Decision support for extensive form negotiation games

Sujata Ghosh[1] Thiri Haymar Kyaw[2] and Rineke Verbrugge[2] *,

[1] Indian Statistical Institute, Chennai Centre
SETS Campus, MGR Film City Road
Chennai 600113, India.
sujata@isichennai.res.in
[2] Department of Artificial Intelligence
University of Groningen
The Netherlands
thirihaymarkyaw@gmail.com, rineke@ai.rug.nl

Abstract. This paper presents a tool, NEGEXT, for finding individual and group strategies to achieve certain goals while playing extensive form negotiation games. NEGEXT is used as a model-checking tool which investigates the existence of strategies in negotiation situations. We consider sequential and parallel combinations of such games also. Thus, it may aid students of negotiation in their understanding of extensive game-form negotiation trees and their combinations, as well as in their learning to construct individual and group strategies.

1 Introduction

Negotiation may be found everywhere: From mundane conversations between partners about who will fetch the children from school and who will cook dinner, to the sale of an apartment whilst the seller is trying to hide from the buyer that she has bought a new house already, and to fully-fledged international multi-party multi-issue negotiations about climate control, and so forth. Negotiation is a complex skill, and one that is not learnt easily. Thus, many negotiations are broken off, even when they have potential for a win-win solution. Moreover, in many negotiations that do result in an agreement, one or more participants "leave money on the table": they could have done better for themselves [1]. Thus, it is no wonder that several scientific fields have made contributions to analyzing, formalizing, and supporting negotiation.

Kuhn [2] highlighted the importance of using extensive form games in modeling negotiation situations in an objective way, by focusing on the temporal and dynamic nature of the negotiations. For a general overview on negotiation games, see [3]. Researchers in multi-agent systems investigate many aspects of negotiations: some design negotiation mechanisms [4], some analyze negotiation as a form of dialogue [5], whilst others build software to simulate and support

* This research was supported by the Netherlands Organisation of Scientific Research grant 600.065.120.08N201 and Vici grant NWO 227-80-001

negotiation [6, 7]. In summary, the fast-growing body of research on negotiation provides varied sophisticated models for negotiations.

This paper reports the development of a simple tool, NEGEXT (`http://www.ai.rug.nl/~sujata/negext.html`), written in the platform-independent Java language. NEGEXT has been constructed to aid students of negotiation. This toolkit, based on extensive form games [8], will aid in understanding how to combine negotiations, and in planning one's strategic moves in interaction situations when the opponents' possible moves can be approximated. Even though some visualization tools for extensive form game trees already exist,[3] as well as software for negotiation support,[4] we believe we are the first to make a tree-based negotiation toolkit that incorporates the possibility of representing learning from game to game, by sequential and parallel composition (cf. [9]). Moreover, the toolkit has a model-checking component which computes whether and how an individual or a specific coalition can achieve a given objective.

2 Analyzing Negotiation Situations Using Game Trees

A finite extensive form game can be represented by a finite tree where the nodes correspond to players' positions and edges correspond to moves of the players. The terminal nodes of the tree are the end-points of the game, which are generally termed as leaves of the tree. A strategy for a player i is a subtree of the finite game tree, which consists of single edges from player i nodes, and all possible edges from the other players' nodes. A strategy for a group of players K is a subtree consisting of single edges from player k's nodes, where $k \in K$, and all possible edges from player k''s nodes, where $k' \notin K$. Note that all the players have complete knowledge of the whole game. Let us briefly describe the notion of sequential and parallel combinations of extensive form games and players' strategies in such games, providing an application of each in negotiations.

2.1 Sequential composition

Any two extensive form games can be *sequentially composed* by plugging in the second game at each leaf node of the first game, each leaf node of the first game becoming the root node of the second game [9]. One can extend this idea to define sequential composition of a game with a set of games. Here, at each leaf node of the first game, some game from a given set of games is plugged in.

