UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

NEW VARIANTS OF THE MICRO NEGOTIATION STRATEGY

DAVID AGUILERA LUZON

Thesis supervisor
FRANCISCO JAVIER LARROSA BONDIA (Department of Computer Science)

Thesis co-supervisor
DAVE DE JONGE

Degree
Master's Degree in Informatics Engineering

Master's thesis

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

30/06/2025

CSIC
Consejo Superior de Investigaciones Científicas

IIIA
Institut d'Investigació en Intel·ligència Artificial

Abstract

Automated negotiation plays a vital role in enabling autonomous agents to reach agreements in distributed and dynamic environments, especially in multi-agent systems (MAS) where multiple parties with potentially conflicting interests must coordinate. While extensive research has focused on bilateral negotiation, multilateral negotiation, where more than two agents negotiate simultaneously, remains a significantly more complex and underexplored domain.

This thesis addresses the challenge of developing effective negotiation strategies for multilateral settings by extending and evaluating **MiCRO**, a minimal, model-free negotiation strategy originally designed for bilateral scenarios. The key motivation is to examine whether a strategy as simple as MiCRO can remain competitive in multilateral domains and to assess whether existing benchmark frameworks such as those used in the *Automated Negotiating Agents Competition (ANAC)* provide sufficient complexity to evaluate negotiation intelligence fairly.

We introduced three variants of MiCRO—*Minimum*, *Mean*, and *Maximum*—each representing a distinct approach to concession timing in response to other agents' behavior. We tested these variants against top-performing agents from past ANAC competitions using the ANAC2015 domain set. The experiments measured mean utility, utility on agreement, and agreement rate across fixed and randomized preference profiles. Additionally, we conducted a game-theoretic best-response analysis to evaluate each strategy's stability in strategic environments.

The results show that all MiCRO variants are highly competitive, with the **Minimum version emerging as the most robust and manipulation-resistant**. Unlike the other variants, which adversarial agents can exploit through strategic offer flooding, MiCRO-Minimum maintains stable and favorable performance even in adversarial contexts. Moreover, its presence in multiple equilibrium scenarios under best-response dynamics highlights its theoretical solvency.

These findings suggest that MiCRO not only generalizes effectively to multilateral negotiation but also exposes critical weaknesses in existing benchmarks like ANAC, where even non-adaptive strategies can outperform complex learning-based agents. As such, this work reinforces the importance of revisiting evaluation standards and presents MiCRO-Minimum as a reliable baseline for future negotiation research.

Index Terms—Automated Negotiation, Multi-Agent Systems, Multilateral Negotiation, Concession Strategies, MiCRO Strategy, Negotiation Benchmarking, Best-Response Dynamics, ANAC, Opponent Modeling, Strategy Robustness, Nash Equilibrium.

Contents

List of Figures

List of Tables

Listings

1 Introduction

Negotiation is a fundamental mechanism for resolving conflicts and reaching agreements, a cornerstone of human interaction since the dawn of time. It underpins decision-making processes in diverse domains such as economics, political science, and social interactions. We find examples of negotiations in the most mundane daily activities, such as two friends deciding where to go for dinner, to more critical decisions like allocating resources for a new server within an organization, or even decisions with profound impact on our society, such as determining international trade tariffs.

While some negotiations can be straightforward, their complexity can escalate rapidly with an increase in the number of issues to be agreed upon, deteriorating relationships between parties, or the involvement of multiple parties, each with potentially divergent interests. In every negotiation scenario, each involved party possesses clear preferences and defined objectives they aim to satisfy, alongside other parameters and options where they might be willing to concede to achieve a consensus.

In computer science, particularly within the field of multi-agent systems (MAS), negotiation serves a similar vital purpose: enabling autonomous software agents to reach agreements in distributed and dynamic environments. Multi-agent systems are composed of multiple interacting agents—software entities capable of perceiving their environment, making decisions, and executing actions autonomously. These agents can be designed to collaborate, compete, or coexist while pursuing individual or collective goals. MAS are particularly useful in solving problems that are too large, complex, or decentralized for a single agent or centralized controller to handle effectively[1], [2].

In such systems, each agent typically operates with partial information, limited capabilities, and distinct preferences or objectives, which may conflict with those of other agents. To function effectively in these environments, agents must communicate, coordinate, and often negotiate: whether to share resources, assign tasks, resolve conflicts, or agree on joint plans. This is especially relevant in domains such as distributed robotics, smart grids, collaborative filtering, traffic management, and autonomous e-commerce, where decision-making responsibilities are delegated to intelligent agents acting independently across the system.

As computational systems become increasingly decentralized and agents are designed to act on behalf of users, organizations, or even other systems, the need for automated negotiation becomes ever more critical. Automated negotiation refers to the design and implementation of autonomous agents capable of engaging in negotiation without direct human intervention. These agents must make strategic decisions to maximize their own outcomes while taking into account the preferences, constraints, and likely behavior of other agents, in order to ensure that mutually acceptable agreements are reached[1]–[4].

1.1 Motivation for Automated Negotiation

The drive towards automated negotiation stems from its potential to address challenges in a multitude of real-world applications that require autonomous decision-making under constraints and uncertainty. The idea of automating negotiations arises from the necessity to manage scenarios of increasing complexity, where the sheer volume of variables, the required speed of interaction, or the frequency of negotiations might exceed human capabilities or could benefit from the consistency and computational rationality of an artificial agent. Furthermore, as noted by Renting, Hoos, and Jonker [5] in the context of multi-agent meeting scheduling, autonomous agents can enhance the privacy of participants and reduce the human effort involved.

Specific examples illustrating the utility of automated negotiation include[4]:

- In **e-commerce**, agents can negotiate prices, delivery times, or product bundles on behalf of customers or sellers, streamlining transactions and potentially discovering more optimal deals.
- In **cloud computing**, automated agents can negotiate the allocation of resources such as CPU time, storage, or bandwidth among competing applications, ensuring efficient and fair usage of shared infrastructure.

- In **smart grids**, negotiation enables households, businesses, and utility providers to dynamically balance energy production and consumption, contributing to grid stability and efficient energy distribution.
- In **multi-robot systems**, agents controlling robots negotiate task allocation, an ordering of tasks, or shared access to limited physical space or equipment, facilitating coordination and preventing conflicts in complex operations.

In each of these cases, automated negotiation helps address a core computational challenge: how to coordinate autonomous entities that possess limited information and potentially conflicting preferences, in a manner that leads to acceptable, efficient, and timely decisions. Without effective negotiation mechanisms, such systems can suffer from inefficiencies, deadlocks, or suboptimal resource utilization.

1.2 Fundamentals of Automated Negotiation

At its core, automated negotiation involves several key elements. To illustrate, consider two friends, Bob and Alice, deciding on dinner plans. Bob is known for his firmness (rarely concedes), while Alice tends to concede gradually. Bob prefers pizza in Barcelona, early; Alice prefers sushi in Badalona, late. In this scenario:

- **Agents**: Bob and Alice (or the software representing them).
- **Negotiation Domain**: The subject of the negotiation, here "dinner."
- **Issues (Negotiation Variables)**: The aspects to be agreed upon, e.g., location (Barcelona, Badalona), time (early, late), type of restaurant (pizza, sushi).
- **Preference Profile (or Utility Function)**: This describes each agent's preferences over all possible values of the issues. It is individual to each agent and quantifies the utility or satisfaction they derive from each potential agreement. Alice would get a lot of satisfaction (utility) if the dinner takes place in Badalona, late and on a sushi restaurant, while, with this outcome, Bob would have very little utility.

