DipGame: a challenging negotiation testbed

Angela Fabregues and Carles Sierra

IIIA-CSIC: Institut d'Investigació en Intel·ligència Artificial del CSIC, Campus UAB, E-08193 Bellaterra, Barcelona, Spain.

Abstract

There is a chronic lack of shared application domains to test advanced research models and agent negotiation architectures in Multiagent Systems. In this paper we introduce a friendly testbed for that purpose. The testbed is based on The Diplomacy Game where negotiation and the relationships between players play an essential role. The testbed profits from the existence of a large community of human players that know the game and can easily provide data for experiments. We explain the infrastructure in the paper and make it freely available to the AI community.

Keywords: Multiagent Systems, Negotiation, Testbed, Diplomacy Game

1. Introduction

Current research trends in multiagent systems (MAS) include models of trust [1], reputation [2, 3], and argumentation [4, 5] to improve negotiating strategies [6, 7, 8]. Research progress in the development of these theoretical models has made them very sophisticated (based on cognitive, information or game theoretical grounds) and has enabled software agents to interact with and help humans in a more efficient and believable way. Agents are being endowed with techniques to decide how and when to interact, argue and negotiate, as well as whom to trust and why. Unfortunately, the practical impact of the models has been lower than expected. This is partly due to the fact that researchers lack rich-enough use cases to compare the models. The use cases described in the MAS literature tend to be rather artificial and usually biased to a particular negotiation model. Rich enough testbeds are urgently needed to simulate the latest sophisticated models without having to oversimplify them. In particular, software agents must be able to interact not just with other software agents but also with humans and in realistic scenarious. The testbed that we present in this paper, DipGame, is rich enough to be useful in negotiation contexts with: (1) a huge space of solutions, such that purely strategic thinking is not feasible; (2) a partially observable environment, where agents cannot observe all actions made by other agents (e.g. offers); (3) the environment changes continuously due to the actions of other agents, agents decide moves autonomously that affect our future action repertoire; and (4) decision making in time-bounded. These properties are common in realistic scenarios and are not particular to any negotiation model.

The field of negotiation is very broad. Why humans co-operate and how they reach agreements has been since long a fascinating area of inquiry. It has been studied from many different perspectives: game theory [9, 10], psychology [11, 12], business [13], neuroeconomics [14], or psychopharmacology [15] just to mention a few. From an artificial intelligence perspective two main approaches have been followed. First, approaches based on game theory, where agents are assumed to follow a *constructivist* sort of rationality [16], are selfish and look for some sort of 'fair' allocation. In this approach models assume perfect rationality and complete information [17], so that if the optimal solution is known to the players then it is adopted. These assumptions are, unfortunately, not very common in practice. To overcome these difficulties, negotiation strategies have been, to a large extend, based on heuristics that try to use the available information about the problem and the opponents to approximate the 'ideal' game theoretical results [18, 19]. Usually, these heuristics look for solutions that are close to the pareto-efficient frontier without revealing all preferences [6]. The heuristics are applied in negotiation frameworks with simple protocols (e.g. the classical alternating protocol [17], or slightly more sophisticated ones as in [20]) and over simple negotiation domains (e.g. the "AMPO vs City domain" in [21]). A recent comparison of these heuristics can be found in [8].

Second, and more recently, an increasing interest on the psychological and cultural aspects of negotiation has been observed in the field. For instance, agent negotiation architectures inspired in human relationship building [22], or analysis of cultural influences in negotiation strategies [23]. The community of social simulation has also studied negotiation models in as much as they help in defining policies (e.g. [24] in an environmental setting) or norms [25]. These human-oriented approaches follow an *ecological* model of rationality [16] based on two basic notions: *everything* in the world is constantly changing and not *all* facts can be known by an agent. This view on rationality is in line with an evolutionary approach: knowledge evolves, individuals evolve and societies evolve. Ecological rationality, observed in humans and human societies, seems also the right one when we face the design of *software agents*. If knowledge is dynamic and evolves along time, this will certainly be also the case for beliefs or intentions of agents that will change constantly due to the interaction with the environment (or with evolving societies of humans). Actually, the very same structure of the environment where agents and services appear and disappear all the time, and where every active component of the environment is not necessarily persistent in its goals or intentions, makes the view of the world imperfect and necessarily evolving along time.

We do believe that after so many years of development of negotiation models whose validation has been made in very simple toy environments the community requires a more realistic scenario where negotiation models can be tested. The problem with artificial scenarios is that the behaviour of humans is not necessarily the same as in real life. It is not the same to decide on artificial money or on real money. That is why areas like auctions have moved into more realistic scenarios like the different TAC competitions, or why areas like neuroeconomics [26] or experimental economics in general have moved into real scenarios. Current research in negotiation proposes models far more complex than those adapted to game theoretical settings [8] and based on an ecological rationality. Current testbeds [27, 28, 29] are rather simple and we believe that the validation of negotiation models that aim at human-agent interactions [4, 5, 22] require more sophisticated testbeds as the one introduced in this paper: DipGame is a testbed that uses Diplomacy as the negotiation environment.

We start this paper briefly introducing the rules of The Diplomacy Game and discussing why we think that this game is ideal for testing the research work on negotiation; sections 2 and 3. Then we introduce the negotiation language and the infrastructure, sections 4 and 5. Then, the methodology for building negotiation agents is presented together with an example of use of the testbed, section 6. And in section 7 we describe how do we involve humans in the experiments. We finally end the paper with a description of the related work, section 8, and a discussion on the future work, section 9.

2. Diplomacy in a nutshell

In Diplomacy, players negotiate with the aim of conquering Europe. The rules of the game are unambiguous and the available information about the game is rather rich as it is a quite old game being enjoyed by several generations of players: the game appeared in 1954. It is situated on Europe at the beginning of the 20^{th} century, before World War I. Each player is in charge of the armed forces, organised in *units*, of a major European power and must decide, in each turn, which movements the various units should execute. There is a maximum of one unit per province. The game ends when someone has an army powerful enough to control half of the special European 'provinces' called supply centres. This is achieved by defeating other players' units, conquering their provinces and controlling an increasing number of supply centres that allow a player to get more units.

One of the most interesting features of Diplomacy is the absence of random movements: there are no cards and no dices. Also, this is not a turn-taking game. That is, all players move their units simultaneously: there is no advantage in a player being the first or last to move. All units are equally strong and consequently, when a unit attacks another, the winner of the battle is decided by taking only into account the number of units helping the fighting units. This feature is what makes Diplomacy so compelling for our purposes: the most relevant skills for a player are the negotiating ability, the intuition (knowing whom to trust) and the persuasive power.

