\textbf{def}\ \mathit{DeliveryService}\ \textbf{with}\ (d,t) \Rightarrow \{\texttt{product}^{\downarrow}?(p).\,\texttt{details}^{\downarrow}!(\mathit{data}); T\}; \mathbf{0}]^{t_3}_{z}
\end{array}$$

where process $P_i \triangleq \texttt{offer}^{\downarrow}!(\mathit{prod}).\,\texttt{price}^{\downarrow}?(p).\,\texttt{com}_i{}^{\uparrow}!(p).\,\texttt{details}^{\downarrow}?(d)$ represents the default behavior of the buyer when interacting with Seller_i (the compensating behavior Q_i is left unspecified) and $\mathit{Control} \triangleq \star(\texttt{com}_1{}^{\uparrow}?(p).\,(c_2^{\dagger} \mid c_3^{\dagger}) + \texttt{com}_2{}^{\uparrow}?(p).\,(c_1^{\dagger} \mid c_3^{\dagger}) + \texttt{com}_3{}^{\uparrow}?(p).\,(c_1^{\dagger} \mid c_2^{\dagger}))$. With a little abuse of notation, we use $\star P$ for denoting the replication of P, with the usual semantics. We consider one buyer and three sellers. *NewBuyer* creates three instances of the `BuyService` service, one from each seller. The part of each such instances residing at *NewBuyer* can be aborted by suitable messages on c_i, which we assume fresh to the rest of the process. The part of the protocol for `BuyService` that resides at *NewBuyer* (denoted P_i), is similar as before, and is extended with an output signal $\texttt{com}_i$ which allows to commit the selection of seller i. The commitment to one particular seller (and the discard of the rest) is implemented by process *Control* using an exclusive (guarded) choice. The duration of *NewBuyer* is given by *tmax*; its compensation activity (*CancelOrder*) is left unspecified. As for Seller_i, it follows the lines of the basic scenario, extended with compensation signals y_i which trigger the compensation process $\mathit{CancelSell}_i$.

This extended example illustrates two of the features of C^3: explicit conversation abortion and conversations bounded in time. The first one can be appreciated in the selection implemented by *Control*, which ensures that only one provider will be able to interact with *NewBuyer*, by explicitly aborting the conversations at *NewBuyer* with the other two providers. However, *Control* only takes care of the interactions at the buyer side; there are also conversation pieces residing at each Seller_i, which are not under the influence of *Control* (we assume $c_i \neq w_i$)[1]. The "garbage-collection" of such pieces is captured by the second feature: since such conversations are explicitly defined with the time bound w_i, we are sure that after a fixed amount of time they will be automatically collected (i.e. aborted). That is, the passing of time avoids "dangling" conversation pieces. This simple example reveals the complementarity between the explicit conversation abortion (achieved via abortion signals) and the more implicit conversation abortion associated to the passing of time.

[1] We could have assumed $c_i = w_i$ so as to allow *Control* to abort conversations at both buyer and seller sides. However, it is somewhat more realistic to assume that *Control* restricts to services in *NewBuyer*.

Even if the study of C^3 is in an early stage, our preliminary results are promising. Up to now we have defined a basic transition semantics for C^3, which extends that of the CC. We have also developed a few compelling examples, which reveal the usefulness of C^3 in the realm of healthcare scenarios. Unsurprisingly, the increased expressiveness of C^3 with respect to the CC comes at the price of a slightly more involved semantics. Preliminary investigations suggest that the definition of, e.g., behavioral equivalences and associated equalities, will require similar considerations and extra care.