The negotiation process typically unfolds through an exchange of offers and counter-offers. For instance, in the dinner example:

- **Round 1**: Bob proposes dinner at 7 PM in Barcelona at a pizza place.
- **Round 2**: Alice counters with dinner at 9 PM in Badalona at a sushi place.
- **Round 3**: Bob compromises on time and suggests 8 PM in Barcelona at a pizza place.
- **Round 4**: Alice agrees to 8 PM but suggests Badalona and sushi.
- **Round 5**: They reach an agreement: 8 PM in Badalona, but at a fusion restaurant that serves both pizza and sushi.

This exchange exemplifies how agents iteratively adjust their offers, guided by their preferences, to arrive at a mutually acceptable outcome. Both agents understand that failure to agree results in no dinner (zero utility for both of them). Thus, negotiation inherently involves **concessions** (strategic compromises) made to improve the likelihood of reaching an agreement[2], [4].

A common paradigm for these interactions is **Proposal-Based Negotiation (PBN)**, where agents iteratively exchange offers or decide to terminate if no agreement is reached [1]. Within PBN, standard protocols include the **Alternating-Offers Protocol (AOP)** for bilateral (two-party) negotiations, where agents take turns making offers. For multilateral negotiations involving more than two agents, an extension of the bilateral alternating offers protocol known as the **Stacked Alternating Offers Protocol (SAOP)** is commonly used [1]. SAOP is designed to generalize turn-based negotiation to settings with three or more agents. In this protocol, agents take turns in a predefined order, where during each round, one agent can either propose a new offer or accept the most recent offer on the table. An agreement is reached if all agents accept the same offer;

otherwise, the negotiation continues to the next agent in the turn order. SAOP preserves the sequential structure of bilateral negotiation while enabling multilateral interaction[2].

The negotiation setting consists of the negotiation protocol, the negotiating agents, and the negotiation scenario.[6] The negotiation protocol defines the rules of encounter to which the negotiating agents have to adhere. The negotiation scenario takes place in a negotiation domain, which specifies all possible outcomes (the so-called outcome space). The negotiation agents have a preference profile, which expresses the preference relations between the possible outcomes. Together, this defines the negotiation scenario that takes place between the agents. The negotiation scenario and protocol specify the possible actions an agent can perform, given the negotiation state [6]. See figure 1

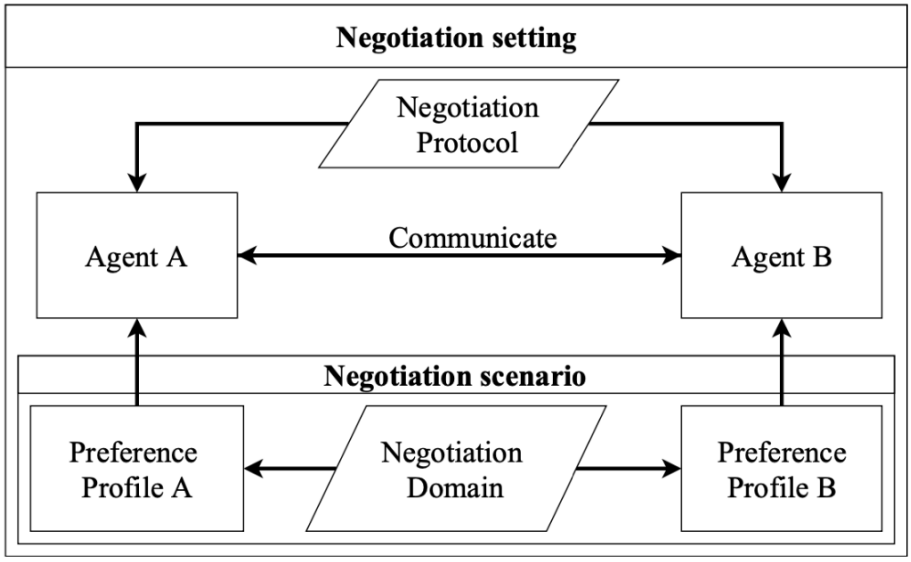

Figure 1: An illustration of the elements defining an automated bilateral negotiation. Reproduced from [6].

The core dynamic of automated negotiation involves agents, while acting in their own self-interest to maximize individual gains, inherently needing to collaborate to achieve mutually beneficial outcomes. A central challenge is striking an effective balance: maximizing one's own utility while ensuring proposed terms offer adequate benefits to others, thereby increasing acceptance likelihood. Consequently, a primary question in automated negotiation research is how to effectively make this trade-off, how to identify and implement an optimal concession strategy[4].

1.3 State of the Art in Negotiation Strategies

An agent's **negotiation strategy** dictates its behavior: how it formulates offers, responds to opponent proposals, when and by how much it concedes, and when it accepts an agreement or withdraws. The specific mechanics of concession define the strategy.[2]

It is important to note that strategies are not inherently tied to a specific agent. An agent is a digital representation of an entity or person, and that entity can assign different strategies to the agent depending on their goals or preferences. Therefore, it is common to find different agents employing the same strategy, the same agent using different strategies across situations, or even strategies being developed and analyzed independently of any particular agent. This separation allows for greater flexibility in agent design, experimentation, and adaptation to diverse negotiation contexts.

For example, consider two agents: Agent A represents a retail company, while Agent B represents a logistics provider. Both may use the same time-dependent concession strategy where offers become more generous as the deadline approaches because it aligns with their current negotiation objectives. On the other hand, the retail company might choose to assign a more competitive strategy to Agent A in a future negotiation involving a high-stakes supplier. Similarly, researchers

can design and test a new negotiation strategy in isolation, without initially linking it to any specific agent, to evaluate its performance under different simulated conditions. This illustrates that an agent is the "who" in a negotiation, while the strategy is the "how." [2], [4]

To understand the broader landscape of negotiation strategies, it is helpful to briefly consider more standard approaches, such as time-based and adaptive strategies:

- **Time-based strategies** are the simplest kind of negotiation strategy. A time-based strategy makes use of a function $\lambda : \mathbb{R} \to \mathbb{R}$, known as the aspiration function, which would typically be strictly decreasing. This aspiration function controls the amount of concession the agent makes as a function of time. Specifically, the idea is that at any given time t, our agent a_1 will propose an offer ω that concedes as much as possible, under the constraint that his utility value $u_1(\omega)$ must remain greater than or equal to $\lambda(t)$. Time-based agents can be either hardheaded or conceding, depending on the shape of the aspiration function. The faster λ decreases, the more conceding the agent will be. [2]. Early and simpler time-dependent strategies include[2]:
  - **Linear strategies**: The agent concedes value at a steady rate over time.
  - **Conceder strategies**: These make significant concessions early on and then maintain a relatively fixed position.
  - **Boulware strategies**: The agent maintains a firm stance, making minimal concessions until close to a negotiation deadline, then concedes rapidly if no agreement has been reached.
- **Adaptive strategies** represent a more sophisticated class of methods designed to adjust an agent's behavior based on the opponent's actions. Their effectiveness lies in the use of **opponent modeling**, where the agent infers the preferences, utility function, or behavioral patterns of the other party by analyzing received offers or observed concessions during the negotiation. This allows the agent to dynamically estimate the opponent's acceptance threshold and future concession behavior[2].