The game splits in several years with two movement turns per year. In every movement turn, the players decide what movements their units should perform. Possible movements are: to move, to hold or to help another unit supporting its hold or move. When every player has decided their movements, those are made public at the same time and the game state is updated following the rules of the game.[1] The rules describe how to resolve conflicts that may emerge because of the concurrent announcement of movements. At the end of the year if you have a unit over a supply centre province, the supply centre becomes yours. It will be yours until a unit of another power conquers it at the end of a subsequent year period. At the end of each year, the number of units of every player is made equal to the number of owned supply centres by either building new units (when the player increased the number of owned supply centres) or removing existing units (when the player

[1] Available at `http://www.wizards.com/default.asp?x=ah/prod/diplomacy`.

decreased the number of owned supply centres). Remember that owning half of the supplier centres of the Continent allows you to win the game. Therefore, the goal of the players is to increase the number of owned supply centres every year quicker than other players do.

All units have always the same strength. When there is an attack, it is resolved taking into account the number of supports that the attacking and defending units got from other units. Players can support the movements of other players' units. In fact, these are the basics of the game: the players must convince other players to be their allies and to help them. This is done by negotiations that take place in parallel among all the players. In those negotiations, players ask others for help and sign deals on future plans of action like attacking together a unit of a player that they agree is a common enemy. From a player's point of view, the most important aspect of the game is the negotiation process: deciding allies, selecting whom to ask for help, arguing with other players to get information about their objectives or to find out what they know, building trust and reputation, deciding when to honour a deal, maintaining relationships, and so on.

3. Diplomacy for negotiation

Diplomacy is a strategically simple game for humans to play in comparison to other classic games like Chess or Go. This is because the true complexity of the game lies in the management of the relationships among players. Relationships are constantly changing and may appear at first sight difficult to analyze. However, humans negotiate constantly in their every day life, and they are used to do it. Diplomacy is thus not perceived by humans as a difficult game to play, but it is really difficult for a computer as it cannot take advantage from massive computations. The search space is huge and the key for success relies on the information obtained from negotiation rounds and on the persuasive capability of the player.

Focusing just on the possible moves that the units can perform on the board, the combinations are very large. There is an average of 30 units on the board during a game. A movement has to be assigned to each one of these units per turn. The movements that a unit can perform depend on the number of direct neighbours and neighbour units.[2] Assuming an adjacency

[2]By *direct neighbour* we refer to units that are in an adjacent region. And *neighbour*

factor of 4 and a neighbourhood factor[3] of 2, we obtain that the number of possible movements per turn is $30 \cdot 15$.[4] Overall, a branching factor of 450. The branching factor of the search is thus so high that even a no-press (without negotiation) game cannot be treated by standard search mechanisms. Neither can we reasonably use game theory to strategically solve the problem. Think that the branching factor for chess is around 35. Besides that, the execution of the movements in Diplomacy depend on the movements done by the other units. The moves that a player performs are not independent from the ones performed by other players. All players perform their moves at the same time, hence a player cannot be sure about the outcome of a move because there can be conflicts between different players' movements. Therefore, it is very difficult to predict the outcome of a movement without information about the opponent's intentions. In fact, the essence of Diplomacy relies on the diplomatic moves that were not taken into account when analyzing the complexity above. Taking every single negotiation step into consideration, the number of possible moves is simply overwhelming [30].

From the point of view of AI research, Diplomacy is a MAS environment where competitive self interested agents need to cooperate to improve the outcome. This is done by the signature of agreements where agents involved commit to do a plan of action. Agreements in such environments are reached as the result of successful negotiation processes in which agents exchange proposals and information with the aim of convincing the other agent to accept a deal, sometimes using argumentation. Because of the repetition of negotiation dialogues, negotiations get quicker since agents can learn the preferences of others from previous discussed agreements. The agents can guess which are the beliefs, desires and intentions of other agents just analyzing the past dialogues and the state of the game. And as time goes by, agents can observe how their counterparts honour up the agreements they sign. This information can be used to build a model of the other agents' behaviour. This model will help in future negotiations, even to decide which agent should we negotiate with. Concepts like trust, honour, sincerity, and

is used for those units that are placed in a region that is at distance two, that is, they are units that are direct neighbours of a direct neighbour.

[3]The average number of neighbour units.

[4]The average number movements for a unit in every turn are: 1 hold, 4 movements to the 4 adjacent regions, 2 supports to hold to the 2 neighbors and $4 \cdot 2$ supports to move for every two neighbors of our 4 adjacent regions. Therefore $1 + 4 + 2 + 4 \cdot 2 = 15$.

others can summarize the perception that we have of an agent. The reputation of an agent can also be taken into account because agents can also talk (gossip) about other agents performance, promises, intentions, ... The problem is really rich.

In 1985, Daniel Lehmann decided to work on creating a software player (bot) of the game. This ended some years later with the PhD of Sarit Kraus and two master thesis [31]. Since then, several other researchers have also tried to build bots but without much negotiation capabilities [32]. Despite this little success, we think that now is the moment to continue this work. Computer and network technologies have evolved so much in the interim that many people now accept entertainment online from their homes as a matter of course. Thus, it is now much easier to find people interested in playing Diplomacy online against our agents.

4. Negotiation language

The complexity of building an agent capable to negotiate is correlated to the complexity of the language that the agent must be able to understand. The higher the language complexity the higher the richness of the models underpinning agent architectures. In this section we structure the expressions of increasing levels of complexity via a modular, flexible and reusable language hierarchy L. We propose it as a standard for the dialectical communication between the agents that use the testbed and provide infrastructure to support the language parsing. Nonetheless, other languages could be used as the testbed is quite modular and the language tools (i.e. parsers) are separated from the game engine.

Figure 2 graphically represents the language hierarchy L that is defined as an eight level hierarchy; starting from L_1 and increasing the expressiveness as the language level increases. Check the definition of L in Figure 1. The higher the language level that we use, the more complex the actions and the predicates, and thus the expressivity. If it is desired, some of the levels could be skiped, such us for instance Level 5 of sharing feelings. By this way, you can reach level 8 corresponding to argumentation without expressing any feelings. We kept a linear approach for two reasons. One for simplicity, as it gives you a clear roadmap to building ever more complex negotiation agents. Second, to follow the familiar level ordering in the languages proposed by

Level 1: Negotiating a deal
L_1 ::= propose$(\alpha, \beta, \mathit{deal}_1)$ | accept$(\alpha, \beta, \mathit{deal}_1)$ |
reject$(\alpha, \beta, \mathit{deal}_1)$ | withdraw(α, β)
deal_1 ::= Commit$(\alpha, \beta, \varphi)^+$ | Agree(β, φ)
φ ::= $\underline{\mathit{predicate}}$ | Do$(\underline{\mathit{action}})$ | $\varphi \wedge \varphi$ | $\neg\varphi$
β ::= α^+
α ::= $\underline{\mathit{agent}}$

