An Implementation of Norm-based Agent Negotiation

Pieter Dijkstra
Centre for Law & ICT
University of Groningen
The Netherlands
+31503635898
p.dijkstra@rug.nl

Henry Prakken
Centre for Law & ICT
University of Groningen
Department of Information and Computing Sciences
Utrecht University
The Netherlands
henry@cs.uu.nl

Kees de Vey Mestdagh
Centre for Law & ICT
University of Groningen
The Netherlands
+31503635790
sesam@rechten.rug.nl

ABSTRACT

In this paper, we develop our previous outline of a multi-agent architecture for regulated information exchange in crime investigations. Interactions about information exchange between agents (representing police officers) are further analysed as negotiation dialogues with embedded persuasion dialogues. An architecture is proposed consisting of two agents, a requesting agent and a responding agent, using a communication language and protocol with which they can interact in order to promote optimal information exchange while respecting the law. Furthermore, the agents' negotiation policies are defined and implemented and an implementation of the agent execution cycle is proposed, which will ultimately enable us to field test our model in order to supply a proof of concept.

1. INTRODUCTION

In Dijkstra et al [2] we presented an outline of a multi-agent system architecture for regulated information exchange. In this paper we extend our proposal for modelling interactions between agents (representing police officers). The agent architecture consists of two agents, a requesting agent and a responding agent. We recapitulate the definition of the communication language and protocol with which these agents can interact to promote optimal information exchange while respecting the law. We model their interactions as negotiation dialogues about whether information will be exchanged. A negotiation dialogue may shift to an embedded persuasion dialogue about whether such exchange is allowed and at the same time does not violate the responding agent's interests.

In addition to this we develop our analysis of the interactions as negotiation dialogues by proposing an implementation of the negotiation policies. The negotiation policies specify the agent's behaviour in a negotiation dialogue. When an agent receives an offer it deliberates whether it is obliged, forbidden or whether it is a violation of its own interests to perform the action specified in the offer. If the agent concludes that it is obliged to perform the action specified in the offer, it accepts the offer. If the agent concludes it is forbidden to perform the action, it rejects the offer. If the agent concludes it is permitted to perform the action but there is a violation of the agent's own interests then it tries to find conditions under which there will be no violation of its own interests. If it has succeeded in finding such conditions it makes a counteroffer with the extra conditions, else it rejects the offer.

Finally, we define and implement an execution cycle for the individual agents, which will ultimately enable us to field-test our model in order to supply a proof of concept.

In summary this paper adds a specification and implementation of the dialogue policies for negotiation and of the agent execution cycle within the police intelligence domain to our previous work.

This research is part of an ongoing research project ANITA (Administrative Normative Information Transaction Agents), which aims at performing the fundamental research needed to develop a multi-agent system for regulated information exchange in the police intelligence domain [12].

The paper is organised as follows. In Section 2 we discuss the problem of regulated information exchange and how it manifests itself in actual instances of information exchange between Dutch police forces. In Section 3 we list the main requirements for a multi-agent architecture in this domain. In Section 4 we recapitulate the dialogue system for negotiation with embedded persuasion. In Section 5 we propose an architecture, including the negotiation policies and the agent execution cycle, that meets the requirements. Finally in Section 6, we illustrate our architecture with an example.

2. THE PROBLEM OF REGULATED INFORMATION EXCHANGE

In this section we summarize the problem of regulated information exchange we presented in Dijkstra et al [2]. Information exchange is regulated by legal norms and by the policies of the exchanging institutions. This regulation of information exchange serves several goals. On the one hand, the

privacy of the persons who are the subjects of the information must be protected. On the other hand, the legitimate interests of the information exchanging institutions must be served. Typically, institutions must balance the goal to exchange as much information as possible with the obligation to stay within the limits of the law. In most cases there is a central supervising institution (e.g. the ministry of Justice) that has to give account of the effectiveness and lawfulness of the actual information exchanges and that is therefore interested in both optimal and legitimate information exchange. Besides the central institution there are the regionally or functionally distributed local institutions with their own interests and objectives. The central institution often takes the interests at the local level into account by formulating legal norms and central policies that give room for fine tuning in local policies and individual decisions by granting discretionary authority to the local institutions.

Ideally, dialogues between officials of different local institutions guarantee that an optimal and legitimate balance is found in the exchange of information. In consequence, when exchanging information with each other, police officers often have to interact to make sure they conform to the regulations and at the same time serve local interests. However, in practice this ideal is not always realised. Police departments are very reluctant to share crime investigation information with other departments, even if the sharing of information is allowed. One of the main research goals of the ANITA project is to investigate whether such problems can be tackled by providing automated support for information-exchanging police officers.

