Specifying and Implementing Multi-Party Conversation Rules with Finite-State-Automata

Maira Gatti de Bayser, Melina Alberio Guerra, Paulo Cavalin, Claudio Pinhanez
IBM Research, Rio de Janeiro, Brazil

Abstract
Current existing chatbot engines do not properly handle a group chat with many users and many chatbots. This prevents chatbots from developing their full potential as social participants. This happens because there is a lack of methods and tools to design and engineer conversation rules. The work presented in this paper has two major contributions: the presentation of a Finite-State-Automata-based DSL (Domain Specific Language), called DSL-CR, for engineering multi-party conversation rules for inter-message coherence to be used by chatbot engines; and its usage in a real-world dialogue problem with four bots and humans. With this tool, the amount of domain and programming expertise needed for creating conversation rules is reduced, and a larger group of people, like linguists, can specify the conversation rules.

Introduction
Nowadays, chatbot engines are enabling chatbots to participate in multi-party conversations but they do not properly handle a group chat with either more than one user or more than one chatbot (Gatti de Bayser et al. 2017). In these types of conversations, natural language becomes the communication protocol exchanged not only by the human users but also among the chatbots themselves.

Turn-taking is the process in which a participant in a conversation determines when it is appropriate to make an utterance and how that is accomplished. Although turn-taking is important in dyadic conversations, it becomes a much more important and complex issue in multi-party conversations. Unlike two-party dialogues, in multi-party dialogues a participant has to determine not only whether the last speaker has finished talking but also who among the participants is the best candidate to be next speaker. There are many social settings, such as assemblies, debates, one-channel radio communications, and some formal meetings, where there are clear and explicit rules of who, when, and for how long a participant can speak.

Chat communication in particular is one of the primary areas of interest in Computer-Mediated Discourse Analysis (Herring 1999) (Holmer 2008), since chat discussions are very different from face-to-face discussions. In this context, the aim of Discourse Structure Analysis is to provide an approach that combines different methods in a comprehensive and extensible way and is implemented in software for automation (Holmer 2008). Adjacent pairs of messages are used to create the discourse structure, which may consist of several branched threads. In this area, the task of analyzing inter-message coherence is an ongoing research work and the rules for detecting references still need to be formalized and implemented. And considering chatbot engines, there is a need of tools for engineering inter-message coherence as conversation rules to be used by chatbots.

Domain-specific languages (DSLs) are languages tailored to a specific application domain and they offer substantial gains in expressiveness and ease of use compared with general-purpose programming languages (GPPL) in their domain of application, like Java or Python. DSLs trade generality for expressiveness in a limited domain (Mernik, Heering, and Sloane 2005). In this work, the domain is chat-oriented dialogue and we present the DSL-CR, a Domain Specific Language for Conversation Rules, for modeling, specifying and enforcing inter-message coherence in multi-party conversation. It is important to note that the DSL-CR is not a language or protocol for communication among software agents as KQML (Finin et al. 1994) or FIPA ACL. However, it does follow the FIPA ACL standard for the message structure.

We have tested the DSL-CR usage in a real-world application called finch, a finance advisor group chat with four collaborative chatbots, with hundreds of users. With less than 25 rules that we created, we achieved high reliability. We believe this language provides a foundation for creating sophisticated tools to a larger group of people, especially chatbot developers and linguists to specify multi-party conversation rules.

Challenges in Multi-Party Conversation
To illustrate some of the challenges in multi-party chattering, we present in Table 1 an example of dialogues in which the participants need knowledge of when to speak given the context of the dialog.

The turn-taking process or inter-message coherence is described here as permissions, obligations and prohibitions

\footnote{FIPA Agent Communication Language is a standard message specification for heterogenous agents communication.}
which change at every turn. These three concepts were first described for reasoning about ideal behaviour in Deontic Logic (von Wright 1951). The concepts of deontic logic have traditionally been used for the analysis and reasoning about normative law and are widely used in normative multi-agent systems (MASs) to formally describe norms and their modalities since early 1990s (Meyer 1993) (Meyer and Wieringa 1993) and a good state of the art has been published in 2017 with the work done since then (Santos et al. 2017). In the context of conversation, we consider that for a given turn:

A **permission** allows the participant to emit utterances if she wants to;

An **obligation** requires the participant to pro-actively or re-actively emit an utterance;

A **prohibition** forbids the participant to emit utterances or just states that they are not expected in that turn.