Level 2: Sharing information
L_2 ::= L_1 | inform$(\alpha, \beta, \mathit{info}_2)$
info_2 ::= deal_1 | Obs(α, β, φ) | Belief(α, φ) |
Desire(α, φ) | $\mathit{info}_2 \wedge \mathit{info}_2$ | $\neg\mathit{info}_2$

Level 3: Asking for direct information
L_3 ::= L_2 | inform$(\alpha, \beta, \mathit{info}_3)$ | query$(\alpha, \beta, \mathit{info}_3)$ |
answer$(\alpha, \beta, \mathit{info}_3)$
info_3 ::= info_2 | Unknown$(\alpha, \mathit{info}_3)$ | $\mathit{info}_3 \wedge \mathit{info}_3$ | $\neg\mathit{info}_3$

Level 4: Asking for indirect information
L_4 ::= L_3 | inform$(\alpha, \beta, \mathit{info}_4)$ | query$(\alpha, \beta, \mathit{info}_4)$ |
answer$(\alpha, \beta, \mathit{info}_4)$ | inform(α, β, L_4) | query(α, β, L_4) |
answer(α, β, L_4)
info_4 ::= info_3 | Unknown$(\alpha, \mathit{info}_4)$ | Unknown(α, L_4) |
$\mathit{info}_4 \wedge \mathit{info}_4$ | $\neg\mathit{info}_4$

Level 5: Sharing feelings
L_5 ::= L_4 | inform$(\alpha, \beta, \mathit{info}_5)$ | query$(\alpha, \beta, \mathit{info}_5)$ |
answer$(\alpha, \beta, \mathit{info}_5)$ | inform(α, β, L_5) | query(α, β, L_5) |
answer(α, β, L_5)
info_5 ::= info_4 | Unknown$(\alpha, \mathit{info}_5)$ | Unknown(α, L_5) |
Feel$(\alpha, \mathit{feeling})$ | $\mathit{info}_5 \wedge \mathit{info}_5$ | $\neg\mathit{info}_5$
$\mathit{feeling}$ ::= $\mathit{VeryHappy}$ | Happy | Sad | Angry

Level 6: Taking into account the passage of time
L_6 ::= L_5 | propose$(\alpha, \beta, \mathit{deal}_6, t)$ | accept$(\alpha, \beta, \mathit{deal}_6, t)$ |
reject$(\alpha, \beta, \mathit{deal}_6, t)$ | withdraw(α, β, t) | inform$(\alpha, \beta, \mathit{info}_6, t)$ |
query$(\alpha, \beta, \mathit{info}_6, t)$ | answer$(\alpha, \beta, \mathit{info}_6, t)$ | inform(α, β, L_6, t) |
query(α, β, L_6, t) | answer(α, β, L_6, t)
info_6 ::= info_5 | deal_6 | Obs$(\alpha, \beta, \varphi_6, t)$ | Belief(α, φ_6, t) |
Desire(α, φ_6, t) | Unknown$(\alpha, \mathit{info}_6, t)$ | Unknown(α, L_6, t) |
Feel$(\alpha, \mathit{feeling}, t)$ | $\mathit{info}_6 \wedge \mathit{info}_6$ | $\neg\mathit{info}_6$
deal_6 ::= deal_5 | Commit$(\alpha, \beta, \varphi_6, t)^+$ | Agree(β, φ_6, t)
φ_6 ::= $\underline{\mathit{predicate}}$ | Do$(\underline{\mathit{action}}, t)$ | $\varphi_6 \wedge \varphi_6$ | $\neg\varphi_6$ | $\varphi_6; \varphi_6$
t ::= $\underline{\mathit{time}}$

Level 7: Explaining
L_7 ::= L_6 | inform$(\alpha, \beta, \mathit{info}_7, t)$ | query$(\alpha, \beta, \mathit{info}_7, t)$ |
answer$(\alpha, \beta, \mathit{info}_7, t)$ | inform(α, β, L_7, t) | query(α, β, L_7, t) |
answer(α, β, L_7, t)
info_7 ::= info_6 | Unknown$(\alpha, \mathit{info}_7, t)$ | Unknown(α, L_7, t) |
Explain(info_7, t) | Explain(L_7, t) | $\mathit{info}_7 \wedge \mathit{info}_7$ | $\neg\mathit{info}_7$

Level 8: Arguing
L_8 ::= L_7 | inform$(\alpha, \beta, \mathit{info}_8, t)$ | query$(\alpha, \beta, \mathit{info}_8, t)$ |
answer$(\alpha, \beta, \mathit{info}_8, t)$ | inform(α, β, L_8, t) | query(α, β, L_8, t) |
answer(α, β, L_8, t)
info_8 ::= info_7 | Unknown$(\alpha, \mathit{info}_8, t)$ | Unknown(α, L_8, t) |
Explain(info_8, t) | Explain(L_8, t) | Attack$(\mathit{info}_7, \mathit{info}_7)$ |
Support$(\mathit{info}_7, \mathit{info}_7)$ | $\mathit{info}_8 \wedge \mathit{info}_8$ | $\neg\mathit{info}_8$

Figure 1: Language hierarchy definition in BNF. Note that: $\mathit{expression}^+$ denotes a non-empty sequence of $\mathit{expression}$, non terminal symbols are written in italic, and undefined symbols (referring to terms in the ontology) appear in underlined italics.

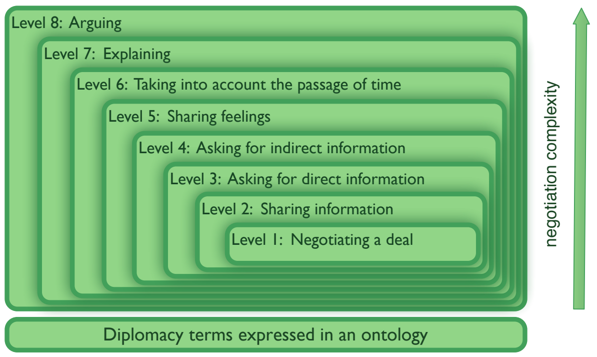

Level 8: Arguing
Level 7: Explaining
Level 6: Taking into account the passage of time
Level 5: Sharing feelings
Level 4: Asking for indirect information
Level 3: Asking for direct information
Level 2: Sharing information
Level 1: Negotiating a deal
negotiation complexity
Diplomacy terms expressed in an ontology

Figure 2: Language hierarchy. Each language L_i extends the languages in lower levels, that is, if there is no re-writing rule for a term in L_i then it can be found in lower levels L_j, with $j < i$.

year ::= $\underline{\text{integer}}$
phase ::= spr | sum | fal | aut | win
power ::= fra | eng | tur | rus | ita | aus | ger
coast ::= ncs | scs | ecs | wcs
$\mathit{regionType}$ ::= amy | sea | coast
$\mathit{supplyCenter}$ ::= spa | mar | par | stp | ...
$\mathit{province}$ ::= $\mathit{supplyCenter}$ | gas | bur | sil | tus | ...
region ::= Region$(\mathit{province}, \mathit{regionType})$
unit ::= Unit$(\mathit{power}, \mathit{region})$
order ::= hld(unit) | mto$(\mathit{unit}, \mathit{region})$ | sup$(\mathit{unit}$, hld$(\mathit{unit}))$ |
sup$(\mathit{unit}$, mto$(\mathit{unit}, \mathit{region}))$ | rto$(\mathit{unit}, \mathit{region})$ | dsb(unit) |
bld(unit) | rem(unit) | wve(power)
offer ::= pce(power^+) | aly$(\mathit{power}^+, \mathit{power}^+)$

Figure 3: Diplomacy ontology.

agent ::= power
action ::= order
$\mathit{predicate}$ ::= offer
time ::= $\langle \mathit{phase}, \mathit{year} \rangle$

Figure 4: Connexion between L and the ontology.