  Once such a model is in place, the adaptive agent sets its own *target value* as the offer that maximizes its own utility while still being likely to be accepted by the opponent. Unlike time-based strategies that follow a fixed aspiration function, adaptive strategies revise their aspiration levels in response to new observations, making their behavior more responsive and context-aware [1], [2].

  Effective opponent modeling provides strategic advantages: it can increase the likelihood of agreement, reduce negotiation duration, avoid exploitation, and help identify mutually beneficial outcomes[1]. Approaches to opponent modeling include:
  - **Bayesian learning**, which maintains probabilistic beliefs about the opponent's issue weights and value rankings, updating them with each observed bid [7].
  - **Gaussian Process regression**, used to model and predict opponent concessions and acceptance behavior, helping optimize the timing and content of offers [8].

  The design of adaptive strategies thus centers on how to learn reliable opponent models in real-time and use them to drive concession behavior. As a result, a central research challenge is to develop opponent modeling algorithms that are both accurate and computationally efficient [1].

Multilateral Negotiation Settings

While many strategies and protocols were initially developed for **bilateral negotiation** involving only two agents there is growing attention on **multilateral negotiation**, where three or more agents interact in a shared environment. Multilateral settings are inherently more complex, as agents must now consider not only their own preferences and their direct opponents' strategies, but also the indirect effects of other agents' decisions on the outcome space[4].

Common real-world examples include resource allocation among multiple stakeholders, multi-party contract negotiations, or joint scheduling among distributed teams. These settings require

agents to reason about coalition formation, fairness across multiple participants, and strategies that scale under increased uncertainty[4].

To support such environments, extensions to existing bilateral protocols have been proposed. For example, the **Stacked Alternating Offers Protocol (SAOP)** a multilateral extension of the Alternating Offers Protocol enables agents to propose or accept offers in a shared turn-based structure [9]. Despite the availability of such protocols, the majority of automated negotiation research still concentrates on bilateral interactions, and only a limited number of strategies have been explicitly tested or optimized for multilateral settings[1].

Table 1: Comparison of Bilateral and Multilateral Negotiation Protocols

Aspect	**Bilateral Negotiation**	**Multilateral Negotiation**
Number of agents	2 agents	3 or more agents
Typical Protocol	Alternating Offers Protocol (AOP)	Stacked Alternating Offers Protocol (SAOP)
Strategic complexity	Lower: only need to model a single opponent	Higher: must consider multiple agents and possible coalitions
Example use cases	Buyer–seller negotiation, contract bargaining	Multi-party scheduling, resource allocation, policy drafting
Support in ANAC	Core track since inception (2010) [9]	Included in later editions via SAOP domains [10]
Research focus	Majority of research and strategies to date [1]	Growing interest; fewer well-tested strategies available

1.3.1 Benchmarking Through ANAC

To foster research and benchmark different negotiation strategies, the **International Automated Negotiating Agents Competition (ANAC)** was established. Since its inception in 2010 [9], ANAC has provided a standardized platform built upon the GENIUS environment that enables researchers to test and evaluate automated agents in a controlled and reproducible setting. The primary motivation behind ANAC is to address the challenges of designing proficient automated negotiators, particularly in *bilateral multi-issue closed negotiations*, where agents must deal with incomplete information, strategic opponent behavior, and real-time decision constraints.

Benchmark Problems. The problems used as benchmarks in ANAC are called *negotiation domains*[9]. A domain is defined by:

- A set of **issues**, each with different discrete possible values.
- A set of **preference profiles**, one per agent, which determine how desirable each possible agreement is to the agent. These preferences are typically represented as utility functions that remain *private and fixed* during negotiation.

The combination of issues and preferences creates a large space of possible offers. These domains can vary significantly in size, structure, and conflict level (for example how opposed the preferences of the agents are). The benchmark problems in ANAC are hand-crafted or based on real-world-inspired scenarios to test negotiation behavior under different levels of complexity[9].

As discussed by Baarslag, Hindriks, Jonker, *et al.* [9], the domains include realistic challenges such as asymmetric utility distributions, varied issue counts, and differing degrees of conflict. However,Jonge [10] points out that many of these domains, especially from earlier ANAC editions, are simple enough that even very basic strategies can perform well without any form of learning or opponent modeling.

Tournament Setup. Each edition of ANAC follows a standardized tournament structure. Agents submitted by participants are matched in round-robin style games, where every agent negotiates multiple times against each other agent across a diverse set of domains[9].

In bilateral tracks, for each domain, the two agents are assigned roles **randomly** meaning they got one random set of preferences and one becomes the initiator (who starts the negotiation) and the other the responder. This randomness is used to ensure fairness and eliminate role-based bias. Each pair of agents negotiates several rounds per domain, and roles are alternated across sessions to further reduce asymmetry[9].

A single negotiation session is carried out under a specific protocol typically the Alternating Offers Protocol (AOP) within a given domain and with fixed deadlines. If an agreement is reached before the deadline, the session ends and both agents receive utility scores based on the outcome. If no agreement is reached, both agents receive their reservation value if there's one, otherwise they receive zero utility[9].

The performance of each agent in ANAC tournaments is then evaluated across multiple dimensions:

- **Average utility**: Mean utility achieved by the agent over all sessions.
- **Agreement rate**: Fraction of sessions in which an agreement was reached.
- **Social welfare**: Sum of the utilities of both agents.
- **Fairness metrics**: Such as distance to the Nash or Kalai–Smorodinsky solution.[11]

Different rankings can be made based on this parameters depending on what metric it is valued the most, even weighted combination of these performance indicators can be used ensuring that agents are rewarded not only for selfish behavior but also for achieving socially desirable and efficient outcomes. ANAC used to grade the agents their Average Individual Utility[9].

While early editions of ANAC focused exclusively on bilateral negotiation, later editions have introduced multilateral tracks, most notably using the **Stacked Alternating Offers Protocol (SAOP)** to support more than two agents. These multilateral domains increase the strategic complexity of the tournament, though they remain less explored compared to the bilateral setting.

Overall, ANAC has become a critical driver in advancing the field of automated negotiation, establishing a shared experimental framework that fosters reproducibility, comparability, and continuous methodological progress, and now it is gradually extending into the multilateral space.

1.4 Evaluating Negotiation Success

The perception of a "good" or "effective" strategy is highly context-dependent. For instance, an agent with a high agreement rate but slow negotiation speed might be detrimental in time-sensitive scenarios where agreement value diminishes over time [1]. Conversely, an agent quick to agree but securing low utility might also be suboptimal.