2 Syntax and Semantics

Syntax. As discussed before, the syntax of C^3 extends the CC (as introduced in [3] with timed, compensable conversation contexts and a process for aborting running conversations. Let $\mathscr{N}$ be an infinite set of names. Also, let $\mathscr{V}$ and $\mathscr{L}$ be infinite sets of variables and labels, respectively. Using d to range over $\uparrow$ and $\downarrow$, the set of actions α and processes P is given as follows:

$$\alpha ::= \quad \mathtt{l}^d!(\vec{n}) \mid \mathtt{l}^d?(\vec{x}) \mid \mathbf{this}(x) \quad P,Q ::= \quad P \mid Q \mid \Sigma_{i\in I}\,\alpha_i.\,P_i \mid (\nu n)\,P \mid \mu X.\,P \mid X \mid n \blacktriangleleft [P;Q]^t_\kappa \mid \kappa^\dagger$$

Given $\vec{n} \in \mathscr{N}$ and $\vec{x} \in \mathscr{X}$, actions can be the output $\mathtt{l}^d!(\vec{n})$ or the input $\mathtt{l}^d?(\vec{x})$, as in the π-calculus, assuming $\mathtt{l} \in \mathscr{L}$ in both cases. The *message direction* $\downarrow$ (read "here") decrees that the action should take place in the *current* conversation, while $\uparrow$ (read "up") decrees that the action should take place in the *enclosing* conversation. The context-aware prefix $\mathbf{this}(x)$ binds the name of the enclosing conversation context to x. Processes include parallel composition, a guarded choice construct, name restriction, and recursive processes, all with their standard π-calculus interpretation. Notions of free and bound names of a process P, denoted $fn(P)$ and $bn(P)$, are also as expected. Given $\Sigma_{i\in I}\,\alpha_i.\,P_i$, we write $\mathbf{0}$ when $|I| = 0$, and $P_1 + P_2$ when $|I| = 2$. Moreover, the set of processes includes the conversation context $n \blacktriangleleft [P;Q]^t_\kappa$ and the abortion process $\kappa^\dagger$, where $n, \kappa \in \mathscr{N}$ and $t \in \mathbb{N}_0 \cup \{\infty\}$. As in the CC, notice that labels in $\mathscr{L}$ are not subject to restriction or binding.

Transition Semantics. We give the operational semantics of C^3 in terms of the labelled transition system (LTS) in Figure 1. It is an extension of the LTS proposed for the CC [15, 3]. The LTS features transitions of the form $P \xrightarrow{\lambda} P'$, with the usual meaning: P is able to perform *transition label* λ and then evolve to P'. Transition labels are defined in terms of *actions* σ, as defined by the following grammar:

$$\sigma ::= \tau \mid \mathtt{l}^d?(\vec{x}) \mid \mathtt{l}^d!(\vec{n}) \mid \mathtt{this} \mid \kappa^\dagger \qquad \lambda ::= \sigma \mid c\,\sigma \mid (\nu n)\,\lambda$$

Action τ denotes internal communication, while $\mathtt{l}^d?(\vec{x})$ and $\mathtt{l}^d!(\vec{n})$ represent an input and output to the environment, respectively. Action `this` represents a conversation identity access. The new action $\kappa^\dagger$ represents the throwing of an abortion message. A transition label λ can be either the (unlocated) action σ, an action σ *located at* conversation c (written $c\,\sigma$), or a transition label in which n is bound with scope λ. This is the case of bounded output actions. We therefore have notions of free/bound names in a transition label, denoted $fn(\lambda), bn(\lambda)$, and defined as expected. Also, we use $out(\lambda)$ to denote the names produced by a transition, so $out(\lambda) = n$ if $\lambda = \mathtt{l}^d!(n)$ or if $\lambda = c\mathtt{l}^d!(n)$ and $c \neq n$. A transition label λ denoting communication, such as $\mathtt{l}^d?(\vec{x})$ or $\mathtt{l}^d!(\vec{n})$ is subject to *duality* $\overline{\lambda}$. We write $\overline{\mathtt{l}^d?(\vec{x})} = \mathtt{l}^d!(\vec{n})$ and $\overline{\mathtt{l}^d!(\vec{n})} = \mathtt{l}^d?(\vec{x})$.