As a possible solution to the above-mentioned problem of regulated information exchange we investigate the use of a multi-agent architecture. The idea is that the overall goals of an organisation (optimal and lawful information exchange) and local goals (for instance the protection of informants and the continuity of the investigations) are promoted by the design of the individual agents and the ways they interact. We illustrate this with two examples of such interactions between police officers from Dutch police practice. The Dutch police organisation is divided into separate departments, which each operate in its own region. In order to solve criminal cases, departments often need information held by other departments. Information exchange between police departments is governed by national and international privacy regulations and these regulations are supplemented by local rules and policies of the departments.

A typical interaction is about the exchange of information acquired from informants (about 80% of police information on heavy crime in the departments we examined is obtained from informants). Police departments are very cautious about the exchange of this kind of information, since crime suspects who are confronted with information obtained from informants may find out who supplied the information, and this may endanger the safety of the informant and also the continuity of the investigation performed by the department that supplied the information. Therefore, in most cases the department that 'runs' the informant will not be willing to supply the information unless the receiving department offers certain guarantees.

2.1 Examples

Example 1 shows how information is exchanged through a process of persuasion within negotiation. In our multi-agent architecture the acceptance of a claim by an agent depends on its internal persuasion policy. Example 2 elaborates on the internal reasoning of an agent before it agrees to give the requested information on a guarantee stated in the condition.

Example 1: national importance

Agent *A* working in police region *a* requests information about trading in explosive materials from agent *B* working in region *b*. The responding agent finds a unit of information matching the query but initially rejects the request.

A: Give me all information about trading in explosive materials.

B: I will not give you this information.

A: Why don't you give me that information?

B: Because it is forbidden to do so.

A: Why is it forbidden to share that information?

B: Because sharing could endanger the continuity of an investigation.

A: You may be right in general but in this case it not forbidden to share the information because this is a matter of national importance.

B: Ok, I admit that in this case it is not forbidden to share the information, so I retract that it is forbidden to share the information. I will give you this information on the following condition: the given information may not be exchanged with other police officers.

A: I agree with this condition.

Example 2: statistical purposes

Agent *A* working in police region *a* requests information about Soprano from agent *B* working in region *b*. The responding agent finds a unit of information and reasons it only can share the information when it is used for statistical purposes.

A: Tell me all you know about Soprano.

Now the responding agent internally deliberates whether it is obliged, forbidden or whether there is violation of his own interests to give information about Soprano. The responding agent concludes that he can only give information about Soprano if it used for statistical purposes, otherwise there will be a violation of his own interests. Therefore he makes a counteroffer stating the condition about statistical purposes.

B: I will give this information about Soprano under the following condition: the given information may only be used for statistical purposes.

A: I agree with this condition.

3. REQUIREMENTS FOR THE MULTI-AGENT ARCHITECTURE

In this section we describe the requirements that should be met by the multi-agent architecture in order to be a tenable model of regulated information exchange in police practice.

3.1 Knowledge

In order to regulate distributed information exchange, agents must have knowledge of the relevant regulations and the local interpretations of those regulations, their goals and the likely consequences of their actions.

- *Relevant regulations and the local interpretations of those regulations*
  Typical examples of the relevant regulations are "Information must be exchanged if this is necessary for the execution of the requesting party's appointed task" [any agent (cf. police officer) authorized to access a police database is obliged to supply another agent with all the information from its database as far as the other agent needs the information to fulfil his duty] and "It is allowed to refuse to exchange information if such refusal is necessary for the execution of one's own appointed task" [an agent authorized to access a crimes database is permitted to refuse another agent information from the database or supply the information under conditions if necessary for an appropriate implementation of his duties] (sections 14 and 13a3 of the Dutch Police Registers Act).
- *Goals*
  Typical examples of goals are the protection of informants and the contribution to the overall goals of the organisation.
- *The likely consequences of their actions*
  A typical example of the likely consequences of actions is the satisfaction of the overall goal of all agents to share information, however the same action can have the consequence that it conflicts with the goal to protect informants.

3.2 Reasoning

As illustrated by the examples in the previous section, the interaction between agents often involves argumentation. Therefore, the agents should be capable of generating and evaluating arguments for and against certain claims and they must be able to revise their beliefs as a results of the dialogues. Finally, in order to generate conditional offers, the agents should be able to do some form of hypothetical reasoning.

3.3 Communication

Walton and Krabbe [13] distinguish six classes of dialogue, of which we use three: information seeking, negotiation and persuasion. Information seeking dialogues are used for transferring information, negotiation dialogues are used for resolving a conflict of interests and persuasion dialogues are used for resolving differences of opinion. The responding agent's goals sometimes lead him to state conditions under which he is willing to give information. Therefore, the agents must be able to negotiate with each other. Also, the responding agent may be mistaken in believing that he must or should not give the requested information. Therefore, the agents must be able to engage in persuasion dialogues. To enable such interactions, a suitable dialogue protocol must be implemented. Also, the agents must be given policies for their behaviour in the dialogues. These policies should be designed to further the agent's goals. Since these goals also include those of the overall institution, the agents' policies should induce a fair degree of cooperativeness.