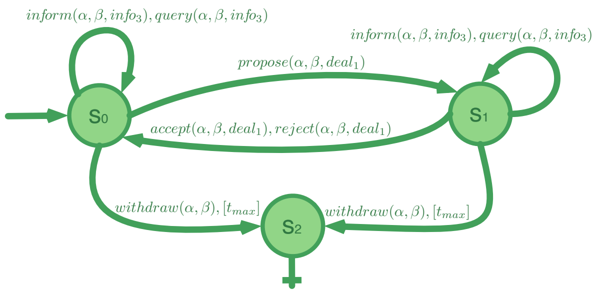

$inform(\alpha, \beta, info_3), query(\alpha, \beta, info_3)$
$inform(\alpha, \beta, info_3), query(\alpha, \beta, info_3)$
$propose(\alpha, \beta, deal_1)$
S0
S1
$accept(\alpha, \beta, deal_1), reject(\alpha, \beta, deal_1)$
$withdraw(\alpha, \beta), [t_{max}]$
S2
$withdraw(\alpha, \beta), [t_{max}]$

Figure 5: A communication protocol for L_3.

the DAIDE community.[5] Researchers set their experiments on DipGame selecting the language level to use.

L is a generic language that could be used for many other applications. It defines the illocutions that the agents can use to communicate and the basic concepts like *Agree*, *Desire*, *Feel*, etc. The language is parametric on the vocabulary for a specific application domain, described as an ontology. The undefined non terminal symbols that appear in L should be specifically defined for each application domain. These symbols are: *time*, *agent*, *action* and *predicate*. Figure 3 represents the ontology for the Diplomacy game and Figure 4 contains the connexion between the ontology and L. In this way, we allow researchers to reuse as much code as possible when applying their work to real world applications after testing it with DipGame.

In the rest of this section, we describe and illustrate the expressivity of each language level in Diplomacy. For instance, `Unit(rus, Region(stp, scs))` is a term meaning that 'There is a unit from Russia in the south coast of Saint Petersburg', `pce([ita rus])` is a predicate meaning 'Peace between Italy and Russia', and `sup(Unit(rus, Region(spa, ecs)), mto(Unit(ita, Region(mar, amy)), Region(par, amy)))` is an example of action where 'The unit of Russia in the east coast of Spain supports the movement of the Italian army in Marseilles to Paris'.

L_1**: Negotiating a deal.** This is the first language level. It allows agents to negotiate deals following the protocol illustrated in Figure 5. The deals can be either a sequence of commitments, one for every agent involved in the deal, or a global agreement in which a set of agents agree on something, usually the truth of a predicate. Here you have two examples of sentences in L_1:

E.g. 'Italy proposes to Russia a deal by which Italy commits to do a movement from its army in Marseilles to Paris and Russia commits to support the Marseilles italian army's movement with the unit in the east coast of Spain':[6]

```
propose(ita, rus,
   [Commit(ita,rus,
        Do(mto(Unit(ita, Region(mar, amy)),
               Region(par, amy))))
   Commit(rus, ita,
        Do(sup(Unit(rus, Region(spa, ecs)),
               mto(Unit(ita, Region(mar, amy)),
```

[5]The Diplomacy AI Development Environment (DAIDE) is a community that creates bots with good strategies to play Diplomacy. They propose a language standard for communication. We use their protocol and their language at level 0 that roughly corresponds to our ontology language.

[6]We abuse notation and represent sets of agents containing just one agent by the agent name itself. e.g. Commit(`ita`, [`rus`], *deal*) will be represented as Commit(`ita`, `rus`, *deal*).

```
                    Region(par, amy)))))])
```

E.g. 'Italy accepts to Agree with Russia that they are allied against England':

```
accept(ita, rus, Agree([ita rus], aly([ita rus], eng)))
```

L_2: **Sharing information.** This language level adds the ability of sharing information with other agents. It can be information about previous commitments, observed actions, beliefs, desires or deals.
E.g. 'Italy informs England that Italy keeps a peace agreement with Russia':

```
inform(ita, eng, Agree([ita rus], pce([ita rus])))
```

L_3: **Asking for direct information.** At level three, agents can request other agents for information. Answers to queries are similar to informs.
E.g. 'England asks Italy whether Italy and Russia have a peace agreement':

```
query(eng, ita, Agree([ita rus], pce([ita rus])))
```

E.g. 'Italy answers England that Italy and Russia do have a peace agreement':

```
answer(ita, eng, Agree([ita rus], pce([ita rus])))
```

L_4: **Asking for indirect information.** Level four allows to inform about dialogical moves between agents.
E.g. 'Russia asks Italy whether Italy answered to England that Italy and Russia had a peace agreement':

```
query(rus, ita,
      answer(ita, eng,
             Agree([ita rus], pce([ita rus]))))
```

L_5: **Sharing feelings.** This level is the emotional one. Feelings can be exchanged between agents.
E.g. 'Italy asks Russia whether Italy's answer to England that Italy and Russia had a peace agreement made Russia feel sad':

```
query(ita, rus,
      answer(ita, eng,
             Agree([ita rus], pce([ita rus]))) →
      Feel(rus, Sad))
```

L_6: **Taking into account the passage of time.** L_6 adds time to L_5. Within L_6 we can speak about the past and make promises about the future. Time is added as an extra argument to predicates and illocutions. Time variables are considered universally quantified. In subsequent levels, L_7 and L_8, we omit time to simplify notation.
E.g. 'Russia informs Italy that if Italy informs in the future to any power that Italy and Russia have a peace agreement, then Russia will feel Angry':

```
inform(rus, ita,
       (inform(ita, power,
               Agree([ita rus],
                     pce([ita rus])), t1) ∧ t1 > t0) →
       (Feel(rus, Angry, t2) ∧ t2 > t1),
       t0)
```