The LTS relies on the following definition, which formalizes time passing for conversation contexts:

Definition 2.1 (Time-elapsing function)**.** *$\phi(P)$ denotes the function from processes to processes decreasing the time durations in P by 1, given by:*

$$\phi((\nu n)\,P) = (\nu n)\,\phi(P) \qquad \phi(Q \mid R) = \phi(Q) \mid \phi(R) \quad \phi(n \blacktriangleleft [Q;R]^{t+1}_\kappa) = n \blacktriangleleft [\phi(Q);R]^t_\kappa$$
$$\phi(n \blacktriangleleft [Q;R]^0_\kappa) = n \blacktriangleleft [Q;\phi(R)]^0_\kappa \quad \phi(P) = P \;\; \textit{Otherwise.}$$

$$(\textsc{In})\ \mathtt{1}^d?(x).\,P \xrightarrow{\mathtt{1}^d?(\vec{n})} P[\vec{n}/\vec{x}] \quad (\textsc{Out})\ \mathtt{1}^d!(n).\,P \xrightarrow{\mathtt{1}^d!(\vec{n})} P \quad (\textsc{This})\ \mathbf{this}(x) \xrightarrow{c\ \mathtt{this}} P[c/x]$$

$$(\textsc{Abort})\ \kappa^\dagger \xrightarrow{\kappa^\dagger} \mathbf{0} \quad (\textsc{Res})\ \frac{P \xrightarrow{\lambda} Q \quad n \notin n(\lambda)}{(\nu n)\,P \xrightarrow{(\nu n)\,\lambda} (\nu n)\,Q} \quad (\textsc{Sum})\ \frac{\alpha_j.\,P_j \xrightarrow{\lambda} P' \quad j \in I}{\Sigma_{i\in I}\,a_i.\,P_i \xrightarrow{\lambda} P'}$$

$$(\textsc{Open})\ \frac{P \xrightarrow{\lambda} Q \quad n \notin out(\lambda)}{(\nu n)\,P \xrightarrow{(\nu n)\,\lambda} Q} \quad (\textsc{Close1})\ \frac{P \xrightarrow{(\nu n)\,\overline{\lambda}} P' \quad Q \xrightarrow{\lambda} Q' \quad n \notin fn(Q)}{P \mid Q \xrightarrow{\tau} (\nu n)\,(P' \mid Q')} \quad (\textsc{TauPar1})\ \frac{P \xrightarrow{\tau} P'}{P \mid Q \xrightarrow{\tau} P' \mid Q}$$

$$(\textsc{Par1})\ \frac{P \xrightarrow{\lambda} P' \quad bn(\lambda)\#fn(Q) \quad \lambda \neq \tau}{P \mid Q \xrightarrow{\lambda} P' \mid \phi(Q)} \quad (\textsc{Comm1})\ \frac{P \xrightarrow{\lambda} P' \quad Q \xrightarrow{\overline{\lambda}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \quad (\textsc{Rec})\ \frac{P[X/\mu X.\,P] \xrightarrow{\lambda} Q}{\mu X.\,P \xrightarrow{\lambda} Q}$$

$$(\textsc{Fail})\ \frac{P \xrightarrow{\kappa^\dagger} P'}{P \mid c \blacktriangleleft [Q;R]^t_\kappa \xrightarrow{\kappa^\dagger} P' \mid c \blacktriangleleft [Q;R]^0_\kappa} \quad (\textsc{Comp})\ \frac{Q \xrightarrow{\lambda} Q'}{c \blacktriangleleft [P;Q]^0_\kappa \xrightarrow{\lambda} c \blacktriangleleft [P;Q']^0_\kappa}$$

$$(\textsc{LocL})\ \frac{P \xrightarrow{\lambda^\downarrow} P' \quad \kappa \notin fn(\lambda)}{c \blacktriangleleft [P;Q]^{t+1}_\kappa \xrightarrow{c\ \lambda^\downarrow} c \blacktriangleleft [P';Q]^t_\kappa} \quad (\textsc{HereL})\ \frac{P \xrightarrow{\lambda^\uparrow} P' \quad \kappa \notin fn(\lambda)}{c \blacktriangleleft [P;Q]^{t+1}_\kappa \xrightarrow{\lambda^\downarrow} c \blacktriangleleft [P';Q]^t_\kappa}$$