As we have argued in [2] we contend that most interactions in our domain start as a negotiation, viz. as an offer to give information. This seems more like negotiation than like information-seeking. Such a dialogue may shift to embedded persuasion if the requesting agent tries to persuade the responding agent that he is wrong about a rejection, e.g. that granting would endanger the continuity of the investigations. After the persuasion terminates the interrupted negotiation resumes. If that terminates successfully, a (trivial) information-seeking dialogue starts; its termination is also the termination of the overall interaction.

4. FORMALISATION

In this section we summarize the dialogue system for negotiation with embedded persuasion we described in [2].

4.1 Dialogical interaction

Dialogue systems have a *communication language C* with associated *protocol P* and a *topic language T* with associated *logic L* (in our case nonmonotonic). The communication language consists of *speech acts l(content)* where *l* is a locution and *content* is an element or subset of *T* or an argument in *L*. Our language and protocol essentially is that of [10], which combines a negotiation protocol of [14] with a persuasion language and protocol of [8].

4.1.1 Communication language

In [2] we presented the sublanguages for negotiation and persuasion and defined their combination. In this section we summarize the two combined communication languages.

Following [8], our communication language has a reply structure: each speech act replies to one preceding speech act in the dialogue. Moreover, a reply can be of two kinds, being either an attacking or a surrendering reply. **Table 1** specifies how to attack or surrender to another speech act.

Table 1 contains the two combined communication languages. The idea is that the *why-reject p* speech act triggers a persuasion dialogue. The only possible reply to this move is *claim q*, where *q* is a ground for the rejection and this claim starts a persuasion dialogue.

Table 1. A combined communication language (C_{np})

speech acts	**attacks**	**surrenders**
offer *p*	offer *q* ($q \neq p$) reject *p*	accept *p* withdraw
reject *p*	offer *q* ($q \neq p$) why-reject *p*	
accept *p*		
withdraw		
why-reject *p*	claim *q*	
claim p	why *p*	concede *p*
why p	*p* since *Q*	retract *p*
p since *Q*	why *q* ($q \in Q$) *p*' since *Q*'	concede *q* ($q \in Q$) concede *p*
concede *p*		
retract *p*		

4.1.2 Communication protocol

The negotiation and persuasion protocols have the following rules in common. A *move* is a speech act, made according to the reply structure of *C*. If it is not the first move, it is a reply to a unique preceding move in the dialogue made by the other dialogue participant. A dialogue is terminated if a player is to move but has no legal moves.

In a negotiation the requesting agent begins with an offer, and then the agents take turns after each move, replying to the last move of the other party. Negotiations can be of arbitrary length and terminate after an accept or withdraw move but they are not guaranteed to terminate. The *persuasion protocol* is not the primary focus of this paper, for a sketch of the main rules the reader is referred to [2].

As for the *combined protocol*, the main idea is that if a negotiation dialogue shifts to a persuasion dialogue, their relation is one of embedding (cf. [1]): the embedded persuasion dialogue is undertaken until its termination, after which the embedding negotiation dialogue is resumed. So whenever a persuasion move is allowed by the protocol, no negotiation move is allowed. For a specification of the combined protocol the reader is referred to [10].

5. AGENT ARCHITECTURE

In this section we describe the components of the internal negotiation agent architecture (see Figure 1) and we specify the negotiation policy. The agents are part of a bi-agent architecture in which one agent has the role of requesting agent, i.e. an agent that requests information and the other agent has the role of a responding agent. The responding agent has access to information available in the local database and only supplies information if it is obliged or else if it is permitted and does not violate its own interests. Typical users will be police officers who are authorized to investigate crime. After being authorized by the system the user can enter a query and an agent in the role of requesting agent will, on behalf of the user, request the responding agent for information. The responding agent searches in its database for information-units matching the query. For the matching information-unit it deliberates whether it is obliged, forbidden or whether there is a violation of its own interests to supply the information-unit. If the responding agent concludes that it is obliged to give the information-unit, it sends the information-unit to the requesting agent. If the responding agent concludes that it is forbidden to give the information-unit it sends a reject to the requesting agent. If it is permitted but there is a violation of its own interests the responding agent tries to find a condition under which there will be no violation of its own interests. If it has succeeded in finding such a condition it offers to give the information-unit under the found condition, otherwise it sends a reject.

5.1 Description of the Components

The *user communication module* provides for the external interaction with the user of the agent. The external input is a query (e.g. "Soprano") from the user and the external output is the result of the interaction about the (requested) information-unit. Internally the user communication module sends the query to the execution cycle and receives the end results of the interactions from the execution cycle. The possibility of more extensive user interaction will be investigated in future research. For example when the agents are stuck in a dialogue they could query the users to decide what to do next. Another example is when an agent receives a previously unknown condition or argument and queries the user if it can add this new knowledge to its knowledge base.