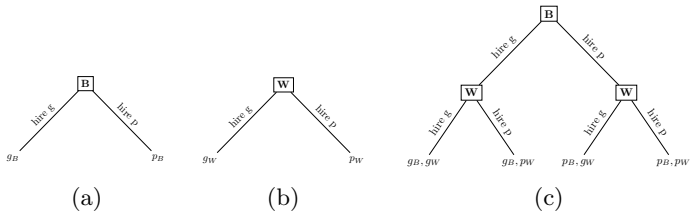
Story-I: Sequential cooperation between two companies We now describe a situation which can be dealt with by combining two trees sequentially. Suppose that two Research & Development companies on biotechnology, Biocon and Wockhard, have just entered into a joint project concerning the discovery of biomarkers. They need to hire two types of specialists, a genomic expert and a proteomic expert, one specialist per company. Unfortunately the two companies did not discuss beforehand which of them should hire which type of expert.

[3] See for example `http://www.gametheory.net/Mike/applets/ExtensiveForm/ExtensiveForm.html` and `http://www.gambit-project.org`

[4] See for example `http://www.negotiationtool.com/`

The hiring process can be represented by the simple game trees in Figure 1a,1b, where the nodes B and W stand for Biocon and Wockhard, respectively. The action "hire g" stands for "hire a genomic expert" and action "hire p" stands for "hire a proteomic expert". The propositional atoms have the following meanings:

g_B: genomics is covered by Biocon	g_W: genomics is covered by Wockhard
p_B: proteomics is covered by Biocon	p_W: proteomics is covered by Wockhard



(a) (b) (c)

Fig. 1. Hiring game trees: 1(c) is the sequential combination of 1(b) after 1(a)

Wockhard notices that it is a good idea to await Biocon's hiring decision. In order to analyze the different possibilities, they combine the two games sequentially (Figure 1c), plugging in the game of Figure 1b at each leaf node of the game of Figure 1a. Figure 1c clearly shows that hiring one expert after another, in a perfect information game, is much better than the imperfect information game represented by the two game trees in Figure 1a, 1b. Wockhard *learns* who was hired by Biocon, and hires someone with complementary expertise. The two 'middle branches' in Figure 1c provide win-win solutions.

2.2 Parallel composition

An *interleaving parallel combination* of two extensive form games, as defined in [9], gives rise to a set of games. The main idea of the interleaving parallel operator is as follows: A play of such a game basically moves from one game to the other. One player can move in one of the games, and in the next instant some other player or even the same player may move in the other game. A parallel game takes care of such interleaving. The different orders in the way moves of the players can be interleaved give rise to different games in a parallel combination of two games. Interleaving parallel games allows for copy-cat moves, and in general enables the transfer of strategies in between games. For the formal details, see [9].

Story-II: Analysis of job application in lockstep synchrony This example has been inspired by the trick regarding *how not to lose while playing chess with a grandmaster* [9], but with additional aspects of translation. Suppose that Prof. Flitwick (FL) applied for a position in the research group of Prof. Sprout (SP), and he turns out to be the favored candidate. Job negotiations N_1 between Flitwick and Sprout are on the verge of starting. In the same period, Prof. Sprout herself applied for a professorship in the department of Prof. Quirrell (QR), and

has just been selected by Prof. Quirrell. The time arises to discuss the particulars of this position as well, in negotiation N_2.

Prof. Sprout is not a very savvy negotiator herself, but she knows that both Prof. Flitwick and Prof. Quirrell are. Moreover, Prof. Sprout notices that the issues of negotiation are quite similar in both cases: The prospective employer can offer either a much higher salary than the standard one, or a standard salary. On the employee's turn, he or she can choose to offer teaching subjects of their own choice, or whatever the department demands (Figure 2).

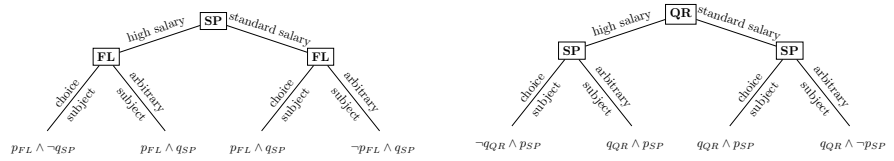


Fig. 2. Appointment game trees for negotiation N_1 (left) and negotiation N_2 (right)

The main pay-offs of the negotiators can be given in terms of propositional letters as follows, where:

- p_i: the new employee i gets more satisfaction in the job (for i is FL or SP);
- q_j: the new employer j is happy with the terms (where j is SP or QR).