To objectively evaluate and compare strategies, in the literature, authors typically use various different success metrics:[2], [4], [10]

- **Individual utility**: The benefit or value each agent obtains from the final agreement.
- **Social welfare (or Joint utility)**: The combined utility of all participating agents.
- **Pareto efficiency**: Whether the agreement lies on the Pareto frontier or not. The Pareto frontier is the set of outcomes where no agent can be made better off without making another worse off. Non-Pareto optimal solutions imply potential for mutual improvement[12], [13].
- **Fairness**: In automated negotiation, fairness is a crucial evaluation criterion that captures whether the outcome of a negotiation is equitably beneficial to all parties. It is often assessed using formal solution concepts from cooperative game theory, particularly the *Nash Bargaining Solution (NBS)* and the *Kalai-Smorodinsky (KS) solution*[4], [11].

  The **Nash Bargaining Solution** seeks to maximize the product of the agents' utilities above their respective reservation values. Formally, for two agents with utility functions u_1 and u_2 and reservation values r_1 and r_2, the NBS is the outcome $\omega^* \in \Omega$ that maximizes:

$$(u_1(\omega^*) - r_1) \cdot (u_2(\omega^*) - r_2)$$

This solution is Pareto optimal, symmetric, and independent of affine transformations of utility making it a widely accepted notion of fairness. It captures the idea that both agents should gain as much as possible from cooperation, in a way that balances their respective benefits Vente, Kimmig, Preece, *et al.* [1].

Closely related is the concept of a **Nash equilibrium**, which, although originally from non-cooperative game theory, is also used to evaluate negotiation outcomes. In a Nash equilibrium, no agent can unilaterally change its strategy to improve its outcome. While the equilibrium itself does not guarantee fairness, when the agents' strategies result in an equilibrium that aligns with or approximates the NBS, it is often seen as fair. In many evaluations, like ANAC tournaments, identifying whether an agent forms part of a (pure or mixed) Nash equilibrium is one of the primary ways to assess its strategic robustness and fairness under competition[9], [10].

The **Kalai-Smorodinsky solution**, on the other hand, is the maximal point which maintains the ratios of gains. That is, it is a point μ on the Pareto frontier of F, such that:

$$\frac{\mu_1 - d_1}{\mu_2 - d_2} = \frac{Best_1(F) - d_1}{Best_2(F) - d_2}$$

This concept prioritizes equity in relative satisfaction rather than product maximization.[11]

Together, these metrics provide distinct but complementary perspectives on fairness: the NBS emphasizes joint benefit, while the KS solution emphasizes proportional equality. In practice, the Nash Bargaining Solution is often used as the principal benchmark for fairness in evaluating automated negotiation outcomes[9].

- **Robustness**: The agent's ability to perform well against a variety of opponents and resist manipulation or exploitation.
- **Efficiency**: The time or number of messages needed to reach an agreement.

Different strategies aim to maximize one or more of these metrics, often involving trade-offs among them [1].

1.5 Challenges and Ethical Considerations

The increasing sophistication and autonomy of negotiating agents also spark important social and ethical discussions. As these agents take on more responsibility in domains with real-world impact, several critical questions arise:

- **Fairness**: Can an agent exploit a human or a simpler agent in an unjust manner? Should all agents be required to behave ethically, even in competitive environments?
- **Transparency**: How does an agent make its decisions, and are these decisions comprehensible or explainable to humans, especially in high-stakes scenarios?
- **Accountability**: Who is responsible for a detrimental outcome negotiated by an autonomous agent? The developer, the deploying party, or the system as a whole?
- **Privacy**: Automated agents often process sensitive information like the user preferences or organizational priorities. How should this data be protected and used during negotiation? Agents that actively probe opponents' preferences may inadvertently or deliberately violate privacy boundaries or lead to manipulative behavior if safeguards are not in place. [3]

Addressing these ethical considerations is crucial for the trustworthy adoption of automated negotiation technologies. The more negotiation agents are deployed in real-world settings such as labor disputes, healthcare resource allocation, or international trade the more pressing these concerns become.

Another fundamental challenge is the difficulty of translating human values and trade-offs into formal, computable utility functions. In negotiations involving non-monetary or subjective aspects

such as labor rights, work-life balance, or environmental sustainability quantifying preferences is inherently ambiguous. For instance, in a labor negotiation scenario, how should an agent weigh the value of reducing working hours per day versus increasing vacation days? These are inherently human judgments, and automated agents cannot reliably make them without human guidance. Therefore, preference elicitation must often involve real humans to configure preference profiles that reflect nuanced, context-dependent values.

Furthermore, most existing negotiation strategies are designed to be *greedy* that is, focused primarily on maximizing their own utility without regard for the welfare of the opposing party. While this can be rational in purely competitive settings, it risks producing outcomes that are ethically questionable or socially undesirable. Ideally, agents should be configurable with moral or prosocial values, allowing them to seek favorable outcomes while avoiding excessive harm to their counterparts. This requires research into strategies that balance self-interest with fairness, cooperation, or other normative goals[14].

Incorporating ethical reasoning and human value alignment into automated negotiation is a complex but necessary step toward building agents that are not only effective, but also responsible participants in socio-technical systems.

2 Scope and Objectives of This Thesis

This thesis investigates the development of effective automated negotiation strategies with a particular focus on enhancing agent behavior in dynamic and multi-agent environments. It is within this dynamic landscape of diverse strategies, competitive benchmarking through ANAC, and emerging model-based approaches that the **MiCRO** strategy, a concession-based negotiation algorithm, was initially conceived [10]. MiCRO was designed to demonstrate how a remarkably simple, model-free strategy could outperform highly sophisticated agents in bilateral negotiation domains, particularly those used in past editions of ANAC. However, its design is inherently tied to two-agent interactions, and its applicability to more general negotiation settings remains unexplored.

The core contribution of this work is the **extension of a previously bilateral negotiation strategy MiCRO to multilateral settings**. This thesis addresses the non-trivial challenges of adapting such a strategy to domains involving more than two agents, where negotiation dynamics become significantly more complex due to the increased number of participants, strategic dependencies, and coordination requirements.

This work builds upon MiCRO by exploring its extension and applicability in more complex **multi-agent negotiation scenarios**, which typically involve larger decision spaces, non-trivial coalitional effects, and increased strategic uncertainty. The thesis aims to answer whether MiCRO's simplicity and effectiveness can be retained or must be revised when applied to these more realistic and demanding environments.

A key motivation of this thesis is to assess the *actual challenge* posed by classical ANAC benchmarks, not only in bilateral settings but also when extrapolated to multilateral contexts. While ANAC has contributed significantly to the field by offering a standardized testing framework, de Jonge's analysis suggests that many of its domains (especially those from early competitions) may be too simplistic, allowing even non-adaptive strategies to outperform state-of-the-art agents. This raises the question: can such results still hold in multilateral environments, or does the increased complexity invalidate the effectiveness of MiCRO?