The *database communication module* provides for the communication with the external database. The database contains the information-units and information-unit indices. An information-unit is a grouped piece of information that has an associated information-unit index which is also used in the knowledge base of the argumentation system.

The *agent communication module* provides for the communication with another agent and parses messages for processing in the agent execution cycle

In the *execution cycle module* messages are processed and the selection of the appropriate dialogue moves trigger the necessary modules. In this paper we only discuss the negotiation policy module but a persuasion dialogue module can (and will) be developed similarly. The execution cycle also obtains the information-unit index matching a query and the information-unit corresponding to the information-unit index from the external database. In our implementation the execution cycle is developed in Java.

The *negotiation policy module* is called from the execution cycle when the agent has to deliberate whether or not it will give information. The negotiation policy contains the domain-dependent negotiation rules. In our implementation the negotiation policy is developed in Java as a set of forward chaining if-then rules

The *argumentation system module* consists of an argumentation engine and a knowledge base. The knowledge base contains the knowledge of the relevant regulations for information exchange, the local interpretations of those regulations, the goals of the agents and how actions can violate or promote those goals. The knowledge base also contains information-unit indices, which are references to the information-units in the external database. Each agent has its own knowledge base and database and both are not accessible for other agents. In our implementation we use the ASPIC argumentation system which is developed within the ASPIC project (IST-FP6-002307) and which is based on the algorithm described in [11]. ASPIC is an inference engine for an argument-based nonmonotonic logic. Knowledge is represented in a rule-based language and arguments are constructed by chaining rules into trees. Arguments can be defeated in two ways: they can be rebutted with arguments for contradictory conclusions and they can be undercut with arguments holding that there is an exception to a rule. Conflicts between rules are decided with rule priorities. The tool computes the dialectical status of arguments according to grounded semantics (cf. [4]). We are especially interested if an argument is *justified*, i.e. whether it is a member of the (unique) grounded extension. We call an argument *overruled* if it is defeated by a justified argument an not in the grounded extension and we call an argument *defensible* otherwise.

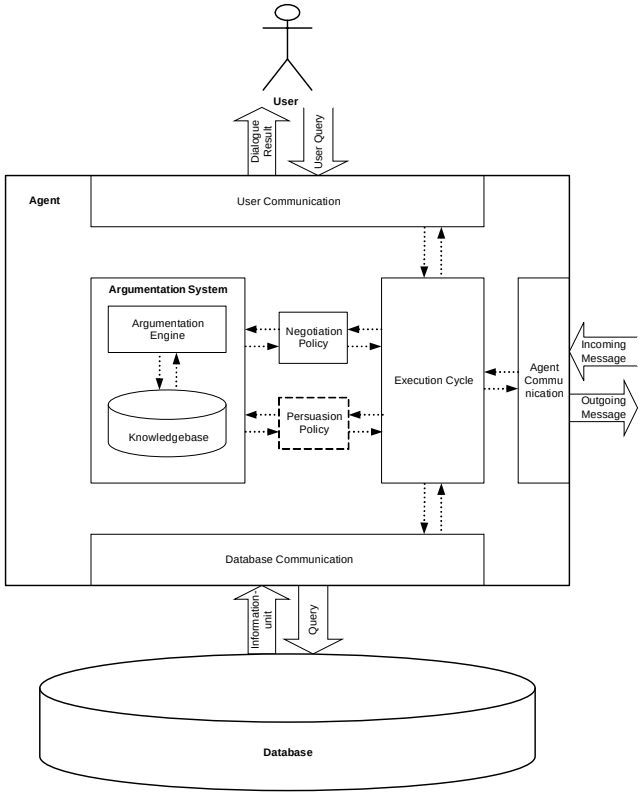

Figure 1: internal agent architecture

5.2 Negotiation Policy

We next specify how an agent will respond to negotiation utterances that it receives. The communication language consists of speech acts *l(content)* where *l* is a locution (e.g. offer) and the *content* is divided into two parts $q \wedge conditions$ where *conditions* is a (possible empty) conjunction of conditions and *q* has the following elements: *sender* and *receiver* denote the sender and the receiver respectively and *action* denotes the requested action. Since the logic is not the main concern of this paper, we assume that act descriptions are expressed as conjunctions of first-order literals without free variables. As we explained in [2], conditional offers are more naturally represented with a conjunction than with conditional operators, since the content of a conjunctive agreement allows the agents to infer what they have committed to irrespective of whether the other agent keeps his part of the agreement The conjunctions can be used to state the conditions for acceptance, for example if the requesting agent says 'offer: you tell me all you know about Soprano' then the responding agent could state conditions in a counteroffer 'offer: I give you all information about Soprano $\wedge$ you may only use the information for statistical purposes'.