L_7**: Explaining.** Dialogues often include explanations and explanation requests. This level adds that possibility to allow agents to explain why things are like they are.
E.g. 'Italy asks Russia for an explanation of why the fact that Turkey believes that there is a peace agreement between Russia and Italy makes Russia feel Angry':

```
query(ita, rus,
      Explain(
           Belief(tur,
                  Agree([ita rus], pce([ita rus]))) →
           Feel(rus, Angry)))
```

L_8**: Arguing.** And finally, level 8 allows agents to express rebuttals and supports between arguments.
E.g. 'Russia informs England that its alliance with Italy against England and Italy's desire to conquer Paris together support the imminent Italian attack from Marseilles to Paris':

```
inform(rus, eng, Support(
   Agree([ita rus],aly([ita rus],eng)) ∧ Desire(ita, par),
   Do(mto(Unit(ita, Region(mar, amy)), Region(par, amy)))))
```

5. Infrastructure

In this section we give some minimal technical information to a potential developer on what kind of support/help the platform offers. Readers aiming at a general understanding of the testbed can skip this section. The infrastructure is modular and freely available at `http://www.dipgame.org`. It is composed of: the game engine, a framework for agent development and negotiation tools. In this section we describe them giving special attention to the negotiation tools.

The game engine is structured as a client-server program following the guidelines of DAIDE. The server is the game manager. It receives the movement decisions of the players, updates the game state and broadcasts it. Nowadays, there are two software alternatives for the server: *parlance*[7] and *AiServer*.[8] There are two types of clients: observers and players. Both receive the information about the state of the game but only players can send movements. Players can be autonomous (bots) or humans. There are several bots available online but they focus on the strategy and tactics of the game rather than on the dialectic moves, that is, they do not focus on negotiation.

Developing a client from scratch is tedious. That is why we provide *dip*, a java framework that copes with the communication with the game manager

[7]Parlance is a multiplatform game manager (`http://pypi.python.org/pypi/Parlance`).

[8]AiServer is DAIDE's game manager (`http://www.ellought.demon.co.uk/dipai/`).

and the representation of the game state and the movements (referred in the game as orders to send to the units). *dip* is very easy to use, for instance, creating a player means just to implement a new class extending the abstract class *Player* (see Figure 6) and adding the extra functionality to decide what movements to do next, as explained in Figure 7. The rest of the work is done by the framework itself. To illustrate how to use *dip* we provide *ConsoleObserver* and *ConsolePlayer* that are console applications that allow a user to observe a game and play respectively.

Deciding what movement to do next usually requires to search the space of possible actions to perform. For those researchers that want to test their work and are not interested in the search process, we provide an extension of *dip*, called *bot*, that calculates the potential actions that the bot may choose based on an action evaluation function that the researcher defines. This simplifies even more the implementation of a player because it then consists basically on defining an evaluation function. This function can be defined as a combination of the implementation of three class interfaces: *RegionEvaluator*, *OrderEvaluator* and *OptionEvaluator*. Each class interface is in fact an evaluation function itself over a different dimension (region, order and option). The combination of them is thus not optimal (as it considers them in isolation) but is a good trade-off between memory usage and solution quality. The trade-off level is fixed by the programer: the more memory the higher the solution quality. An example of bot implemented using *bot* is *RandomBot* whose evaluation function always returns 0. Figures 6 and 7 summarize the content and use of the infrastructure.

We also provide support for the negotiation between players. We take the language L as a standard for this testbed and provide a parsing utility called *dipNego* that checks the syntax of messages and represents them in a structure of objects. In Figure 9 we illustrate a class diagram with the most relevant classes of *dipNego* that are necessary to represent the messages in L_1. There is a parser available for all L language levels and the Diplomacy ontology. Nonetheless, as L is domain independent, *dipNego* can be used for other application domains.

The negotiation dialogues between players are handled independently from the game engine. Messages do not use DAIDE's protocol, instead, players negotiate using *negoServer*, an instant messaging program specially created for this testbed, and *negoClient*, a library that implements the functionality required to connect a client with the negotiation server. *RandomNegoBot*, available online, is an extension of *RandomBot* able to randomly

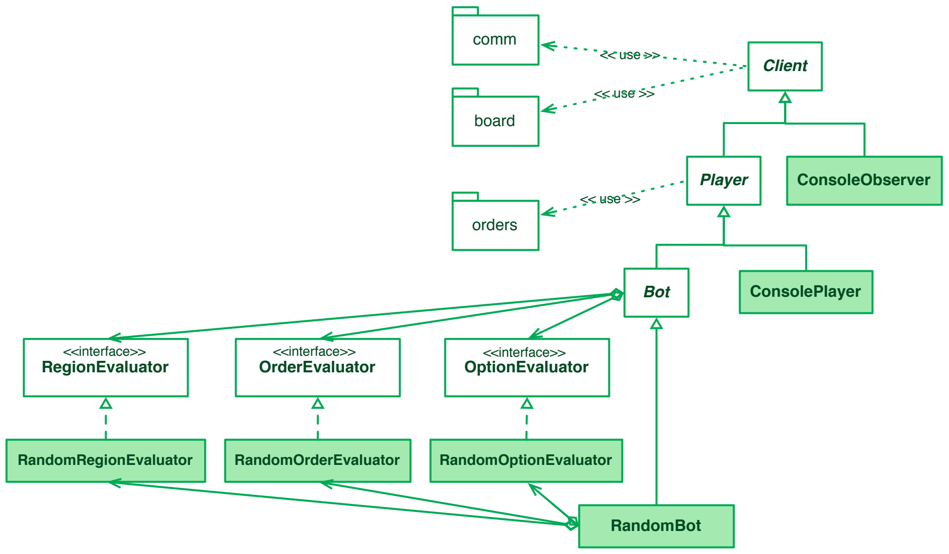


Figure 6: UML 2.0 class diagram of the most important classes in the framework.

Client type	Plays?	Provided funct.	Required funct.	Required library	Objects to implement	Implement. example
observer	no	game state	nothing	dip	Observer	ConsoleObserver
player	yes	game state	action selection	dip	Player	ConsolePlayer
player	yes	game state, best action	action evaluation	dip+bot	Bot RegionEvaluator OrderEvaluator OptionEvaluator	RandomBot

Figure 7: Bot development framework.

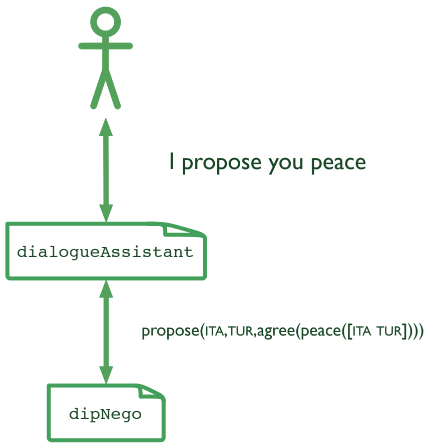


Figure 8: Message transformation.