We therefore investigate whether similarly simple strategies can also be successful in multi-agent negotiations, or whether additional complexity such as opponent modeling, group coordination, or adaptive behavior becomes necessary to remain competitive. This investigation forms the backbone of the thesis and guides both the design of experiments and the theoretical analysis.

The main objectives of this thesis are:

- To adapt the MiCRO strategy to multi-agent negotiation settings and evaluate its effectiveness in domains involving more than two agents.
- To investigate whether classical ANAC-style domains still pose a sufficient challenge when extended to multi-agent scenarios, or if they can also be exploited by simple heuristic strategies.

- To compare the performance of simple strategies like MiCRO against established agents in terms of robustness, social welfare, fairness, and efficiency.
- To analyze the theoretical properties of MiCRO and similar strategies in multi-agent domains, including whether they lead to near-optimal or game-theoretically stable outcomes.
- To assess the validity of using linear ANAC domains as benchmarks for evaluating strategy quality, particularly when no learning or opponent modeling is used.

Ultimately, this work aims to determine whether existing benchmark competitions are still suitable for driving progress in the field, and whether strategies like MiCRO can serve as a *baseline* or even as a *challenge* for future negotiation agents in multi-agent environments. In line with de Jonge's conclusion [10], this thesis follows the principle that if negotiation strategies are to be evaluated using traditional ANAC-style domains then those strategies should, at minimum, be compared against MiCRO. This ensures that any observed performance improvements are meaningful and not merely the result of domain simplicity or overfitting to weak baselines.

3 Thesis Outline

The remainder of this thesis is structured as follows:

- Chapter 4: Brief explanation of the main concepts involved in Automated Negotiation.
- Chapter 5: Presents the foundational MiCRO strategy.
- Chapter 6: Introduces the proposed extensions to MiCRO and discusses their implementation details.
- Chapter 7: Presents the experimental setup, results, and a thorough evaluation of the developed strategies.
- Chapter 8: Summary of the findings, final remarks and a Game Theory Analysis of the results.
- Chapter 9: Conclusions from the work done and directions for future research.

4 Formalization of Core Concepts in Automated Negotiation

An automated negotiation session is formally defined as a structured interaction among a set of autonomous agents $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ who exchange proposals within a negotiation domain Ω according to a predefined protocol $\mathcal{P}$ [15]. Each agent's objective is to maximize its own utility function $u_i : \Omega \to \mathbb{R}$ while operating under the constraints imposed by $\mathcal{P}$ and the preferences of other agents[1], [2], [4].

In this section we revisit the fundamental concepts of automated negotiation with increased precision and formalism, especially in the context of the ANAC (Automated Negotiating Agents Competition) framework[9].

Agents and Strategies

An **agent** $a_i \in \mathcal{A}$ is a computational entity that implements a negotiation policy π_i, which governs its observable behavior: when to propose, which offer to generate, when to accept or reject offers, and how to adapt to the negotiation dynamics. Formally, this policy can be represented as a function:

$$\pi_i : \mathcal{H} \to \Omega \cup \{\text{accept}, \text{reject}, \text{end}\}$$

where $\mathcal{H}$ denotes the negotiation history observed up to that point[1].

The **strategy** of an agent encompasses this policy, its opponent modeling (if any), concession tactics, and time or risk management. Strategy classes include [6]:

- *Holding strategy*: maintains proposals near its maximum utility.
- *Linear concession strategy*: reduces utility aspiration linearly over time at a linear rate.
- *Conciliatory strategy*: concedes rapidly.

More advanced strategies incorporate learning and opponent modeling to adjust π_i dynamically.

Negotiation Domain: Ω

The **negotiation domain** or **outcome space** Ω is the Cartesian product of m discrete negotiation **issues**.

Issues are the distinct parameters, variables, or attributes over which the agents must reach an agreement. Each issue is formally defined as a countable set, and the elements within this set are referred to as the possible values for that issue.

$$\Omega = D_1 \times D_2 \times \cdots \times D_m$$

where each D_j is a finite set of possible values for issue j. Every $\omega \in \Omega$ is a complete proposal (also called an *offer*) representing an assignment of values to all issues [15].

For example, in a car-buying scenario:

$$\begin{aligned} D_1 &= \{\text{red}, \text{blue}, \text{white}, \text{black}\} && \text{(color)} \\ D_2 &= \{\text{Audi}, \text{BMW}, \text{Peugeot}, \text{Seat}\} && \text{(brand)} \\ D_3 &= \{\text{less than 10k}, \text{10k to 30k}, \text{more than 30k}\} && \text{(price)} \\ D_4 &= \{\text{electric}, \text{gasoline}, \text{diesel}\} && \text{(powertrain)} \end{aligned}$$

yielding $|\Omega| = |D_1| \cdot |D_2| \cdot |D_3| \cdot |D_4|$.

Utility Function

This is a mathematical function, typically denoted as $\mathrm{u} : \Omega \to R$, which assigns a specific numerical value (utility) to each possible outcome (ω) in the negotiation domain (Ω). This numerical value quantifies how much an agent desires that particular outcome. The utility function effectively operationalizes an agent's preferences allowing for quantitative comparison between different potential agreements. A higher utility value indicates a more preferred outcome for the agent. Every utility function induces a preferenceprofile, which is a ranking of all possible outcomes from most to least preferred. For each unique combination of values assigned to the issues, the utility function returns a corresponding utility score. A common simplification used in many negotiation models is the assumption of linear additive utility functions [1] Each agent a_i has a utility function $u_i : \Omega \to [0, 1]$ that encodes its quantitative preferences over offers. A common simplification assumes a linear additive form [16]:

$$u_i(\omega) = \sum_{j=1}^{m} w_j^i \cdot e_j^i(\omega_j)$$

where:

- $w_j^i \in [0, 1]$ is the normalized weight agent i assigns to issue j ($\sum w_j^i = 1$),
- $e_j^i : D_j \to [0, 1]$ is the evaluation function for the value ω_j.

In more complex domains, utility functions may include interdependencies between issues, requiring non-separable models.

Agreement and Optimality

A proposal $\omega^* \in \Omega$ becomes an **agreement** if it is explicitly accepted by all agents or accepted according to the termination rules of the protocol $\mathcal{P}$.

Ideally, an agreement should be Pareto optimal. If an agreement is not Pareto optimal, it implies that there was potential for a "better" deal for at least one party without disadvantaging others, indicating some inefficiency in the negotiation process or outcome [12], [13]. The set of all Pareto optimal agreements defines the **Pareto frontier**.

Reservation Value

The **reservation value** (or **reservation utility**) r_i of agent a_i represents the utility the utility the agent receives when he does not make an agreement. [1], [2].

Negotiation Protocols $\mathcal{P}$

The **negotiation protocol** defines the rules governing interaction among agents. A protocol $\mathcal{P}$ specifies:

- The allowed message types: offers, acceptances, rejections...
- The message ordering or turn-taking mechanism: sequential, round-robin...
- The termination conditions: agreement reached, time expired, maximum rounds...

The most common protocols include:

- **Alternating Offers Protocol (AOP)**: sequential bilateral negotiation.
- **Stacked Alternating Offers Protocol (SAOP)**: for multilateral negotiation involving more than two agents [17].
- **Argumentation-Based Negotiation (ABN)**: supports richer interactions via arguments and justifications[18], [19].