When the negotiation policy module receives a message it queries the argumentation system whether it can create a justified or defensible argument for the given conclusion. The argumentation system tries to find a defensible or justified conclusion given the union of the knowledge base of the argumentation system, the action and the conjunction of conditions. We use $[A] \vdash B$ to denote that a reasoner is called to infer that the set of rules *A*, implies conclusion *B*. For pragmatic reasons no full logic for the modalities will be implemented but only the axioms that are necessary for our purposes will be represented. We will respect the meaning of deontic operators by using reification. The relations between the deontic modalities will be defined as in standard deontic logic.

Our negotiation policies consider two issues: the normative issue of whether accepting an offer is obligatory or forbidden, and the teleological issue whether accepting an offer violates the agent's own interests. Of course these policies can be different for the requesting and the responding agent. Also, different agents can have different policies.

The first step of the negotiation policy for an offer is to check whether it is possible to create a justified or defensible argument (status-type) for the conclusion that the requested content is obliged. If that is possible, the negotiation policy returns the answer to accept the offer.

IF offer($q \wedge conditions$)

$\wedge [KB_{argumentation\text{-}engine} \cup conditions]$

$\vdash_{status\text{-}type}$ obliged (q)

THEN accept ($q \wedge conditions$)

The second step of the negotiation policy for an offer is to check whether it is possible to create a justified or defensible argument for the conclusion that the requested content is forbidden. If that is possible, the negotiation policy returns the answer to reject the offer. Since we use only the obliged operator, forbidden(q) is rewritten as obliged(not(q)) in our negotiation policies.

ELSE IF offer($q \wedge conditions$)

$\wedge [KB_{argumentation\text{-}engine} \cup conditions]$

$\vdash_{status\text{-}type}$ obliged (not(q))

THEN reject ($q \wedge conditions$)

Since the KB language does not contain double negation, the occurrence of 'not(q)' in this policy should, when q is a negated formula, strictly speaking be replaced with q. For simplicity we leave this complication here and below implicit.

The third step of the negotiation policy for an offer is to check whether it is possible to create a justified or defensible argument for the conclusion that the requested content is a violation of the agent's (denoted by 'a') own interests. If that is impossible, the negotiation policy returns the answer to accept the offer.

ELSE IF offer($q \wedge conditions$)

$\wedge [KB_{argumentation\text{-}engine} \cup \text{holds}(q) \cup conditions]$

$\nvdash_{status\text{-}type}$ violation-of-own-interests (a)

THEN accept ($q \wedge conditions$)

In the fourth step of the negotiation policy for an offer, the argumentation system tries find a minimal set of conditions for the conclusion such that the responding agent's own interests are not

violated and the conclusion that the requested content is not forbidden. If that is possible, the negotiation policy returns the answer to make a counteroffer with the extra conditions. If that is not possible the negotiation policy returns the answer to reject the offer.

ELSE IF offer($q \wedge$ *conditions*)

$\wedge$ [$KB_{argumentation\text{-}engine} \cup$ holds(q) $\cup$ (subset-minimal set c)]

$\nvdash_{status\text{-}type}$ violation-of-own-interests (a)

$\wedge$ [$KB_{argumentation\text{-}engine} \cup$ (subset-minimal set c)]

$\nvdash_{status\text{-}type}$ obliged (not(q))

THEN offer ($q \wedge c$)

ELSE reject ($q \wedge$ *conditions*)

The fourth step of the negotiation policy uses hypothetical reasoning to find conditions under which the offer can be accepted. The used argumentation engine is not capable yet of hypothetical reasoning therefore in the current implementation this is done manually.

In the negotiation policy for a reject, the policy returns a why-reject move which starts an embedded persuasion dialogue. The specification and implementation of embedded persuasion dialogues will be the subject of future research.

5.3 Agent execution cycle

In this section we describe the algorithm for the execution cycle of a negotiation agent. The agent execution cycle processes messages and triggers other modules during the selection of the appropriate dialogue moves. First the speech act, locution and content are parsed from the incoming message, then depending on the locution (offer, accept, withdraw or reject) the next steps are taken.

If the locution of the incoming message is an *offer* and it is the initial message (indicated by message id 1), it is considered to be a query. The query is then parsed and the corresponding information-unit index (`'INFO_UNIT_ID')` is retrieved from the database (for simplicity, we assume a query can match only one `INFO_UNIT_ID`. If there is no matching `INFO_UNIT_ID` for the query, a *withdraw* message is sent to the agent communication module, else the query is instantiated by the corresponding `'INFO_UNIT_ID'` by using the `'instantiateQuery'` function. Next, the instantiated speech act is submitted to the negotiation policy to reason about whether it is obliged, forbidden or whether there is a violation of the agent's interests to supply the `'INFO_UNIT_ID'`. The locution is then rewritten by the result of the `'negotiationPolicy'` function (reject, offer or accept) and the message is sent to the agent communication module.