$$(\textsc{ThisClose1})\ \frac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{(\nu n)\,c\,\overline{\sigma}} Q'}{P \mid Q \xrightarrow{c\ \mathtt{this}} (\nu n)\,(P' \mid Q')} \quad (\textsc{ThisComm1})\ \frac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{c\,\overline{\sigma}} Q'}{P \mid Q \xrightarrow{c\ \mathtt{this}} P' \mid Q'}$$

$$(\textsc{ThruL})\ \frac{P \xrightarrow{c\ \lambda^\downarrow} P' \quad \kappa \notin fn(\lambda)}{c \blacktriangleleft [P;Q]^{t+1}_\kappa \xrightarrow{c\ \lambda^\downarrow} c \blacktriangleleft [P';Q]^t_\kappa} \quad (\textsc{TauL})\ \frac{P \xrightarrow{\tau} P'}{c \blacktriangleleft [P;Q]^t_\kappa \xrightarrow{\tau} c \blacktriangleleft [P';Q]^t_\kappa}$$

$$(\textsc{ThisLocL})\ \frac{P \xrightarrow{c\ \mathtt{this}} P'}{c \blacktriangleleft [P;Q]^t_\kappa \xrightarrow{\tau} c \blacktriangleleft [P';Q]^t_\kappa}$$

Figure 1: An LTS for C^3.

We now give intuitions on the rules in Figure 1. As a convention, rules ending in "1" have a symmetric counterpart which is elided. Similarly, for each one of the all the left rules—which describe evolution in the default behavior of conversation contexts, and have names ending in "L"—, there is a set of right rules describing evolution in the compensation behavior. Apart from rules (ABORT), (THIS), and (TAUPAR1), the first four rows in Table 1 collect basic transition rules for a π-calculus with recursion. The distinction between (TAUPAR1) and (PAR1) captures the fact that time passes only as a result of visible actions. While rule (FAIL) formalizes the abortion of a conversation context, rule (COMP) represents the evolution of the compensated part of the conversation context. Rule (LOCL) locates an action to a particular conversation context, and rule (HEREL) changes the direction of an action occurring inside a context. Rules (THISCLOSE1) and (THISCOMM1) are located versions of (CLOSE) and (COMM), respectively. Rules (THRUL) and (TAUL) formalize the way actions change when they "cross" a conversation context. Rule (THISLOCL) hides an action occurring inside a conversation context.

Let us illustrate the LTS of C^3 by revisiting the extended purchase scenario discussed in the introduction. We describe the evolution of the system where the buyer invokes $Seller_1$ with an expected response time of two time units. We recall the definition of the complete system:

$$\begin{array}{l} \mathit{NewBuyer} \blacktriangleleft [\prod_{i\in\{1,2,3\}} \mathbf{new}\ \mathit{Seller}_i \cdot \mathtt{BuyService}\ \mathbf{with}\ (e_i,v_i) \Leftarrow \{P_i;Q_i\} \mid \mathit{Control};\mathit{CancelOrder}]^{tmax}_x \\ \mid \prod_{i\in\{1,2,3\}} \mathit{Seller}_i \blacktriangleleft [\mathit{PriceDB} \mid \mathbf{def}\ \mathtt{BuyService}\ \mathbf{with}\ (b_i,w_i) \Rightarrow \{\mathtt{offer}^\downarrow?(\mathit{prod}).\mathtt{askPrice}^\uparrow!(\mathit{prod}). \\ \qquad\qquad \mathtt{priceVal}^\uparrow?(p).\,\mathtt{price}^\downarrow!(p). \\ \qquad\qquad \mathbf{join}\,\mathit{Shipper} \cdot \mathtt{DeliveryService} \Leftarrow \mathtt{product}^\uparrow!(\mathit{prod});R_i\}; \\ \qquad\qquad \mathit{CancelSell}_i]^{t_i}_{x_i} \\ \mid \mathit{Shipper} \blacktriangleleft [\mathbf{def}\ \mathit{DeliveryService}\ \mathbf{with}\ (d,t) \Rightarrow \{\mathtt{product}^\downarrow?(p).\,\mathtt{details}^\downarrow!(\mathit{data});T\};\mathbf{0}]^{t_3}_z \end{array}$$