4.1 ANAC Scenario Configuration

ANAC scenarios are defined by the triple $(\Omega, \{u_i\}, \mathcal{P})$, where:

- Ω and each u_i are unknown to the agents prior to the negotiation and are loaded at runtime,
- $\mathcal{P}$ is typically SAOP,
- agents are evaluated through multi-instance tournaments, involving several scenarios and multiple repetitions,
- performance metrics include average utility, agreement rate, social efficiency (like distance to Pareto frontier), and fairness (like the Nash product).

This configuration is described in detail across various competition overview papers [9], [16].

5 The MiCRO Benchmark Strategy

In 2022 the MiCRO (Minimal Concession in Reply to new Offers) strategy was introduced by Dr. Dave de Jonge as a critique of the perceived simplicity of linear bilateral domains commonly used in the Automated Negotiating Agents Competition (ANAC) and as a new benchmark for future negotiation agent development. The central argument presented is that these traditional ANAC domains, despite being widely used, might be too simplistic to truly differentiate the capabilities of complex negotiation algorithms, particularly those employing sophisticated opponent modeling techniques [10].

5.1 Purpose and Rationale

The primary purpose of developing MiCRO was:

- To demonstrate that the linear bilateral ANAC domains, often used as default benchmarks, could be effectively tackled, and even outperformed, by an extremely simple strategy that deliberately refuses to use complex mechanisms like opponent modeling or machine learning [10].
- To propose MiCRO as a baseline benchmark strategy. The paper argues that any new negotiation algorithm tested on these linear ANAC domains should, at a minimum, be compared against MiCRO to provide a more meaningful assessment of its advancements [10].

Dr. Dave de Jonge insists on that MiCRO is not intended as a strategy for practical applications or for negotiations with humans, but rather serves as an analytical tool to assess the complexity of domains and the relative strength of other negotiation strategies [10].

5.2 How MiCRO Works

MiCRO's design is characterized by its minimalism and its reactive nature, which is tied to the novelty of the opponent's offers.

Simplifying, MiCRO works as follows: if the agent using the MiCRO strategy has not made more unique proposals than its opponent, it makes a new offer. The offers are selected from the list of all possible agreements, which is sorted in descending order of utility for the agent. The agent then proposes the next offer on that list, so it proposed the one with the highest utility that has not yet been proposed. If, on the other hand, the agent has already made more proposals than the opponent, it simply repeats a previous offer selected at random from those it has already made.

This behavior ensures that MiCRO always makes the smallest possible concession in response to a new offer. Importantly, the strategy does not require any knowledge of the opponent's preferences or even the agent's own utility values it only needs a complete preference ordering over the domain's offers[10].

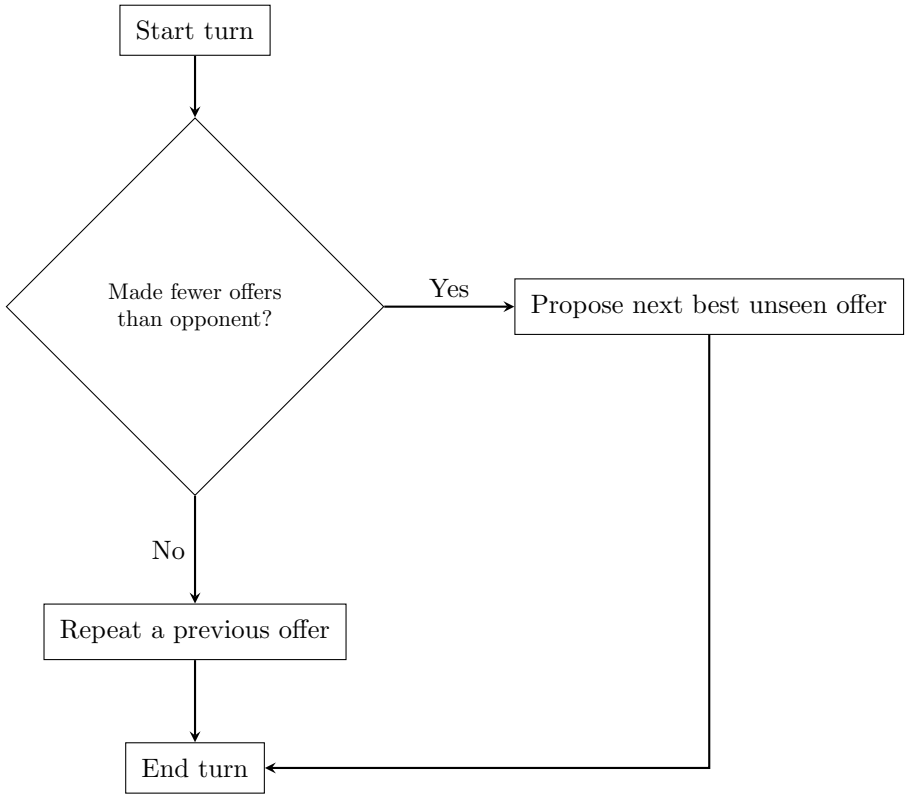

Figure 2: Simplified flowchart of MiCRO strategy behavior on an offering turn.

Concession Strategy

The core mechanism of MiCRO is its concession strategy[10]:

1. **Offer Sorting:** Before negotiation begins, MiCRO creates a list of all possible offers in the domain (Ω), sorted in descending order of utility for itself. Let this sorted list be $(\omega_1, \omega_2, ..., \omega_K)$, where $u_1(\omega_1) \geq u_1(\omega_2) \geq \cdots \geq u_1(\omega_K)$ and $K = |\Omega|$. Notably, MiCRO does not need to know its actual utility function values; it only requires a complete preference ordering over the offers.
2. **Proposal Mechanism:** When it is MiCRO's turn to make a proposal, its action depends on the history of offers received and made:
   - Let n be the count of *different* offers received from the opponent so far.
   - Let m be the count of *different* offers MiCRO has proposed so far.
   - If $m \leq n$ (meaning the opponent has introduced at least as many new offers as MiCRO), MiCRO proposes the next offer on its sorted list that it has not yet proposed, i.e., ω_{m+1}. This is considered its "minimal concession."
   - If $m > n$ (meaning MiCRO has proposed more unique offers than it has received), MiCRO repeats one of its previously proposed offers by picking a random integer r such that $1 \leq r \leq m$ and proposing ω_r.
3. **Reservation Value Handling:** If proposing ω_{m+1} would result in an offer below MiCRO's reservation value ($u_1(\omega_{m+1}) < rv_1$), it will instead repeat a random previous proposal, even if $m \leq n$.

A key aspect of MiCRO is that it considers any new, distinct offer from the opponent as a form of "concession," regardless of whether that offer provides more or less utility to MiCRO than the opponent's previous offers. The rationale is that the opponent's utility function is unknown, so an offer that seems worse to MiCRO might still have been a concession from the opponent's perspective. MiCRO itself always makes the smallest possible concession by moving to the very next offer in its preference-sorted list [10].