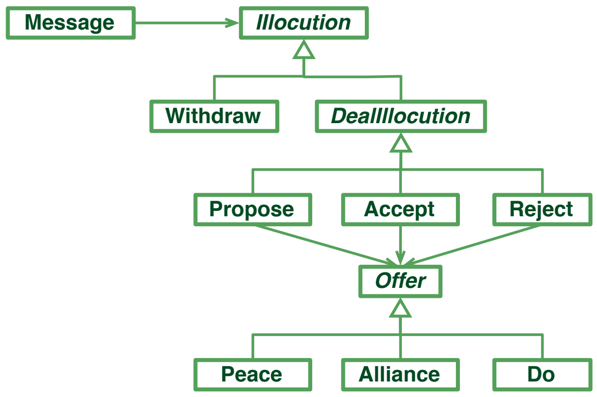


Figure 9: UML 2.0 class diagram of the most important classes for L_1 in *dipNego*.

negotiate with other players. We exemplify the usage of the testbed with *RandomNegoBot* in section 6.

In section 7 we discuss the use of this testbed by humans. In that section we describe *dialogueAssistant*, a restricted natural language interpreter that translates the messages written by humans into L.

6. Using DipGame

Building agents capable to negotiate playing Diplomacy is quite easy with the framework and the tools introduced in the previous section. In this section we suggest a methodology for testing agents with DipGame and exemplify it describing how can we create and test a very simple agent: a random negotiator, that is, a player that performs random moves and negotiates also randomly. It is based on *RandomBot*'s code.[9]

The simple methodology consists of six steps that end with the analysis of the results and the optional improvement of the agent:

1. Download all resources from `http://www.dipgame.org`. The required resources will depend on the type of client that you want to create and the implementation options that you decide to use, see Figure 7.
   For our example we need the game manager called *Parlance*, *negoServer*, *RandomBot*'s code and the libraries that it requires, that are dip and bot.[10]
2. Create a client extending the corresponding classes. Remember that to be able to negotiate you should use *negoClient* and *dipNego* in addition to the libraries specified in Figure 7.
   We extend *RandomBot*'s functionality adding the capability to negotiate randomly thus we also need the language parsing utility, *dipNego*, and the library that provides the connection with *negoServer* that is *negoClient*. From *negoClient* we must implement the *handleMessage* method indicating what to do when a message is received. Our random negotiator agent would throw a coin to decide whether to accept or reject the received proposal. And if the message is already an accept or a reject, it would do nothing. *negoClient* also allows us to send messages. Our random negotiator agent would throw a coin at every new state

[9] *RandomBot* is the player that performs random moves introduced in section 5.

[10] dip and bot libraries are available at `http://www.dipgame.org/browse/dip`.

to decide whether to propose something or not. And the same method would be used to decide whom to propose it and the concrete offer to make, for instance, a peace agreement.

3. Complete your client adding the negotiation model functionality to be tested. This is the step where you integrate your work with the testbed. In the example we take decisions randomly therefore the negotiation model simply defines a negotiation strategy based on throwing coins.
4. Set the experiments choosing the language level, the duration of the game and the deadlines. Fixed game duration and deadlines for turns are optional but recommended specially when humans take part of the experiments.
   Our example experiment might be composed of 100 games with 7 instances of our random agent. The duration of the game and turns are undefined as this is a simple example where agents make just one proposal per turn before deciding their actions.
5. Run the experiments. This means launching first the game manager and *negoServer* with the desired settings and then running the clients that you want to connect to them.
   We will repeat this 100 times and save the results.
6. Analyze the results and extract your conclusions. At the end of each game you get the results as a complete log of movements, state updates and exchanged messages.
   From the log files generated we check whether our agent played well and question whether there is a way to improve its performance changing the internal model of the agent.
7. Optionally make the adjustments necessary in your code to improve the performance of your agent.

The code of this random negotiator agent can be download from `http://www.dipgame.org/downloads/RandomNegoBot.java`. It is just a simple example where the negotiation is driven taking random decisions. More sophisticated agents will take advantage of the messages exchanged, the actions performed and the state of the world observed in order to perform smart negotiations. Those agents can compete against other agents but also against humans.

7. DipGame website

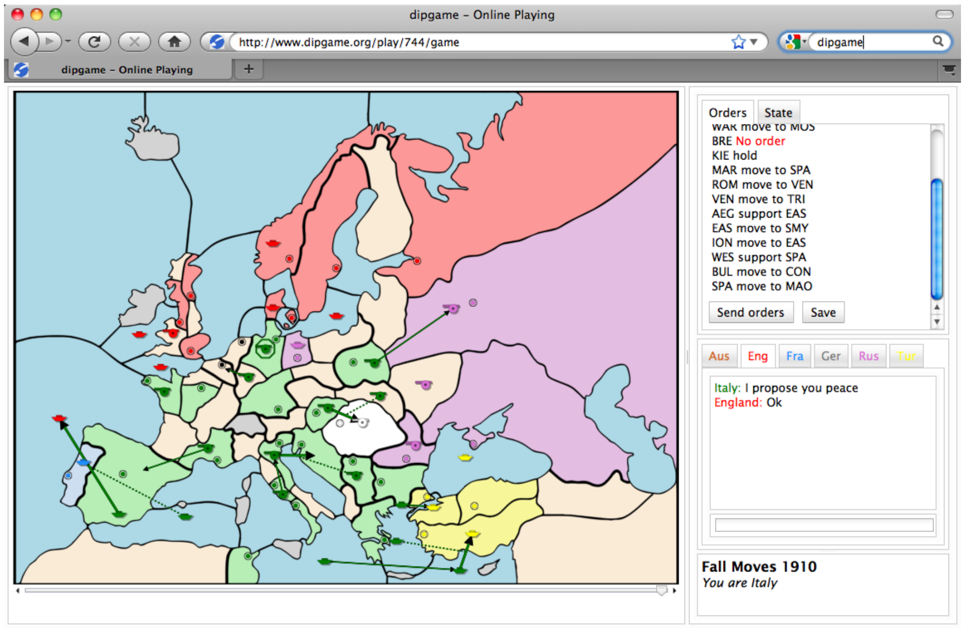
Our aim is to provide a platform with an environment rich enough to test negotiation models that are going to be deployed into the real world. Most of the real world applications of MAS negotiations involve people. Therefore, it should be possible to have humans taking part of our experiments. In this section we describe the problem of having people taking part in research experiments and how do we handle their recruitment.