Table 1 presents a dialogue in a party where Maria announces the time of the party (turn 1). When that happens, although it is not a question, it is implicit that Maria expects that everyone acknowledges the message and confirms if they are coming and on time. This is expressed by the expression (in the **Obl** column) that "All but Maria" are obliged to respond Maria’s utterance. Turn 2 and 3 show replies from Max and Birgit. Note that after they have answered, the obligation for them to respond to Maria does not hold anymore. Since Anna and Leo did not respond, Maria addresses a message to them on turn 4. Anna answers in turn 5 what removes obligations to her, and Maria answers Anna in turn 6. Leo is still obliged to answer, but he sends a question instead, mentioning Maria. She then receives an obligation to reply while Leo still have the obligation to reply to Maria’s utterance on turn 1. Maria answers Leo in turn 8, releasing her obligation. Then Leo confirms on turn 9, finally releasing his obligation and given permission to all participants to answer. Maria ends the dialogue in turn 10.

It is important to notice that social behavior in the context of conversations diverges in different cultures. Therefore, the set of permissions, obligations and prohibitions defined for the party dialogue is not universal and agreed by everyone. Hence, there is a need to extract patterns from the utterances in order to create a few set of more generic conversation rules which can govern a large set of dialogues. Table 2 illustrates some generic conversation rules which could be used to govern the dialogue presented in Table 1.

To represent those conversation rules in a more structured way we suggest the use of two auxiliary structures, *speech acts* and *norms*. *Speech acts* (aka dialogue acts or functional moves by linguists) are mainly studied in Philosophy of Language, and in (Austin 1975), Austin focus on the function of language rather than semantics. He calls performative utterances or, for short, performative, the utterances which indicate that an action is performed. Hence, the speech acts define the action of the performative and it effect changes both the observable world, as well as the mental states of dialogue participants. Speech acts are widely used in agent communication languages since the late 1990s (Traum 1999).

Table 2: Generic Conversation Rules

<table>
<thead>
<tr>
<th>Trigger</th>
<th>Pre-Conditions</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>On utterance sent</td>
<td>The utterance mentions a collective word (like <em>folks</em>)</td>
<td>All participants have obligation to answer except the sender.</td>
</tr>
<tr>
<td>On utterance sent</td>
<td>Utterance’s speech act is <em>query</em> and utterance mentions explicitly a participant</td>
<td>The participant has obligation to reply and the others are prohibited to answer except the sender.</td>
</tr>
<tr>
<td>On utterance sent</td>
<td>Utterance’s speech act is <em>inform</em> and sender had the obligation</td>
<td>The participant obligation is deactivated.</td>
</tr>
</tbody>
</table>

The set of conversation rules described Table 2 also considers that it is possible to define somehow if the utterance is mentioning one or more participants.

**Background and Related Work**

Utterance’s intent detection (i.e., the main purpose or goal of the utterance) is a research problem that has been tackled for the last decades (see Allen et al. (Allen and Perrault 1980), for instance), and nowadays chatbot engines (as IBM Watson Conversation Service) tackles through supervised learning using Support Vector Machines (SVM) or Recurrent Neural Networks (Deep Learning). On the other hand, turn-taking in generic, multi-party spoken conversation has been studied by, among others, Sacks et al. (Sacks, Schegloff, and Jefferson 1974). In broad terms, it was found that
participants in general do not overlap their utterances and that the structure of the language and the norms of conversation create specific moments, called transition-relevance places or TRPs, where turns can occur. In everyday conversations, the last utterances make clear to the participants who should be the next speaker (selected-next-speaker, or SNS), and he or she can take that moment to start to talk. Otherwise, any other participant can start speaking, with preference for the first starter to get the turn; or the current speaker can continue (Sacks, Schegloff, and Jefferson 1974). In its simplest form, a vocative such as the name of the next speaker is uttered. Also, there is a strong bias towards making the previous speaker before the current being the most likely candidate to be the next speaker. On computer-mediated conversation, on the other hand, overlap is a problem in group communication between humans: unrelated messages happens in a likelihood proportional to the number of active participants involved in the communication (Herring 1999).