Acceptance Strategy

The paper describes a simple acceptance strategy for MiCRO[10]: Let ω_{low} be the lowest utility offer MiCRO is willing to propose at that time (defined as ω_{m+1} if $m \leq n$, and ω_m if $m > n$). MiCRO will accept a received offer ω if and only if $u_1(\omega) \geq \max\{u_1(\omega_{low}), rv_1\}$.

5.3 Results and Theoretical Analysis

MiCRO's performance was evaluated empirically by running tournaments against top-performing agents from several past ANAC editions (2012, 2013, 2018, and 2019) using the Genius negotiation platform [10]. Performance was measured using average utility scores obtained by agents against each opponent and across all domains. Both tournament evaluation (overall average utility) and game-theoretical evaluation (identifying Nash equilibrium) were considered.

5.3.1 Empirical Performance

The experimental results consistently showed MiCRO performing very strongly:

- MiCRO outperformed the top agents of the ANAC years tested achieving the highest tournament score both with and without self-play. And also, on most of the settings MiCRO vs. MiCRO formed the only pure Nash equilibrium.
- MiCRO also outperformed standard Boulware strategy implementations.
- **Speed:** Despite making minimal concessions, MiCRO vs. MiCRO sessions were generally much faster than those involving other agents, attributed to the lack of time spent on opponent modeling. Even in very large domains, MiCRO vs. MiCRO always reached agreement quickly.

5.4 Conclusions from the MiCRO Paper

The MiCRO paper by Jonge [10] presents a critical evaluation of the benchmark domains used in earlier editions of the ANAC competition. The key conclusions drawn in the paper can be summarized as follows:

1. **Oversimplicity of Linear ANAC Domains:** The paper provides both empirical and theoretical evidence that many of the classic ANAC negotiation domains are too simple. MiCRO, a strategy that does not use opponent modeling, machine learning, or even explicit utility computations (only ordinal preference rankings), consistently outperforms or matches top-scoring ANAC participants. This reveals that such domains are solvable even without advanced strategic reasoning.

2. **Need for Better Benchmarking:** Given MiCRO's strong results against top agents, including the winners of past ANAC editions, Jonge [10] argue that these domains are inadequate for properly evaluating sophisticated negotiation strategies. New algorithms should no longer be validated solely on linear ANAC domains. Instead, evaluation should be done in more challenging environments, such as:

    - Non-linear utility domains.
    - Domains with interdependencies between issues.

3. **MiCRO as a New Baseline:** Any new negotiation algorithm tested on linear ANAC domains should be compared against MiCRO. Since MiCRO has near-optimal behavior in these settings (in many cases aligning with the Nash Bargaining Solution), it serves as a robust and interpretable baseline to assess the true strength of more complex strategies.

4. **MiCRO is Not for Practical Use, But for Benchmarking:** The authors clearly state that MiCRO is not intended for practical deployments or human-facing negotiations. Rather, it is designed as a benchmark strategy to highlight the structural limitations of certain domains and to act as a litmus test for evaluating new negotiation agents.

6 New MiCRO Strategy for Multi-lateral Scenarios

6.1 Motivation

As previously discussed, the MiCRO strategy introduced by Dave de Jonge was originally designed for bilateral negotiation scenarios. This implies that its results and conclusions are applicable exclusively to negotiations involving two agents. Consequently, a direct application of the strategy to multi-lateral settings where more than two agents participate would be inadequate without adaptation.

This thesis is motivated by the need to explore whether the simplicity and effectiveness of MiCRO can be extended to more complex multi-agent environments. In particular, we seek to understand whether the same reasoning that questions the validity of simplistic benchmarks in bilateral settings also applies to multi-agent negotiation. The motivation is therefore aligned: to challenge existing assumptions, identify the limitations of current benchmarks, and assess whether MiCRO can succeed, without adaptation in it's core logical behavior, in Multi-lateral negotiation contexts.

6.2 Objective

The goal of this work is to extend the existing MiCRO strategy to accommodate multi-lateral negotiation scenarios. This involves leveraging the existing negotiation protocol while modifying the decision-making process to consider all offers made by all participating agents, rather than just one opponent. The aim is to retain the original non-strategic behavior of MiCRO while making it sensitive to the broader multi-agent context.

6.3 Adapting MiCRO to Multi-lateral Settings

To achieve this, the key question becomes: *how should MiCRO determine when to propose a new offer based on the multiple opponents it faces?*

Recall the original MiCRO strategy operates following minimal concessions based on n being the count of *different* offers received from the opponent so far and m being the count of *different* offers MiCRO has proposed so far. Then if $m \leq n$, MiCRO proposes the next offer in its sorted list that it has not yet proposed.

An important note is that the changes to adapt MiCRO to multilateral are only made to the offering strategy, the acceptance method remains the same, accepting if and only if the last offer holds more utility or the same utility as the last offer proposed by the agent using MiCRO.

In a multi-lateral setting, however, determining the appropriate value for n is not straightforward, as there are multiple opponents, each potentially making a different number of offers. To address this, we introduce three new variants of MiCRO, each using a different rule for aggregating the number of offers made by all agents.

The following three options are proposed by Dr. Dave de Jonge, and its expected behavior can be found on the Section 6.6

- **Maximum:** MiCRO uses the number of offers made by the agent who has submitted the most offers. Let n_i denote the number of offers made by agent i. Then, the maximum number of offers is given by $n = \max\{n\_1, n\_2, \ldots, n_k\}$, where k is the total number of opponents.

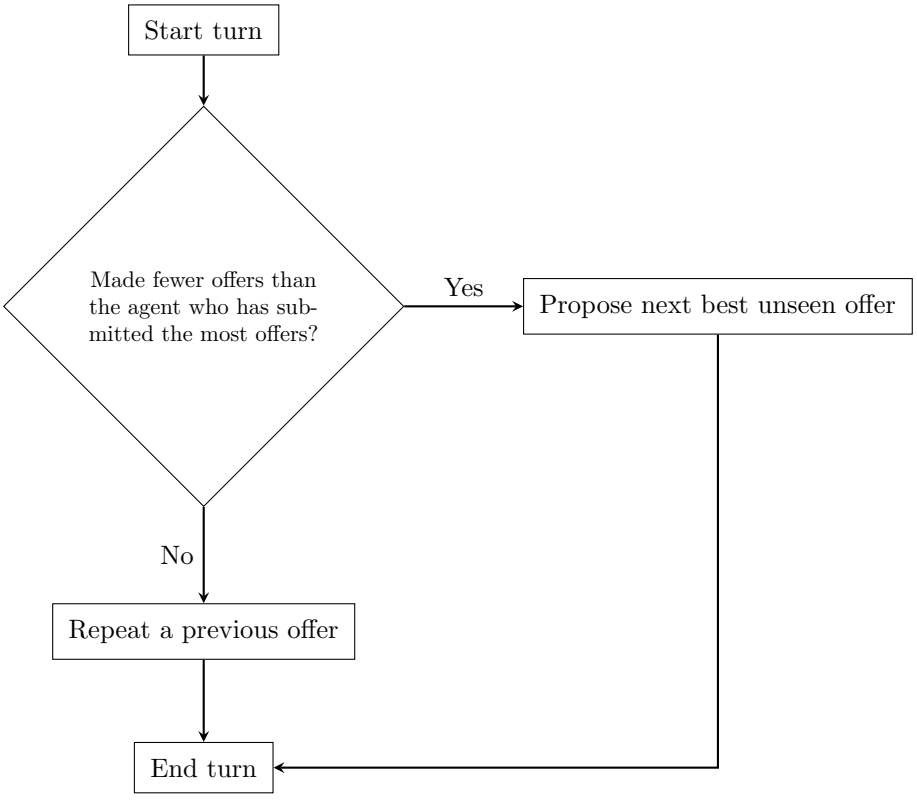

Figure 3: Simplified flowchart of MiCRO-Multi strategy behavior on an offering turn using Maximum Version.