In both negotiations, the goal of the new employee can be formulated conditionally: either she procures a higher salary, and then he or she is ready to do whatever he or she is asked; or, if she gets a standard salary, then she would like to teach her favorite subjects only. The goal of the employer would be to have a settlement favorable for the group, i.e. favoring paying a standard salary and / or the new employee teaching according to demand. The salaries and subjects in question are different in both negotiations, but there is a reasonable one-one map of possible offers from one negotiation to the other, represented by the actions of offering "high salary" versus "low salary" and "choice subject" versus "arbitrary subject". The general idea for Prof. Sprout is to be a copy-cat:

- wait for QR to make an offer in negotiation N_2 about the salary;
- translate that offer to the terms of negotiation N_1, and propose a similar offer to FL;
- await the counteroffer from FL about subjects to teach, translate it to the terms of negotiation N_2, and make that offer to QR.

To model this, one needs to compose the trees in Figure 2 in a parallel fashion with interleaving moves, see Figure 6. It is clear that winning proposals are possible in both subtrees, namely on the two 'inside branches' of each, and therefore also in the parallel copy-cat negotiation.

3 NEGEXT toolkit

We have developed the NEGEXT toolkit to aid students in strategic interactions during negotiations, with negotiations represented as extensive form games and

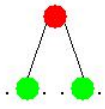
their sequential and parallel combinations. NEGEXT has been written in Java version 1.7.0_02 using the Eclipse editor. It can run on any system that has a Java Virtual Machine (JVM) or Java-enabled web browsers. NEGEXT uses an applet to display the graphical user interface. The user can draw game trees using a menu of input nodes. NEGEXT can also generate the sequential combination of two game trees and all possible parallel combinations. Another feature of the software is that for both individual players and coalitions, it can check whether a strategy to achieve a proposition exists, and if so, point to such a winning strategy in the tree. We used a modified form of binary tree data structure for organizing the tree nodes.

The main frame of the program consists of two panels. In the menu panel on the left, the user can make choices for drawing input trees in a step-by-step fashion. The right panel has two canvas areas for displaying the game trees (Figure 4).

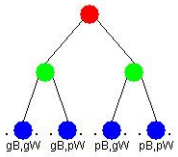
3.1 Drawing a tree using NEGEXT

For negotiation problems that can be represented as extensive form games, one can draw a tree using NEGEXT. The user draws a tree from the root node to the terminal leaf nodes, one after another. For a current node, he needs to define the current level of the tree, the parent node, and the player whose turn it is. For the example described in Figure 1, the user may draw the tree as follows:

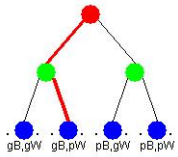
Level 0 for root node and no need to define the parent (as it has none). Select player i on the left panel (i =1 to 10). Players are represented by colors; in this case red represents Biocon.



Level 1 nodes: no need to define the parent (default parent is root node). Select the player i (i =1 to 10) for the left child and right child nodes; in this case both are green, representing Wockhard.



Level 2: terminal nodes with blue color; parent = 1 (level 1, left child) or parent = 2 (level 1, right child). Define the proposition and add the terminal nodes one after another.



Checking strategy for the coalition of player 1 and player 2 and for formula $g_B \wedge p_W$. The red line shows their joint strategy.

3.2 Checking for strategies that achieve goals in NEGEXT

After the user has drawn a tree, NEGEXT can check whether a strategy exists for either one player or a coalition of players by applying Algorithm 1. The

basic idea behind the algorithm for finding strategies for an individual player i is to observe every edge from the root to the terminal nodes, considering a single edge from any node representing this player's turn and all possible edges from the nodes representing other players' turns. If there is no alternate strategy for a different player that prevents the player under consideration from having a winning strategy, then there exists a strategy for this player to achieve the proposition (see **Algorithm 1**).