The detection of TRPs and of the SNS is still a challenge for speech-based machine conversational systems. However, in the case of text message chats, TRPs are often determined by the acting of posting a message, so the main problem facing multi-party-enabled textual chatbots is in fact determining whether there is and who is the selected-next-speaker. In other words, *chatbots have to know when not to speak.*

Bohus and Horowitz (Bohus and Horvitz 2011) have proposed a computational probabilistic model to detect TRPs in speech-based systems. Liu (Liu, Xu, and Sarikaya 2015) proposed a unified multi-turn multi-task spoken language understanding (SLU) solution capable of handling multiple context sensitive classification (intent determination) and sequence labeling (slot filling) tasks simultaneously. The proposed architecture is based on recurrent convolutional neural networks (RCNN) with shared feature layers and globally normalized sequence modeling components. In (Sukhbaatar et al. 2015), a recurrent neural network (RNN) architecture (where the recurrence reads from a possibly large external memory multiple times before outputting a symbol) is presented. But they apply it to textual reasoning tasks, rather than multi-party turn-taking.

Since we are not aware of any work dealing with modeling turn-taking in textual chats for multi-party conversation, we present in this work a first step using Finite-State Automata. We believe this approach gives a foundation for using Hidden Markov Models (HMM) (Wright 1998), or Partially Observable Markov Decision Process (POMDP) (Roy, Pineau, and Thrun 2000) once we have a large multi-party conversation corpus.

In the state of the art of open multi-agent systems though, there are several works related to ours (Minsky and Ungureanu 2000) (Murata and Minsky 2003) (Arcos et al. 2005). In this case, those systems are built upon a self-governed architecture often codified through laws. A law is a set of rules to govern distributed agents interaction. Marvin Minsky published a seminal work describing the Law-Governed Interaction (LGI) conceptual model about the role of interaction laws on distributed systems (Minsky and Ungureanu 2000) and conducted further work and experimentation based on those ideas (Murata and Minsky 2003).

The XMLaw description language and the M-Law framework (Paes et al. 2005) (Paes et al. 2007) were proposed and developed to support law-governed systems, still at the agent level but with more social capabilities. They implement a law enforcement approach as an object-oriented framework which allows normative behavior through the combination between norms and clocks. The M-Law framework works by intercepting direct messages exchanged between agents, verifying the compliance of the messages with the laws, and subsequently forwarding the message to the real addressee if the laws allow it. If the message is not compliant then the mediator blocks the message and applies the consequences specified in the law, if any. A problem with XMLaw is that it supports the specification in the context of scenes within an electronic institution or organization, i.e., no support for dialogues expressivity nor has been tested for that.

With regard to chatbot engines, there is a lack of research or systems directed to governing turn-taking in conversation scenarios. To the best of our knowledge, the work proposed in this paper is the first to support the specification and implementation of social chatbots in a multi-party conversation. Furthermore, this paper presents DSL-CR formalized as a Finite-State Automaton (as described later) which can be used for the validation of conversation rules and for the verification of conversation rules conflicts.

The DSL-CR

To describe the DSL-CR we start by presenting its grammar. We follow by showing how to convert a set of rules described in the DSL-CR grammar into a finite-state automaton which can be used to model and express a set of DSL-CR rules.

DSL-CR Grammar

The DSL-CR grammar which has Conversation as the main concept. A conversation is composed of at least three concepts: Participant, Role and Protocol. The Participant is an agent (a chatbot or a person), while the Role defines the role of the participant in the conversation. The Role concept is used in the grammar because it gives expressivity for defining rules, borrowing from work on multi-agent systems where roles are often employed to specify interaction rules between generic agents. We leverage here a family of reference models for role-based access control (RBAC) (Sandhu et al. 1996), in which permissions are associated with roles, and users are participants with appropriate roles. While a user in an RBAC model is a human being, in our DSL-CR model a user can be a human being or a chatbot. While in an RBAC model a role is a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role (Sandhu et al. 1996), in the DSL-CR grammar model a role is a function within the conversation but also carrying semantics regarding the authority and responsibility conferred to the participant who has that role.

Before describing the Protocol concept, it is important to understand the Norm concept, which is defined by behavioral patterns which guide and control the performance of
actions within a specific context (Santos et al. 2017). In the case of the work herein presented, the conversation represents the context and the norm modality can be a permission, obligation, or prohibition (von Wright 1951). Whenever a Message, that is, a FIPA ACL message which contains the utterance, is exchanged in the conversation, the norms which are currently active in the conversation are checked and, if the utterance is allowed according to these norms, it can (or must in case of obligations) be uttered by the participant. This verification is made using the role of the participant which is specified in the utterance. The message context features which can be used to create fine-grained rules depend on the communication model and may depend on the domain.