where $P_i \triangleq \texttt{offer}^{\downarrow}!(prod).\ \texttt{price}^{\downarrow}?(p).\ \texttt{com}_\texttt{i}{}^{\uparrow}!(p).\ \texttt{details}^{\downarrow}?(d).$ By expanding the definition of **def** and **new**, we have:

$$NewBuyer \blacktriangleleft [(\nu c)\ (Seller_1 \blacktriangleleft [\texttt{BuyService}^{\downarrow}!(c)\,;\mathbf{0}]^{\infty}_{\emptyset} \mid c \blacktriangleleft [P_1\,;Q_1]^{2}_{\nu_1}) \mid S_b \mid Control\,;CancelOrder]^{tmax}_{x}$$
$$\mid Seller_1 \blacktriangleleft [PriceDB \mid \texttt{BuyService}^{\downarrow}?(y).\ y \blacktriangleleft [\texttt{offer}^{\downarrow}?(prod).(\cdots)\,;R_1]^{b_1}_{w_1}\,;CancelSell_1]^{t_1}_{x_1} \mid S_c \mid Shipper$$

where S_b abbreviates the definitions of $Seller_2$ and $Seller_3$ at the buyer side, and S_c is the analogous process at the shippers side. Focusing on *NewBuyer*, we can infer the following transition, using rules (OUT), (RES), (LOCL), and (PAR1):

$$NewBuyer \blacktriangleleft [(\nu c)\ (Seller_1 \blacktriangleleft [\texttt{BuyService}^{\downarrow}!(c)\,;\mathbf{0}]^{\infty}_{\emptyset} \mid c \blacktriangleleft [P_1\,;Q_1]^{2}_{\nu_1}) \mid S_b\,;CancelOrder]^{tmax}_{x}$$
$$\xrightarrow{(\nu c)\ Seller_1\ \texttt{BuyService}^{\downarrow}!(c)} NewBuyer \blacktriangleleft [Seller_1 \blacktriangleleft [\mathbf{0};\mathbf{0}]^{\infty}_{\emptyset} \mid c \blacktriangleleft [P_1\,;Q_1]^{1}_{\nu_1} \mid \phi(S_b)\,;CancelOrder]^{tmax}_{x}$$

which decreases the time bound for conversation context c to 1. The behavior of $Seller_1$ is completely complementary to the above output, as inferred by using (IN), (LOCL), and (PAR1):

$$Seller_1 \blacktriangleleft [PriceDB \mid \texttt{BuyService}^{\downarrow}?(y).\ y \blacktriangleleft [\texttt{offer}^{\downarrow}!(prod).(\cdots)\,;R_1]^{b_1}_{w_1}\,;CancelSell_1]^{t_1}_{x_1}$$
$$\xrightarrow{Seller_1\ \texttt{BuyService}^{\downarrow}?(c)} Seller_i \blacktriangleleft [PriceDB \mid c \blacktriangleleft [\texttt{offer}^{\downarrow}?(prod).(\cdots)\,;R_1]^{b_1}_{w_1}\,;CancelSell_1]^{t_1-1}_{x_1}$$