If the locution of the incoming message is an *offer* and it is not the initial message, the locution is rewritten by the result of the `'negotiationPolicy'` function and sent to the agent communication module.

If the locution of the incoming message is an *accept*, a (trivial) information-seeking dialogue is started by changing the locution to *send* and the corresponding information-unit (`'INFO_UNIT'`) is added to the message.

If the locution of the incoming message is a *withdraw*, the agent stops the dialogue. A withdraw message is only sent when the other agent cannot find an `'INFO_UNIT_ID'` that matches the query.

If the locution of the incoming message is a *reject,* an embedded persuasion dialogue is started, this is not yet implemented and will be subject of future research.

The execution cycle can be represented in Java pseudo-code as follows:

```
for-each incoming-message(MESSAGE){
SPEECH_ACT = MESSAGE.SPEECH_ACT
LOCUTION = SPEECH_ACT.LOCUTION
CONTENT = LOCUTION.CONTENT

if LOCUTION = offer and MESSAGE.ID = 1
then
        QUERY = getQuery(CONTENT)
        INFO_UNIT_ID = getInfoUnitID(QUERY)
        if INFO_UNIT_ID = 0 //no matches
        then
                LOCUTION = withdraw
                MESSAGE.INFO_UNIT ="no matches"
                sendToAgentComm(MESSAGE)
                exit
        else
                SPEECH_ACT =
                instantiateQuery(INFO_UNIT_ID,
                SPEECH_ACT)
                LOCUTION =
                negotiationPolicy(ATTITUDE,
                SPEECH_ACT)
                sendToAgentComm(MESSAGE)

else if LOCUTION = offer and MESSAGE.ID > 1
then
        LOCUTION = negotiationPolicy(ATTITUDE,
        SPEECH_ACT)
        sendToAgentComm(MESSAGE)

else if LOCUTION = accept
then
        LOCUTION = send
        MESSAGE.INFO_UNIT =
        getInfoUnit(INFO_UNIT_ID)
        sendToAgentComm(MESSAGE)

else if LOCUTION = withdraw
then
        exit

else if LOCUTION = reject
```

```
then
        //embedded persuasion…
}
```

6. ILLUSTRATION OF THE AGENT ARCHITECTURE

In this section we will illustrate the agent architecture using example 2. This shows how a responding agent uses its negotiation policy to conclude to give an information-unit under the condition that the information may only be used for statistical purposes.

6.1 Knowledge base

Knowledge is represented in the prolog-like syntax of the ASPIC tool. Function symbols, predicate symbols and constants start with a lower case letter, variables start with a capital letter or underscore. For the implementation in the ASPIC reasoner, the deontic modalities are represented by reification. In order to represent example 2, the following rules, axioms and facts are added to the requesting agent's knowledge base:

```
%RULES
obliged(give(B, A, INFO_UNIT_ID))
<-
holds(public_information(INFO_UNIT_ID)),
offer(A, B, give(B, A, INFO_UNIT_ID)).
```

If information is publicly accessible for other agents then it is obliged to give it to them.

```
goal(o, give(B, A, INFO_UNIT_ID))
<-
holds(crime_information(INFO_UNIT_ID)),
holds(authorized_crime_investigator(A)),
holds(authorized_crime_investigator(B)).
```

A goal of the entire organisation (denoted by constant o) is to exchange as much information as possible with the other authorized crime-investigators of the organisation. This rule is the interpretation of section 14 of the Dutch Police Registers Act: "Information must be exchanged if this is necessary for the execution of the requesting party's appointed task"

```
goal(A, not(identity_revealed(INFO_UNIT_ID)))
<-
holds(informant_of(INFO_UNIT_ID, A)).
```

A possible goal of an individual agent is not to reveal the identity of informants. Since revealing the identity of the informant can endanger the informant and the continuity of the investigation of the agent.

```
holds(identity_revealed(INFO_UNIT_ID, INFORMANT)
<-
holds(informant_of(INFORMANT,INFO_UNIT_ID, A)),
holds(give(B, A, INFO_UNIT_ID)),
holds(traceable(INFO_UNIT_ID, INFORMANT)).
```

If information of an agent comes from an informant and the information is given to another agent and the information is traceable then the identity of the informant is revealed.. Information is traceable when it can lead to identification of the informant. For example, if the informant was the only witness of a crime, then the criminal knows that the information can only originate from the informant.

```
~holds(identity_revealed(INFO_UNIT_ID, INFORMANT))
<-
holds(informant_of(INFORMANT,INFO_UNIT_ID, A)),
holds(give(B, A, INFO_UNIT_ID)),
holds(traceable(INFO_UNIT_ID, INFORMANT)),
holds(statistical_purposes(INFO_UNIT_ID, A)).
```

If information of an agent comes from an informant and the information is given to another agent and the information is traceable and it is only used for statistical purposes then the identity of the informant is not revealed.

```
violation_of_own_interests(A)
<- holds(P), goal(A, not(P)).
```