We also define as a Turn in the conversation as the exchange by one single participant of one message which contains one or more utterances. A turn in DSL-CR represents an event that can change the state of the conversation or the set of norms that are active in the conversation such as an utterance arrival. In real-world human conversation, pauses and silence also carry meaning. So DSL-CR also allows for mechanisms based on lapsed time to trigger changes in the norms of a conversation. For the lapsed time event, the DSL-CR defines the Clock concept. It is defined in the scope of the conversation and is used to define time-behavior turns.

We have added two features in the conceptual model which we believe to be domain-independent but specific to conversations: Speech Act and Topic. While the speech act encapsulates the action of the utterance, the topic only represents its subject. The DSL-CR permits one or more third-party message classifiers to be used to detect in the exchanged utterance speech acts or topics which in turn can be used, if needed, in the conversation rules. Therefore conversation rules like the following can be created with this grammar to express the rules governing a multi-bot financial adviser system described later in the paper:

R1. Whenever an utterance with the inform or query speech act and the dollar topic is sent, a participant with the investment role has the permission to reply.

Conversation rules specifications may have several rules like these and the total number of possible rules in this format is bound by number of speech acts, topics (or any other message feature) and roles defined. Here is another example of a DSL-CR rule, in this case based on lapsed time:

R2. After $\Delta t$ minutes, every prohibited participant becomes permitted.

For the rule R1, the reader may notice the usage of the word Whenever which means every time that. So on every conversation turn with the specified speech acts and topics, the R1 rule holds. In the case of rule R2, it defines a $\Delta t$ time which replaces the prohibition norm that was active for a participant by a permission. In this case the turn is defined by the time not by any aspect of the last utterance.

**DSL-CR Rules as a Finite-State Automaton**

The Protocol concept encapsulates the interaction patterns of a conversation, including all conversation rules. We designed it to be specified through a finite-state automata (FSA) (Booth 1967) due to the fact that it provides a formalism for further conversation rules verification. Note that the Protocol should not be confused with communication protocols standards as FIPA ACL (for Intelligent Physical Agents 2002) or KQML (Finin et al. 1994) which is a language and protocol for communication among heterogeneous software agents and define the message structures to be followed by the agents developers.

That said, each Protocol contains one FSA. The automaton may for instance be defined to take the exchanged utterance into account in order to decide what the new conversation protocol state is, and to activate or deactivate certain norms. That way, a predefined set of norms is active, whenever a particular event is recognized by the system. In order to modularize and foster reuse, a conversation may have several protocols which specify the rules which govern the conversation among the participants in different phases of the conversation, but only one protocol is active at any time.

The formal definition of a conversation protocol in DSL-CR is given by the tuple:

$$ T = (U, R, P, N, C, NS, \delta, s_0, F), $$

where: $U$ is a (finite) messages (FIPA ACL) set; $R$ is a (finite) set of roles which participants can take; $P$ is a (finite) participants set; $N$ is a (finite) norms set; $C$ is a (finite) clock set; $NS$ is a (finite) set of states $\{NS_1, NS_2, ..., NS_n\}$ where each $NS_i$ is a set of norms, $NS_i \subseteq N$; $\delta$ is a state transition function $\delta : NS \times (U \cup C) \rightarrow NS$; $s_0$ is an initial state, $s \in NS$; and $F$ is a set of final states, $F \subseteq NS$.

For brevity, we consider that a conversation with no transition function for a given message implies the conversation state remains the same. Given a set of rules, they can be translated into different protocols (i.e., FSA) with the same semantics. Different strategies can be used to create the FSA. It could be generated having only two states: one that represents that no norm is active and one that represents that there is at least one active norm in the conversation, and each rule will create a transition between these two states. Or the FSA could be generated by creating a new state whenever there is a different norm set. These two approaches try to minimize the number of states, although one could also generate a new state whenever a rule is added.

Figure 1 below illustrates a FSA for the party dialogue presented in Table 1. The following rules were used to create the protocol:

DPR1. Whenever an utterance with query speech act act and one or more participants are mentioned is sent, the mentioned participants have an obligation to reply and the others are prohibited.

DPR2. Whenever an utterance with inform speech act is sent by a participant with an obligation, the participant who sent has no obligation to reply.

DPR3. When no participants have obligations to reply, all participants have a permission to send utterances.

In conversation protocol of Figure 1, the conversation starts on empty ($\emptyset$) state and when the first utterance is sent
(turn 1), transition \( t_1 \) is activated and the conversation state changes to \( NS \) state. The \( NS \) state can hold all the obligations and prohibitions depending on the activated transition.