Joining together hundreds of people in an experiment is costly. It is very hard to organize an experiment with humans because this requires a lot of money, time and effort in coordinating everybody. To be able to perform experiments like these but with a low cost we created a web application to allow people to take part in the experiments from anywhere in the world reducing the logistic problems of gathering people. Although the game is quite popular, it is never mind difficult for a player to physically summon another 6 people to play a game. Playing online makes this easier and moreover allows to secretly meet with other players to negotiate and keep conspiracies under cover. By means of this web application, we join together research and entertainment as it was successfully done previously in other projects.[11]

The web application interface is composed of an interactive map representing the state of the game and where players can indicate their movements clicking on the units. It is quite intuitive how to use it and there are some help facilities for newcomers to easy step into the game. In Figure 10 we include a screenshot of an ongoing game where Italy is about to win. At the right part of the screen there is a summary of the state of the game and the turn movements. Under this summary panel there is an instant messaging tool embedded in the web page that allows the player to negotiate with the rest of players using restricted natural language. To that end, we provide a parser for restricted natural language. It is called *dialogueAssistant* and translates written sentences into an equivalent sentence in L. Figure 8 illustrates the transformation. *dialogueAssistant* is currently only available for messages at level L_1 but is going to be upgraded in order to interpret higher language levels. For the top levels it seems that a click-based approach to build sentences would be more useful. This approach would be something similar to the translator sheets that the face-to-face players use in international tournaments[12]. Also, an argumentative agent might use argument building tools, like those in `http://debategraph.org/`, to structure argumentative

[11]In 2005, Luis von Ahn devised the ESP Game, an online game of image labeling that Google is now using to improve its image search results.

[12]Examples of translator sheets at `http://www.ellought.demon.co.uk/dip_translator/`.



```
WAR move to MOS
BRE No order
KIE hold
MAR move to SPA
ROM move to VEN
VEN move to TRI
AEG support EAS
EAS move to SMY
ION move to EAS
WES support SPA
BUL move to CON
SPA move to MAO
```

Italy: I propose you peace
England: Ok

Fall Moves 1910
You are Italy

Figure 10: Screenshot of the dipgame web application for playing Diplomacy.

dialogues and support decision making.

8. Related Work

In this section we summarize existing work on testbeds for negotiation.

MAGNET is the abbreviation for Multi AGent NEgotiation Testbed. It is rather a generalised market architecture for multiagent contract negotiation using auctions [33]. The required negotiation skills in MAGNET are far simpler than in DipGame.

The Trading Agent Competition (TAC) [29] is a testbed where, in its classical version, agents are travel assistants that try to assemble the best travel package for their clients. To evaluate the assembling of packages they use the summation of the utilities of the clients that are in turn based on their preferences. It is a multi-issue negotiation process organised as a multi-lateral negotiation by means of auctions on every single issue. Bids take the form of pairs where a price per item and a number of items is proposed. There are also other versions of TAC problems that are similar to this: TAC/AA, CAT and TAC/SCM. Competitions are organised annually where several instances of the game are played. Although this is a consolidated testbed, the problems that it aims at are quite different to the ones we worry about. The negotiation language and protocol is highly constrained, humans do not take part of experiments and preferences are assumed to be known and fixed.

The Colored Trails Game (CT) [28] is a research testbed for decision-making in groups comprising people and agents. In every game, two or more players may negotiate to exchange chips that allow them to move from their source position to a target position over a coloured squared board. To move through the board, players must provide chips of the same colour of the squares that they want to pass over. The game is quite abstract and was defined for research purposes so humans taking part in the experiments must be instructed and are not equally motivated. The amount and type of chips and the source location of the players is different so that the initialisation of the game can benefit a player in front of others. Also the richness of the negotiation language is more limited than in the case of DipGame.

GENIUS, the Generic Environment for Negotiation with Intelligent multi-purpose Usage Simulation [27] is a research tool that facilitates research in the area of bilateral multi-issue negotiation. It allows the evaluation of agents playing in various negotiation scenarios where issues and preferences of each party on them must be known from the beginning and are fixed. It can be used in experiments with humans using a constrained communication protocol, but assuming that the preferences are fixed and known is something strange and difficult to find in the real world. Thanks to this, the toolbox for analysis calculates the optimal solutions and represent the evolution of the negotiation using it as a reference. Contrarily to DipGame, GENIUS is limited to bilateral negotiations, there is no relationship building among players, and there is no post-negotiation action verification.

The Agent Reputation and Trust (ART) Testbed, [34], was created with the aim of establishing a testbed for agent trust- and reputation-related technologies. Several competitions were organised where appraiser agents competed to get the higher reputation appraising paintings. It is currently still very popular although the project is no longer maintained. The expertise on completing appraisals was split between the agents and they should decide whether to request help from other appraisals, paying an amount for the info, or dealing with it by themselves. ART testbed is a good testbed for trust and reputation but it is very focused on a utilitarian view and too limited to test negotiation or argumentation models.

These testbeds provide more or less infrastructure and some have a long tradition background being now a reference to many researchers. However, they are too connected to a utilitarian approach and we wanted a scenario in which other approaches could also be tested. In particular, we think that DipGame is richer to model information exchange (due to the possibility of observing public -the moves- and private -the offers- behaviour of agents) and to model trust (due to the longer time that a game lasts). No one of the related testbeds is rich enough to allow for the testing of those complex negotiation models that can be tested in DipGame. The simplest setting of DipGame is already richer and more realistic than any of the existing testbeds. This is so because we wanted to encourage the research on negotiating agents to be capable to interoperate with people, and the other testbeds goal mostly, follows a constructivist approach as mentioned in the introduction. Among all the testbeds described above, the CT is the most compelling to work with humans.

9. Discussion and future work

Diplomacy is an ideal game for testing negotiation models because it allows for complex dialogues among agents while keeping the computational overhead to a minimum as the number of involved agents is small. Moreover, it allows testing of rich negotiation models as the game has a huge space of solutions, actions are only partially observable (e.g. some dialogical actions of other agents are not made public) the state changes over time and there are thousands of humans ready to play. In this paper we introduced the DipGame testbed that allows researchers to run experiments using such negotiation models over Diplomacy. We provide a platform with a complete infrastructure composed of a framework for building agents, tools for communication, interpreters for restricted natural language, and a website to summon people for the experiments.

The DipGame website is in production as beta version at `http://www.dipgame.org` giving access to humans to play Diplomacy online against some of our simplest bots. Although we did not make any advertisement to the Diplomacy players community, the website gets requests from many users to play. We have a total of 185 registered users in April 2011 and there are several groups of researchers developing bots over the platform. Also, the testbed is being used for academic purposes in some master degrees (e.g. `http://www.cse.unr.edu/robotics/bekris/cs483_s10/handouts`) and by several master and PhD students in their theses.

We will continue maintaining the website and the resources and adding functionality. Our next work will be mainly focussed on developing sophisticated negotiating agents capable of playing against humans at the different language levels of L. In fact, this testbed was needed as a step before being able to test our own work on negotiation [35]. Obtaining experimental results of our negotiation model over the testbed is part of our future work.