Given these two transitions, a synchronization can be inferred using rules (CLOSEL) and (PAR1), using c as shared name for *NewBuyer* and $Seller_1$ to communicate. The state of the system is then:

$$(\nu c)\ (NewBuyer \blacktriangleleft [Seller_1 \blacktriangleleft [\mathbf{0};\mathbf{0}]^{\infty}_{\emptyset} \mid c \blacktriangleleft [\texttt{offer}^{\downarrow}!(prod).\ (\cdots)\,;Q_1]^{1}_{\nu_1} \mid \phi(S_b)\,;CancelOrder]^{tmax}_{x}$$
$$\mid Seller_1 \blacktriangleleft [PriceDB \mid c \blacktriangleleft [\texttt{offer}^{\downarrow}?(prod).(\cdots)\,;R_1]^{b_1}_{w_1}\,;CancelS_i]^{t_1-1}_{x_1}) \mid \phi(W)$$

where W represents the rest of the system. At this point, a communication on `offer` between *NewBuyer* and $Seller_1$ becomes possible. Omitting process $Seller_1 \blacktriangleleft [\mathbf{0};\mathbf{0}]^{\infty}_{\emptyset}$, at the buyer side we have:

$$(\nu c)\ (NewBuyer \blacktriangleleft [c \blacktriangleleft [\texttt{offer}^{\downarrow}!(prod).\ \texttt{price}^{\downarrow}?(p).\ (\cdots)\,;Q_1]^{1}_{\nu_1} \mid \phi(S_b)\,;CancelOrder]^{tmax}_{x})$$
$$\xrightarrow{c\ \texttt{offer}^{\downarrow}!(prod)} (\nu c)\ (NewBuyer \blacktriangleleft [c \blacktriangleleft [\texttt{price}^{\downarrow}?(p).\ (\cdots)\,;Q_1]^{0}_{\nu_1} \mid \phi(\phi(S_b))\,;CancelOrder]^{tmax}_{x})$$

while $Seller_1$ makes an input transition located at c, inferred using (OUT) and (LOCL):

$$(\nu c)\ (Seller_1 \blacktriangleleft [PriceDB \mid c \blacktriangleleft [\texttt{offer}^{\downarrow}?(prod).\ \texttt{askPrice}^{\uparrow}!(prod).\ (\cdots)\,;R_1]^{b_1}_{w_1}\,;CancelS_i]^{t_1-1}_{x_1}) \mid \phi(W)$$
$$\xrightarrow{c\ \texttt{offer}^{\downarrow}?(prod)} (\nu c)\ (Seller_1 \blacktriangleleft [PriceDB \mid c \blacktriangleleft [\texttt{askPrice}^{\uparrow}!(prod).\ (\cdots)\,;R_1]^{b_1-1}_{w_1}\,;CancelSell_1]^{t_1-1}_{x_1}) \mid \phi(\phi(W))$$

Again, these complementary transitions can synchronize, thus firing an unobservable transition inferred using rule (COMM). The system then evolves to

$$(\nu c)\ (NewBuyer \blacktriangleleft [c \blacktriangleleft [\texttt{price}^{\downarrow}?(p).\ (\cdots)\,;Q_1]^{0}_{\nu_1} \mid \phi(\phi((S_b))) \mid\ ;CancelOrder]^{tmax}_{x}$$
$$\mid Seller_1 \blacktriangleleft [PriceDB \mid c \blacktriangleleft [\texttt{askPrice}^{\uparrow}!(prod).\ (\cdots)\,;R_1]^{b_1-1}_{w_1}\,;CancelSell_1]^{t_1-1}_{x_1}) \mid \phi(\phi((W))$$

At this point, the default behavior of the system establishes that $Seller_1$ contacts *PriceDB* in order to communicate the price of the selected product to *NewBuyer*. However, we notice that conversation context c inside *NewBuyer* has reached a timeout. As a consequence, the only possible way for progressing is by engaging into the compensating behavior, represented by process Q_1. Assuming $Q_1 \xrightarrow{\lambda} Q_1'$, the evolution of the compensating behavior can be inferred using rule (COMP). We then have:

$$(\nu c)\ (NewBuyer \blacktriangleleft [c \blacktriangleleft [\texttt{price}^{\downarrow}?(p).\ (\cdots)\,;Q_1']^{0}_{\nu_1} \mid \phi(\phi(\phi((S_b)))\,;CancelOrder]^{tmax}_{x}$$
$$Seller_1 \blacktriangleleft [PriceDB \mid c \blacktriangleleft [\texttt{askPrice}^{\uparrow}!(prod).\ (\cdots)\,;R_1]^{b_1-1}_{w_1}\,;CancelSell_1]^{t_1-1}_{x_1})) \mid \phi(\phi(\phi(W)))\ .$$