Let \( u \in U \) be the features of the utterance (speech act, topic, mentions, etc), which means that utterances sent in different turns can be categorized by one utterance "type". In order to better illustrate the mapping between the turns, we illustrate \( u_i \) as the utterances in each turn. Therefore \( u_1 \) means the utterance of turn 1. Similarly, \( t_j \) corresponds a particular transition which activates and deactivates norms and their scopes. Therefore, the walk on the protocol FSA of Figure 1 for the party dialogue would be:

\[
\emptyset \xrightarrow{u_{1, t_1}} NS \xrightarrow{u_{2, t_2}} NS \xrightarrow{u_{3, t_3}} NS \xrightarrow{u_{4, t_4}} NS \xrightarrow{u_{5, t_5}} \emptyset
\]

\[
\emptyset \xrightarrow{u_{6, t_6}} NS \xrightarrow{u_{7, t_7}} NS \xrightarrow{u_{8, t_8}} NS \xrightarrow{u_{9, t_9}} \emptyset \xrightarrow{u_{10, t_{10}}} \emptyset
\]

There are three types of utterances specified as messages. The features that characterize the messages are name_mention and speech_acts. There are four norms that can be activated and deactivated by five transitions.

Given that the number of possible conversation rules can be high, the probability of creating conflicts when they becomes high too. A conflict happens when a prohibition and obligation, or a prohibition and permission are given at the same time to the same participant. We designed the DSL-CR to tackle the conflict at the execution time as the following: an obligation always has precedence over a prohibition and a prohibition always has precedence over a permission.

### Case Study

The DSL-CR protocols can be used both in a decentralized and in a centralized scenarios to enable multi-party conversations. In the former, it can be used to help building individual chatbots which are aware of the multi-party environment they belong to; in the latter, there is a central point that enforces that every chatbot replies when needed and allowed. In both scenarios there is a chat hub mediating the exchange of messages between chatbots and users. However, in the first scenario the DSL-CR is called by the chat hub, while in the second it is called by each chatbot.

The conversation starts when the chat group is created. Then either the owner of the chat group has to inform the conversation rules specification during the creation of the dialogue, or the chat hub is configured with the correct conversation rules for each dialogue. The utterance is then broadcasted to all participants, if allowed.

### finch: a Multi-bot Investment Adviser

We have tested the DSL-CR in a real world system called finch, a Brazilian Portuguese-based interactive investment adviser which helps users to make more informed financial decisions. In Brazil, the typical customer is mass affluent, less than 40 years old, digitally enabled, and not familiar with finances. The strategic intent is to capture wealth advice market by pro-active handling the users’ lack of financial knowledge and uneasiness with money. Following a user-centered design process, we designed the adviser systems as a chat where a user can converse with four chatbots³ representing three investment products and one for the overall support of a friendly finance investment counselor.

For this conversation the goal is to have the investment counselor bot always mediating the conversation with the user while the investment products experts only replying when required. They are required either when they are addressed directly or when the utterance’s topic is about their expertise. In addition, the chatbot which spoke last is encouraged to keep answering the user, except when the user sends utterances related to simulation of the return of investment which is always mediated by the investment counselor.

#### Implementation

In this scenario, we created 15 norms of which 6 were prohibitions, 2 were permissions, and 7 were obligations. These norms were used in 25 transitions, either by activation or deactivation. Below we list the rules that use a subset of these norms and create the initial state of the conversation:

**FR1.** Whenever the chat group is created, the participant with the investment role has the obligation to reply to utterances. (finch rule)

**FR2.** Whenever the chat group is created, the participant with the investment product expert role has the prohibition to reply to utterances. (finch rule)

**FR3.** Whenever the chat group is created, the owner of the chat has a permission to send utterances. (generic rule)

**FR4.** Whenever the chat group is created, the participant with the user role has a permission to send utterances. (generic rule)

These rules were defined because we wanted the investment counselor to start the conversation with the user and we did not want the investment product advisers to reply to the user or to the investment counselor even though they were in the chat group. These four rules, FR1 to FR4, define that the conversation protocol starts with 4 active norms. Note that if the sender is a participant with the investment product adviser role, we have a conflict: the participant will have a prohibition and permission. However, in finch, we specified that whenever there is a conflict between prohibitions and obligations or permissions, the prohibition holds.