10. Acknowledgments

Research supported by the Agreement Technologies CONSOLIDER project under contract CSD2007-0022 and INGENIO 2010, by the Agreement Technologies COST Action, IC0801, and by the Generalitat de Catalunya under the grant 2009-SGR-1434.

References

[1] C. Sierra, J. K. Debenham, Trust and honour in information-based agency, in: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-agent Systems, 2006, pp. 1225–1232.

[2] C. Sierra, J. Debenham, Information-based reputation, in: First International Conference on Reputation: Theory and Technology, Gargonza, Italy, 2009, pp. 5–19.

[3] I. Pinyol, J. Sabater-Mir, Pragmatic-strategic reputation-based decisions in bdi agents, in: Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems, 2009, pp. 1001–1008.

[4] L. Amgoud, H. Prade, Using arguments for making and explaining decisions, Artificial Intelligence 173 (3-4) (2009) 413–436.

[5] T. Alsinet, C. Chesñevar, L. Godo, G. Simari, A logic programming framework for possibilistic argumentation: Formalization and logical properties, Fuzzy Sets and Systems 159 (10) (2008) 1208–1228.

[6] P. Faratin, C. Sierra, N. R. Jennings, Negotiation decision functions for autonomous agents, Robotics and Autonomous Systems 24 (3-4) (1998) 159–182.

[7] S. Kraus, Negotiation and cooperation in multi-agent environments, Artificial Intelligence 94 (1997) 79–97.

[8] K. V. Hindriks, C. Jonker, D. Tykhonov, Towards an open negotiation architecture for heterogeneous agents, in: Proceedings of the 12th international workshop on Cooperative Information Agents, Prague, Czech Republic, 2008, pp. 264–279.

[9] H. Raiffa, Negotiation Analysis: The Science and Art of Collaborative Decision Making, Harvard U.P., 2002.

[10] S. Kraus, Strategic Negotiation in Multiagent Environments, MIT Press, 2001.

[11] J. S. Adams, Inequity in social exchange, in: L. Berkowitz (Ed.), Advances in experimental social psychology, Vol. 2, New York: Academic Press, 1965.

[12] H. Sondak, M. A. Neale, R. Pinkley, The negotiated allocations of benefits and burdens: The impact of outcome valence, contribution, and relationship, Organizational Behaviour and Human Decision Processes (3) (1995) 249–260.

[13] M. H. Bazerman, G. F. Loewenstein, S. B. White, Reversal of preference in allocation decisions: judging an alternative versus choosing among alternatives, Administration Science Quarterly (37) (1992) 220–240.

[14] G. Coricelli, R. Nagel, Neural correlates of depth of strategic reasoning in medial prefrontal cortex, Proceedings of the National Academy of Sciences (PNAS): Economic Sciences 106 (23) (2009) 9163–9168.

[15] C. Eisenegger, M. Naef, R. Snozzi, M. Hienrichs, E. Fher, Sequence of testosterone vs. placebo on ultimatum game behavior in women, Nature (463) (2010) 356–359.

[16] V. L. Smith, Constructivist and ecological rationality in economics, The American Economic Review 93 (3) (2003) 465–508.

[17] M. J. Osborne, A. Rubinstein, Bargaining and Markets, Academic Press, 1990.

[18] P. Faratin, C. Sierra, N. Jennings, Using similarity criteria to make issue trade-offs in automated negotiation, Journal of Artificial Intelligence 142 (2) (2003) 205–237.

[19] R. Lin, S. Kraus, J. Wilkenfeld, J. Barry, Negotiating with bounded rational agents in environments with incomplete information using an automated agent, Artificial Intelligence 172 (6-7) (2008) 823 – 851.

[20] J. Rosenschein, G. Zlotkin, Rules of Encounter, MIT Press, 1998.

[21] H. Raiffa, The Art and Science of Negotiation, Harvard U.P., 1982.

[22] C. Sierra, J. Debenham, The logic negotiation model, in: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-agent Systems, 2007, pp. 1026–1033.

[23] Y. Gal, B. Grosz, S. Kraus, A. Pfeffer, S. Shieber, Agent decision-making in open mixed networks, Artificial Intelligence 174 (2010) 1460–1480.

[24] D. Hales, Cpm-03-109: Neg-o-net — a negotiation simulation test-bed, Tech. rep., Center for Policy Modelling (2002).

[25] R. Conte, J. Sichman, Depnet: How to benefit from social dependence, Journal of Mathematical Sociology 20 (2-3) (1995) 161–177.

[26] A. G. Sanfey, J. K. Rilling, J. A. Aronson, L. E. Nystrom, J. D. Cohen, The neural basis of economic decision-making in the ultimatum game, Science 300 (5626) (2003) 1755–1758.

[27] K. Hindriks, C. M. Jonker, S. Kraus, R. Lin, D. Tykhonov, Genius: negotiation environment for heterogeneous agents, in: Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems, IFAMAS, Richland, SC, 2009, pp. 1397–1398.

[28] Y. Gal, B. J. Grosz, S. Kraus, A. Pfeffer, S. Shieber, Colored trails: A formalism for investigating decision-making in strategic environments, in: IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games, 2005, pp. 25–30.

[29] M. P. Wellman, P. R. Wurman, A trading agent competition for the research community, in: AMEC, IJCAI 1999 Workshop.

[30] S. Kraus, D. Lehmann, E. Ephrati, An automated diplomacy player, in: D. Levy, D. Beal (Eds.), Heuristic Programming in Artificial Intelligence: The 1st Computer Olympia, Ellis Horwood Limited, 1989, pp. 134–153.

[31] S. Kraus, Designing and building a negotiating automated agent, Computational Intelligence 11 (1995) 132–171.

[32] J. Shaheed, Creating a diplomat, Master's thesis, Department of Computing, Imperial College Of Science, Technology and Medicine, 180 Queen's Gate, London, SW7 2BZ, UK (June 2004).

[33] J. Collins, M. Tsvetovat, B. Mobasher, M. Gini, Magnet: A multi-agent contracting system for plan execution, in: Proceedings of Artificial Intelligence and Manufacturing Workshop, 1998, pp. 63–68.

[34] K. K. Fullam, T. B. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. S. Rosenschein, L. Vercouter, M. Voss, A specification of the agent reputation and trust (art) testbed: experimentation and competition for trust in agent societies, in: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, Utrecht, The Netherlands, 2005, pp. 512–518.

[35] A. Fabregues, C. Sierra, An agent architecture for simultaneous bilateral negotiations, in: Proceedings of the 13è Congrés Internacional de l'Associació Catalana d'Intelligència Artificial, Espluga de Francolí, Tarragona, 2010, pp. 29–38.