3 Related Work

Time and its interplay with forms of exceptional behavior do not seem to have been jointly studied in the context of models for structured communication. In our previous work [13] we have studied an LTL interpretation of the session language in [9] and proposed a extension of it with time, declarative information, and a construct for session abortion. The language in [9], however, does not support multiparty interactions. The original presentation of the CC [16] features a try/catch construct which handles compensations in the expected way; this aspect is investigated in detail in [2]. (Subsequent presentations of the CC, in particular those defining type disciplines [3], leave this construct out). The language in [16] appears to be as expressive as the sub-language of C^3 in which durations are all ∞. However, because of the interplay of time and compensations, encoding full C^3 into the CC in [16] appears quite difficult.

Time and exceptional behavior have been considered only separately in orchestrations and choreographies. As for time, Timed Orc [17] introduced real-time observations for orchestrations by introducing a delay operator. Timed COWS [12] extends COWS (the Calculus for Orchestration of Web Services [11]) with operators of delimitation, abortion, and delays; we are not aware of reasoning techniques for Timed COWS. As for exceptional behavior, [8, 4] propose languages for *interactional exceptions*, in which exceptions in a protocol generate coordinated actions between all peers involved. Associated type systems ensure communication safety and termination among protocols with normal and compensating executions. In [4], the language is enriched further with multiparty session and global escape primitives, allowing nested exceptions to occur at any point in an orchestration. Concerning choreographies, [5] introduced an extension of a language of choreographies with try/catch blocks, guaranteeing that embedded compensating parts in a choreography are not arbitrarily killed as a result of an abortion signal.

Our work has been influenced by extensions to the (asynchronous) π-calculus, notably [10, 1]. In particular, the role of the time-elapsing behavior for conversation contexts used in C^3 draws inspiration from the behavior of long transactions in webπ [10], and from the π-calculus with timers in [1]. Notice however that the nature of these languages and C^3 is very different. First, the communication model is different: C^3 is synchronous, while the calculi in [10, 1] are asynchronous. Second, webπ is a language tailored to study long-running transactions, and therefore exceptions in webπ and compensations in C^3 have a completely different meaning, even if both constructs look similar.

4 Concluding Remarks

We have described initial steps towards a joint study of time and exceptional behavior in the context of structured communications. We have presented C^3, a generalization of the Conversation Calculus in which conversation contexts have an explicit duration, a compensation activity, and can be explicitly aborted. Ongoing work concerns equipping C^3 with behavioral equivalences and associated properties in the lines of that proposed for the CC. In particular, we would like to obtain a set of *behavioral equations* (such as those defined for the CC in [16, Prop. 4.3]) as a basic reasoning technique for C^3. Our main interest is in logic-based reasoning techniques for C^3. One option is to exploit linear-temporal logic (LTL) as in our previous work [13]. Examples of the LTL reasoning we have in mind include properties such as "every process failed is followed by a compensation activity", "the specification always compensates in case of failure" or, in the realm of healthcare scenarios, properties such as "when the doctor is unavailable, an urgency treatment will be covered by the head nurse, within 20 minutes". In fact, for C^3 one could follow pretty much the approach we detailed in [13], namely to provide encodings relating languages for structured communication (C^3, in our case) to models or languages with a connection with logic. However, one drawback of this (indirect) approach is that encodings are typically untyped, and so one would have to define them carefully in order to avoid the loss of important information about possible

projections in C^3. We are also considering the definition of a modal logic describing behaviors between conversations, as the one proposed for the choreography language in [6]. A parallel line of future work concerns studying how already proposed type disciplines for the CC can be adapted/extended so as to exploit the additional information on time and exceptional behavior available in C^3 descriptions.