We designed the conversation protocol to contain only 2 states, just like in the party dialogue protocol described before. However, the initial state of the finch conversation protocol is one with active norms because the conversation starts with the counselor having the obligation to talk. Below

³We used IBM Watson Conversation System.
are some other finch rules which are implemented in the 21 transitions:

FR5. Whenever an utterance with greetings or inform speech act is sent, the participant with the investment role has the obligation to reply and the participant with the investment product expert role has a prohibition to reply and the sender has a permission to send more utterances. (finch rule)

FR6. Whenever an utterance which mentions a participant is sent or an utterance with a particular topic is detected, the mentioned participant or the participant which knows about that topic, respectively, has an obligation to reply. All others participants are prohibited to reply and the sender has a permission to send more utterances. (finch rule)

FR7. Whenever an utterance which mentions a prohibited participant is sent, the prohibition is deactivated to the mentioned participant and an obligation to reply is activated. All others participants are prohibited to reply and the sender has a permission to send more utterances. (generic rule)

FR8. Whenever an utterance with query calculation speech act is sent, the participant with the investment role has the obligation to reply and the participant with the investment product expert role has a prohibition to reply and the sender has a permission to send more utterances. (finch rule)

FR9. Whenever an utterance with inform calculation speech act is sent, the sender which is a participant with the investment product expert role has no longer the obligation to reply. The sender has a prohibition to send more utterances. (finch rule)

Although there are a few more rules specified in finch, they are not important in the context of this paper. However, we want to give an insight on the rationale for specifying them using the DSL-CR language and the language expressivity. The norm applicability can be constrained, if desired, to the sender, receiver or last sender, i.e., three options. Since that constraint can be empty, or contains one or more option, and even more, may be a negation of one of the options (for instance, the norm applies to all except the sender), the number of possibilities specifications without repetition is 16.

We specified nine (9) roles: person, bot, user, mediator, investment product expert, investment counselor, savings account adviser, certificate of deposit adviser and treasury bonds adviser. Since the specification of the roles field may be empty, may contain one or more, or may be a negation, the number of possible specifications is 519. We specified the speech act and topic classes based on the classes recognized by a service which we trained for finch. There were (13) speech acts which were: greetings, thank, query, query calculation, request, inform, inform calculation, query and finish calculation, agree, failure, not understood, bye, undefined. We also develop a topic classifier which handle (6) different topics: investment in general, savings account, certificate of deposit, treasury bonds, all, or undefined.

Twenty seven (27) types of messages were specified. They were sufficient although the number of possible message specifications for finch, considering the number of possibilities for each field, is almost 290 million. As an example of a message that could be specified but makes no sense is one which mentions all topics, is classified with all speech acts, and is sent by a participant who has all roles and mentions another participant.

Finally, considering that there are 5 different participants playing 9 different roles in finch, the total number of possible norms specification is 4674. But not all of them may make sense, for instance, a norm with a prohibition to all roles to send a message and no transition to deactivate it.

**Experiments Results**  
Finch was tested in a large experiment with more than 100 users. One of the authors performed qualitative analysis on 37 dialogues resulting on 57 unique intents detected. Only 2 conversations demonstrated some level of misbehavior with regard to inter-message coherence. This happened because we did not anticipate that the user could ask to simulate the return of an investment for only one investment option. Therefore, when she sent the utterance “I would like to invest RS 15 thousands in a certificate of deposit” two norms were activated: one giving an obligation to the counselor to mediate the simulation while prohibiting the investment product experts to answer while the counselor did not ask them (rule FR8); and another giving an obligation to the certificate of deposit chatbot to do the simulation (rules FR6 and FR7). Since we modeled an obligation to always have precedence over prohibitions in finch, both chatbots replied to the user. Nevertheless, the results indicated that the initial set of norms we modeled were sufficient to enable inter-message coherence on 97% of the dialogues.

**Conclusions and Future Work**
In this paper we presented DSL-CR, a Domain Specific Language for Conversation Rules for multiparty chat groups and how it can be used to enforce in real time a set of conversation rules. As a future work we are exploring the DSL-CR with self-regulated participants in order to compare the conversation rules which are needed. We are also developing an interactive tool for creating the conversation rules in a simpler and more intuitive way using visual programming and getting ready to run user experience experiments with both chatbot developers and linguists. Finally, once with a large multi-party conversation corpus, this approach can provide a modeling foundation for using HMM or POMDP.

**Acknowledgments**
For the finch study, we would like to thank all the users of the system and the finch team: Heloisa Candeloro, Marisa Vasconcelos, Mauro Pichiliani and Renan Souza.

**References**
Arcos, J. L.; Esteva, M.; Noriega, P.; Rodríguez-Aguilar, J.; and Sierra, C. 2005. Engineering open environments with


Holmer, T. 2008. Discourse structure analysis of chat communication. Language@Internet 5(10).