Algorithm 1 Algorithm1 Finding strategy for Player(s)

```
Input: A single player i or set of players Sp, Formula φ
for all TreeNode do
  if IsTerminalNode(TreeNode) and IsTrue(Formula φ, Node TreeNode)
  then
    if Input of Player == Player i then
      if ParentOf(Node TreeNode) == Player i then
        if IsRoot(Player i) then
          Print Player i has a winning strategy for Formula;
        else
          Print Player i has no winning strategy for Formula;
        end if
      else if (ParentOf(ParentOf(Node TreeNode) )) == Player i then
        Check both left child and right child of ParentOf(Node TreeNode);
        if (Check is OK) then
          Print Player i has a winning strategy for Formula;
        else
          Print Player i has no winning strategy for Formula;
        end if
      end if
    else if Input of Player == set of players Sp then
      while (ParentOf(Node TreeNode) != null) do
        if ParentOf(Node TreeNode) == Player i ϵ Sp then
          Sp = Sp \ {Player i};
          Node = (ParentOf(Node TreeNode));
        else
          Print set of players Sp has no winning strategy for Formula.
        end if
      end while
      if Sp == ∅ then
        Print set of players Sp has a winning strategy for Formula
      end if
    end if
  end if
end for
```

For the user to find a strategy to achieve a particular formula, he needs to input the player's name ("player 1") or the names of the coalition of players ("player 1, player 2") and input the goal formula corresponding to a proposition at a terminal node. Then NEGEXT checks all tree nodes starting from the root

to the terminal nodes: If a proposition of any terminal node t matches the input formula, NEGEXT checks the parent node of t. If the parent node of t is matched with player i and it is also the root node, then player i has an individual strategy for this proposition. For example, in the left part of Figure 3, player 1 (red) has a strategy to achieve $\neg p$. If it is a node at level 1 at which it is another's turn, then the algorithm needs to check whether all its children have p. For example, in the right part of Figure 3, player 1 (red) has a strategy to achieve p. Otherwise, player i has no strategy to achieve this proposition.

To find a group strategy for a formula, NEGEXT also checks all tree nodes: If a proposition of any terminal node t matches the input formula, NEGEXT checks the parent node of t recursively up to the root. If all predecessor nodes of t are matched with all players in the group, then the group has a strategy for the formula, otherwise it does not. In the toolkit, the input tree is shown in the left canvas area, and the output tree with a strategy path (red line) is shown in the right canvas area (Figure 4).