If something is a goal of an agent but the opposite holds, then his interests are violated .

```
%AXIOMS
~holds(not(P)) <- holds(P).
~holds(P) <- holds(not(P)).
~obliged(not(P)) <- obliged(P).
~obliged(P) <- obliged(not(P)).
```

These axioms in the knowledge base are introduced to preserve the meaning of the deontic modalities according to standard deontic logic.

```
%FACTS
holds(crime_information("3")).
holds(authorized_crime_investigator(a)).
holds(authorized_crime_investigator(b)).
holds(informant_of(informant22, b)).
holds(traceable(informant22, "3")).
```

6.2 Dialogue from example 2

We now illustrate how the negotiation agent internally processes an incoming query in the dialogue of example 2 (Section 2).

The responding agent b receives the initial offer with a query from the requesting agent a to give information about Soprano:

```
1, offer(a, b, give(b, a, "Soprano"), ),
```

The responding agent b tries to find a matching information-unit index for the query (for simplicity we assume only one

information-unit index matches a query). An information-unit index is an identification referring to the information-unit available in the external database. The responding agent finds a matching information-unit index ("3") and instantiates the query of the speech act:

```
offer(b, a, give(b, a, "3"), )
```

The speech act and the agent attitude is given to the negotiation policy function. The agent attitude contains the domain specific choices for the negotiation steps whether a justified or a defensible argument is needed. The negotiation policy receives the following information:

```
attitude, offer(b, a, give(b, a, "3"), )
```

The negotiation policy queries the argumentation system using the first step of the policy for the conclusion whether the argumentation system can find a justified argument for the conclusion that it is obliged to give information:

```
justified, obliged, offer(b, a, give(b, a, "3"), )
```

Since there is no matching rule, the negotiation policy (‘negotiationPolicy’) continues to the second step in the negotiation policy. The second step is whether the argumentation system can find a justified argument for the conclusion that it is forbidden to give information:

```
justified, forbidden, offer(b, a, give(b, a, "3"),
)
```

Also in the second step, the argumentation system cannot find a justified argument for the conclusion that is forbidden to give information, so the third step in the negotiation policy is taken. The third step in the negotiation policy is to check if it there is no violation of the agent's own interests if it provides the information requested:

```
justified, violation_of_own_interests, offer(b, a,
give(b, a, "3"), )
```

There is a violation of interests:

```
holds(identity_revealed("3", informant22)).
<-
holds(informant_of(informant22,"3",b)),
holds(traceable("3", informant22)),
holds(give(b, a,"3")).
```

and:

```
goal(b, not(identity_revealed("3", informant22)))
<-
holds(informant_of(informant22, "3", b)).
```

therefore:

```
violation_of_own_interests(b)
<-
holds(identity_revealed("3", informant22)),
goal(a, not(identity_revealed("3", informant22))).
```

Now the agent continues with the fourth step in the negotiation policy and tries if it can find an extra condition under which there is no violation of its own interests:

```
justified, violation_of_own_interets, offer(b, a,
give(b, a, "3"), )
```

By hypothetical reasoning an extra condition is found in the following matched rule:

```
~holds(identity_revealed("3", informant22))
<-
holds(informant_of(informant, "3", b)),
holds(traceable("3", informant22)),
holds(give(b, a, "3")),
holds(statistical_purposes("3", a)).
```

The negotiation policy for an offer is completed and returns the result back to the execution cycle:

```
offer(b, a, give(b, a, "3"),
statistical_purposes("3", a))
```

The message is sent to the agent communication module which sends the message to the requesting agent.

```
2, offer(b, a, give(b, a, "3"),
statistical_purposes("3", a)),
```

The requesting agent accepts the offer and sends the following message back to the responding agent:

```
3, accept(a, b, give(b, a, "3"),
statistical_purposes("3", a)),
```

Finally the responding agent sends the requested information through an information exchange protocol.

7. CONCLUSION

In this paper we have extended our proposal (in [2]) for a multi-agent architecture for regulated information exchange. We have developed our analysis of the interactions as negotiation dialogues and proposed an implementation of the negotiation policies. Furthermore, we have proposed an implementation of the agent execution cycle, which will ultimately enable us to field-test our model in order to supply a proof of concept. The architecture has been illustrated with an example of information exchange between police forces in the context of crime investigation. The architecture combines and adapts several elements from the literature: a defeasible-argumentation mechanism for the agents’ internal reasoning behaviour, a communication language, a protocol for negotiation with embedded persuasion about reasons for rejections of offers and dialogue policies for negotiation. Also, we have proposed and developed a view on the nature of dialogues in the context of regulated information exchange, viz. as negotiation with embedded persuasion.

As for the state of implementation and validation, the knowledge bases of the agents do not yet fully represent the relevant regulations and policies for the police domain. Furthermore, the persuasion policies should be implemented for a proof of concept system, which is necessary for validation of our proposal in police practice.