Acknowledgements. We are grateful to Thomas Hildebrandt and Hugo T. Vieira for their useful comments and suggestions. This research has been supported by the Trustworthy Pervasive Healthcare Services (TrustCare) project - Danish Research Agency, Grants # 2106-07-0019 (`www.TrustCare.eu`) as well as by FCT / MCTES (CMU-PT/NGN44-2009-12) - INTERFACES.

References

[1] M. Berger and K. Honda. The two-phase commitment protocol in an extended pi-calculus. *ENTCS*, 39(1), 2000.

[2] L. Caires, C. Ferreira, and H. T. Vieira. A process calculus analysis of compensations. In *Proc. of TGC'08*, volume 5474 of *LNCS*, pages 87–103. Springer, 2008.

[3] L. Caires and H. T. Vieira. Conversation types. *Theor. Comput. Sci.*, 411(51-52):4399–4440, 2010.

[4] S. Capecchi, E. Giachino, and N. Yoshida. Global Escape in Multiparty Sessions. In K. Lodaya and M. Mahajan, editors, *Proc. of FSTTCS'2010*, volume 8 of *LIPIcs*, pages 338–351, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[5] M. Carbone. Session-based choreography with exceptions. In N. Yoshida and V. Vasconcelos, editors, *PLACES'08: Procs. of the 1st. Workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software*, volume 241, pages 35–55. ENTCS, 2008.

[6] M. Carbone, T. Hildebrandt, D. Grohmann, and H. A. López. A logic for choreographies. In *Post-proceedings of PLACES 2010*, 2011. To appear.

[7] M. Carbone, K. Honda, and N. Yoshida. Structured Communication-Centred Programming for Web Services. In *Procs. of the 16th European Symposium on Programming (ESOP'07)*, LNCS, pages 2–17. Springer, 2007.

[8] M. Carbone, K. Honda, and N. Yoshida. Structured Interactional Exceptions in Session Types. In *Procs. of the 19th international conference on Concurrency Theory*, pages 402–417. Springer-Verlag, 2008.

[9] K. Honda, V. Vasconcelos, and M. Kubo. Language Primitives and Type Discipline for Structured Communication-Based Programming. In *7th European Symposium on Programming (ESOP): Programming Languages and Systems*, pages 122–138. Springer-Verlag London, UK, 1998.

[10] C. Laneve and G. Zavattaro. Foundations of web transactions. In *Proc. of FoSSaCS'05*, volume 3441 of *LNCS*, pages 282–298. Springer, 2005.

[11] A. Lapadula, R. Pugliese, and F. Tiezzi. A calculus for orchestration of web services. In *Proc. of 16th European Symposium on Programming (ESOP'07)*, volume 4421 of *LNCS*, pages 33–47. Springer, 2007.

[12] A. Lapadula, R. Pugliese, and F. Tiezzi. COWS: a timed service-oriented calculus. In *Procs. of the 4th international conference on Theoretical aspects of computing*, pages 275–290. Springer-Verlag, 2007.

[13] H. A. López, C. Olarte, and J. A. Pérez. Towards a unified framework for declarative structured communications. In *Proc. of PLACES'08*, volume 17 of *EPTCS*, pages 1–15, 2009.

[14] K. M. Lyng, T. Hildebrandt, and R. R. Mukkamala. From paper based clinical practice guidelines to declarative workflow management. In *2nd International Workshop on Process-oriented information systems in health- care (ProHealth 08)*, pages 336–347, Milan, Italy, September 2008.

[15] H. T. Vieira. *A Calculus for Modeling and Analyzing Conversations in Service-Oriented Computing*. PhD thesis, Universidade Nova de Lisboa, 2010.

[16] H. T. Vieira, L. Caires, and J. C. Seco. The conversation calculus: A model of service-oriented computation. In *Proc. of ESOP'08*, volume 4960 of *LNCS*, pages 269–283. Springer, 2008.

[17] I. Wehrman, D. Kitchin, W. R. Cook, and J. Misra. A timed semantics of orc. *Theor. Comput. Sci.*, 402(2-3):234–248, 2008.