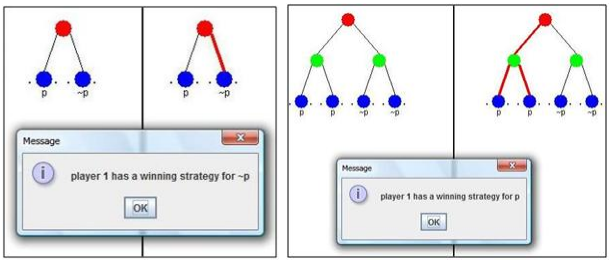


Fig. 3. Strategy checking for an individual player

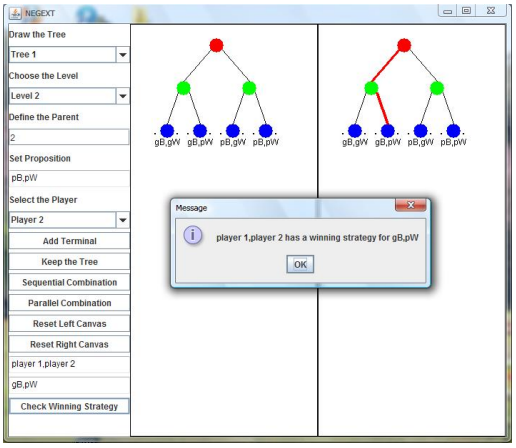


Fig. 4. Strategy checking for a group of players

4 Sequential and parallel combination in NEGEXT

Let us consider how NEGEXT can be used to analyze the stories in Subsections 2.1 and 2.2 which involve combinations of trees. For sequentially combining the trees of Subsection 2.1, NEGEXT first takes as inputs the different trees to combine, one after another, and then gives the sequential combination tree in a separate window (Figure 5). To combine the trees sequentially, NEGEXT first traverses all nodes of tree 1 from the root to the leaf nodes, and then concatenates tree 2 to all leaf nodes of tree 1. Suppose the user asks the system to find one possible strategy for achieving $(g_B \wedge p_W)$ by the coalition $\{B, W\}$, depicted by the red and green players in the screenshots. If company B chooses a genomic expert and company W chooses a proteomic expert, then they can achieve $(g_B \wedge p_W)$, as shown in Figure 5.

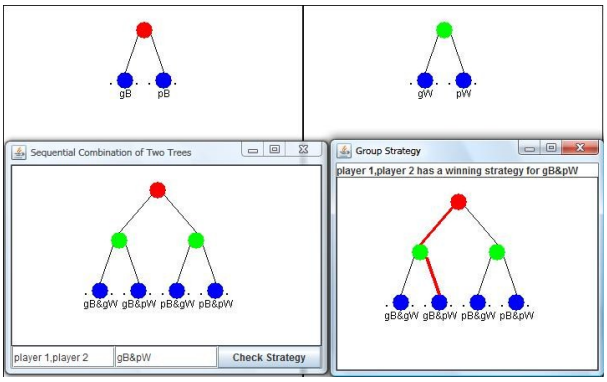


Fig. 5. Sequential combination and strategy checking

Now let us consider the parallel combination of trees, corresponding to the situation described in Section 2.2. As in the previous case, the different trees that are to be combined are taken as separate inputs, one after another, as given in Figure 6. Here, agents QR, SP and FL are represented by pink, red, and green, respectively. Let us define abbreviations as follows: a for $p_{FL} \wedge \neg q_{SP}$; b for $p_{FL} \wedge q_{SP}$; c for $\neg p_{FL} \wedge q_{SP}$; d for $\neg q_{QR} \wedge p_{SP}$; e for $q_{QR} \wedge p_{SP}$; and f for $q_{QR} \wedge \neg p_{SP}$.

Before combining these trees in an interleaving way, we should note here that the parallel combination of two trees will give rise to a bunch of possible trees. These trees will appear in an enumeration, from which the user selects one combination as the final tree. In this case, the particular tree as depicted by the story is given in Figure 6. Figure 7 presents a copy-cat strategy that the common red player (Sprout) can follow in order to end up in a winning situation in the parallel game. Note that formally this strategy is a $\langle QR, SP, FL\rangle$-strategy, which may have been elicited by the user's question as to whether the set $\{QR, SP, FL\}$ can achieve the goal $p_{FL} \wedge p_{SP} \wedge q_{QR} \wedge q_{SP}$.Thus, using NEGEXT enables students to see clearly how players QR, FL, and SP can jointly achieve an intuitive goal.