As for related research, Karacapilidis and Moraitis [5] proposed a general framework for automated dialogues between agents and also paid attention to keep their framework as operational as possible. Furthermore, their framework also enables other types of dialogue, such as persuasion and deliberation dialogues. However, their negotiation policies do not consider the normative issue and the teleological issue which we consider important for modelling regulated information exchange between agents. We use their style '$[A] \vdash B$' to denote that a reasoner is called to infer that the set of rules *A* implies conclusion *B*.

Parsons, Wooldridge, and Amgoud [7] first studied formal dialogue policies for argumentation and called them "agent attitudes". However, they only defined policies for persuasion, while we are interested in policies for negotiation.

Doutre, McBurney and Wooldridge [3] propose a model for regulated information exchange in the medical domain. They model it as information seeking dialogues with embedded persuasion about whether providing the requested information is permitted, while we model the regulated information exchange as negotiation dialogues with embedded persuasion.

As for future research, the embedded persuasion policies should be specified and implemented. Also, we want to investigate other combination patterns of dialogue types, including information-seeking. As for the negotiation part of the dialogue system, we aim to investigate whether besides arguing about rejections, other ways to argue in negotiation, such as those studied by [6], occur in our application domain and should therefore be modelled in our architecture. Furthermore, the possibility of more extensive user interaction should be investigated. For example when the agents are stuck in a dialogue they could query the users to decide what to do next or when an agent receives a previously unknown condition or argument and queries the user if it can add this new knowledge to its knowledge base. Finally, a full implementation should be developed to field-test our model to supply a proof of concept, which is a prerequisite for a professional implementation that will support and improve police practice.

8. ACKNOWLEDGMENTS

This research was supported by the Netherlands Organisation for Scientific Research (NWO) under project number 634.000.017 and is part of the ongoing project ANITA (project manager Kees de Vey Mestdagh). Henry Prakken was also partially supported by the EU under IST-FP6-002307 (ASPIC). Our thanks goes to Matthew South for his help with the ASPIC tool.

9. REFERENCES

[1] McBurney, P. and Parsons, S. (2002). Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information* 11: 315-334.

[2] Dijkstra, P., Bex, F.J., Prakken, H. and De Vey Mestdagh, C.N.J. (2006). Towards a multi-agent system for regulated information exchange in crime investigations. *Artificial Intelligence and Law* 13: 133-151

[3] Doutre, S., McBurney, P. and Wooldridge, M. (2005). Law-Governed Linda as a semantics for agent interaction protocols (research abstract). In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005)*, Utrecht, The Netherlands, July 2005: 1257-1258.

[4] Dung, P.M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n–person games. *Artificial Intelligence*, 77: 321–357.

[5] Karacapilidis, N. and Moraitis, P. (2002). Engineering issues in inter-agent dialogues. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI 2002)*, Lyon, France, July 2002: 58-62

[6] Parsons, S., Sierra, C. and Jennings, N. (1998). Agents that reason and negotiate by arguing. *Journal of Logic and Computation* 8: 261-292.

[7] Parsons, S., Wooldridge, M. and Amgoud, L. (2002). An analysis of formal inter-agent dialogues. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-02)*: 394-401

[8] Prakken, H. (2005). Coherence and flexibility in dialogue games for argumentation. *Journal of Logic and Computation* 15: 1009-1040.

[9] Prakken, H. and Vreeswijk, G. (2002). Logical systems for defeasible argumentation. In D. Gabbay and F. Guenthner (eds.), *Handbook of Philosophical Logic*, second edition, Vol 4: 219-318. Kluwer Academic Publishers, Dordrecht etc.

[10] Van Veenen, J. and Prakken, H. (2005). A protocol for arguing about rejections in negotiation. In S. Parsons, N. Maudet, P. Moraitis & I. Rahwan (eds.), *Argumentation in Multi-Agent Systems*. Springer Lecture Notes in AI 4049: 138-153.

[11] Vreeswijk, G. (2006). An algorithm to compute minimally grounded and admissible defence sets in argument systems. In *Proceedings of the First International. Conference on Computational Models of Argument*, Amsterdam etc., IOS Press: 109-129.

[12] De Vey Mestdagh, C.N.J. (2003). Administrative Normative Information Transaction Agents (ANITA): Legitimacy and Information Technology, the best of two worlds. In *Access to knowledge and its enhancements, Proceedings ToKeN2000 symposium*, Delft University of Technology, February 21, 2003.

[13] Walton, D. and Krabbe, E. (1995). *Commitment in dialogue: Basic Concepts of Interpersonal Reasoning*, State University of New York Press, NY

[14] Wooldridge, M. and Parsons, S. (2000). Languages for negotiation. *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI-2000)*: 393-400. Amsterdam: IOS Press.