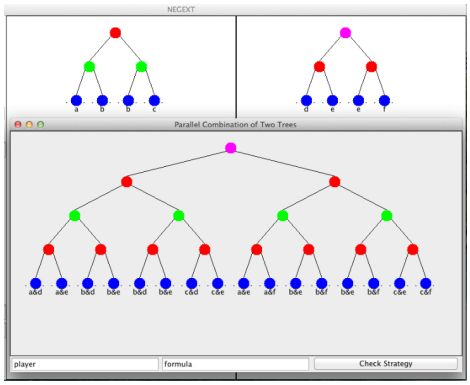


Fig. 6. An interleaving parallel combination

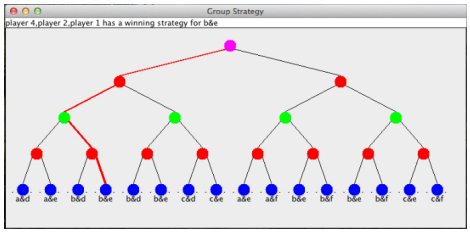


Fig. 7. A group strategy for *QR*, *SP*, and *FL* in the combined tree

5 Conclusions and future work

In this work, we have presented a toolkit to represent negotiations which span over a finite time, and we consider the actions of the negotiators one after another in response to each other. Two examples have been provided to advocate the fact that some real-life negotiations can be aptly described by perfect information extensive form games. The NEGEXT toolkit can help students of negotiation to learn how to respond in order to achieve their goals, including situations where it is not easy to compute the optimal response. The current version of the toolkit has not been tested for learnability and usability yet, so a first step in future research will be to improve it on the basis of a usability study, in which subjects will be asked to use the tool in order to find strategies given particular negotiation trees.

We used a binary tree data structure for drawing trees in the NEGEXT toolkit, implemented on a pure Java applet. In real life, players often have more than two options. In addition, current NEGEXT still allows only at most level 2 trees (root plus intermediate level plus leaves) for sequential or parallel combination, because of the node placing in the current graphical user interface. We

aim to relax both restrictions in future work so that NEGEXT will be able to represent various types of branching at different nodes, as well as deeper trees.

In parallel combination, NEGEXT presents all possible parallel combinations to the user. NEGEXT allows the user to click the "Parallel Combination" button repeatedly in order to view all parallel combined trees in different interleaving ways. If the user wants to know of only an optimally combined tree, NEGEXT may confuse him. Solving this problem is also left for future work. Note that while defining parallel combination of trees we only considered interleaving moves in between trees. We plan to incorporate simultaneous moves as well, bringing NEGEXT closer to the spirit of concurrent games.

The current version of NEGEXT is restricted to perfect information situations. However, in many real-life negotiations, the information dilemma looms large: Which aspects to make common knowledge and which aspects to keep secret or to divulge to only a select subset of co-players? For example, Raiffa distinguishes negotiation styles with "full or partial open truthful exchange" [1]. Such aspects of imperfect or incomplete information cause far-reaching asymmetries between parties, sometimes with grave consequences [10]. It will be future work to extend NEGEXT so that incomplete, imperfect, and asymmetric information can be incorporated in its tree representations and its strategic advice.

References

1. Raiffa, H., Richardson, J., Metcalfe, D.: Negotiation Analysis: The Science and Art of Collaborative Decision Making. Belknap Press of Harvard Univ. Press, Cambridge (MA) (2002)
2. Kuhn, H.: Game theory and models of negotiation. The Journal of Conflict Resolution **6**(1) (1962) 1–4
3. Brams, S.J.: Negotiation Games: Applying Game Theory to Bargaining and Arbitration. Routledge, London (2003)
4. Rosenschein, J., Zlotkin, G.: Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers. MIT-Press, Cambridge (MA) (1994)
5. Parsons, S., Wooldridge, M., Jennings, N.: Agents that reason and negotiate by arguing. Journal of Logic and Computation **8** (1998) 261–292
6. Lin, R., Kraus, S., Wilkenfeld, J., Barry, J.: Negotiating with bounded rational agents in environments with incomplete information using an automated agent. Artificial Intelligence Journal **172**(6-7) (2008) 823–851
7. Hindriks, K., Jonker, C., Tykhonov, D.: Towards an open negotiation architecture for heterogeneous agents. In Klusch, M., others, eds.: Cooperative Information Agents XII. Volume 5180 of LNCS. Springer, Berlin (2008) 264–279
8. Osborne, M., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge, MA (1994)
9. Ghosh, S., Ramanujam, R., Simon, S.E.: Playing extensive form games in parallel. In Dix, J., others, eds.: Proceedings 11th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA XI). Volume 6245 of LNCS., Berlin, Springer (2010) 153–170
10. Oakman, J.: The Camp David Accords: A case study on international negotiation. Technical report, Princeton University, Woodrow Wilson School of Public and International Affairs (2002)