- **Minimum:** MiCRO uses the number of offers made by the agent who has submitted the fewest offers. In this case, $n = \min\{n\_1, n\_2, \ldots, n_k\}$.

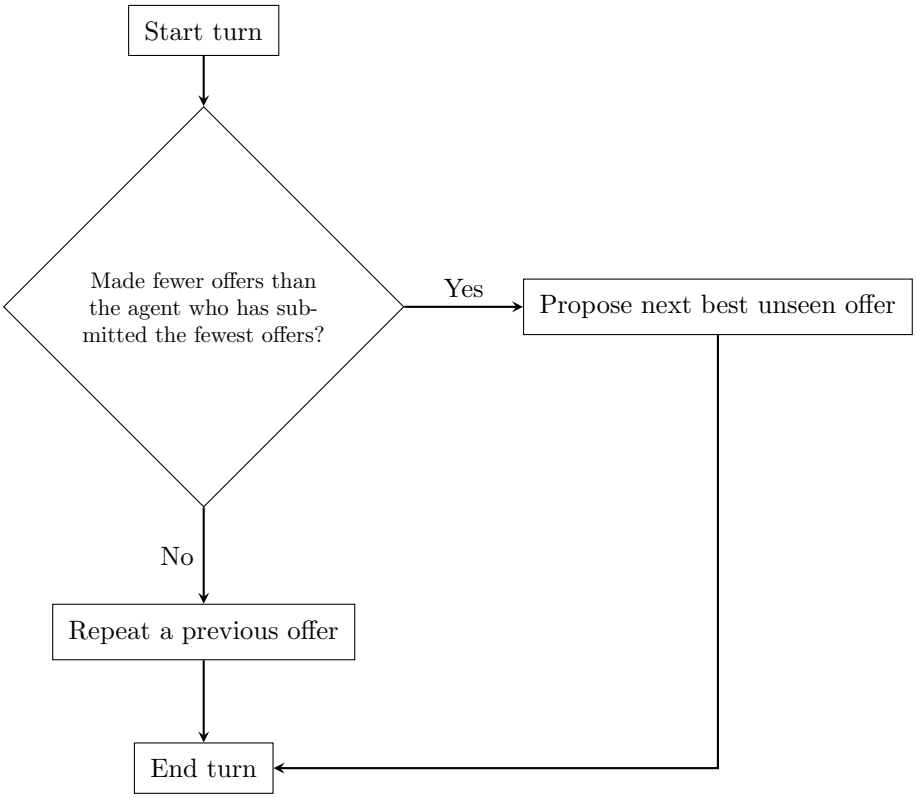

Figure 4: Simplified flowchart of MiCRO-Multi strategy behavior on an offering turn using Minimum Version.

- **Mean:** MiCRO uses the average number of offers made by all agents. That is, $n = \frac{1}{k}\sum_{i=1}^{k} n_i$.

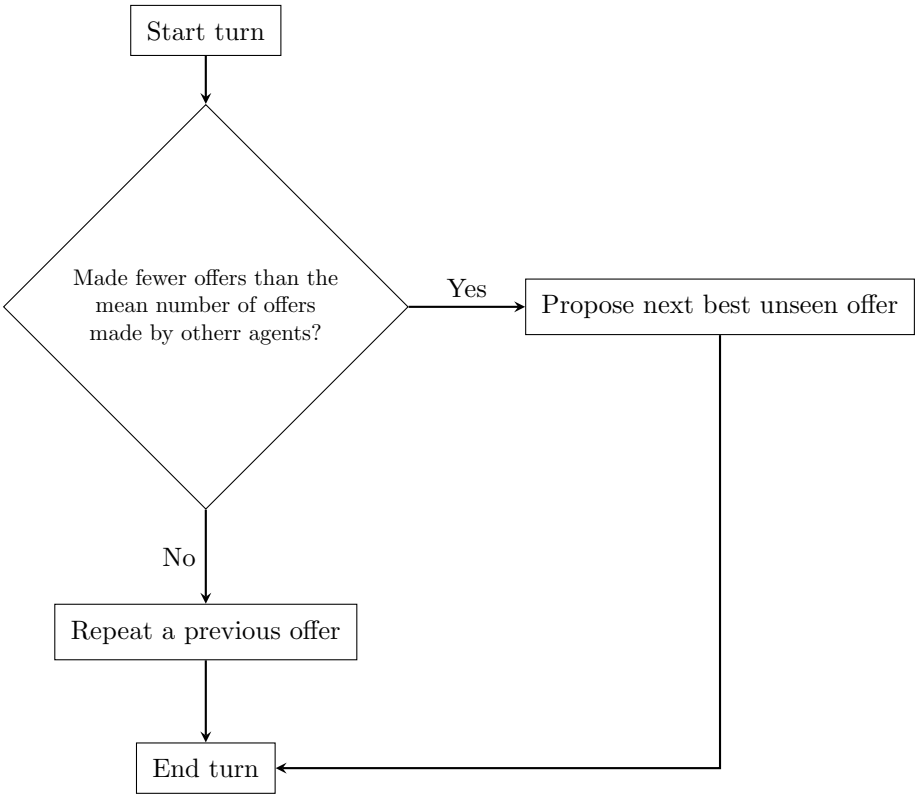

Figure 5: Simplified flowchart of MiCRO-Multi strategy behavior on an offering turn using Mean Version.

These three variants provide different trade-offs between concession levels and responsiveness to opponents' behavior. But all of them still keep the logical behavior of MiCRO, the only difference between MiCRO before and now is how MiCRO consideres if he has to do another offer or not.

6.4 Deadlock in MiCRO-Multi and Proposed Resolution

The extension of MiCRO to multilateral negotiation, based on de Jonge's proposed adaptation of the strategy to multi-agent settings using the thress versions discussed above, initially assumed that each agent would make concessions based solely on the number of offers it had actively proposed. However, during experimentation with the *Minimum* variant of MiCRO in trilateral scenarios, we discovered a previously undocumented deadlock condition, which revealed a critical limitation of this adaptation.

The Deadlock Problem. In scenarios involving three MiCRO agents all using the Minimum strategy, a negotiation could become permanently stuck. The problem arose because each agent tracks its own progress relative to the agent that has made the fewest number of proposals. If an agent has already made more proposals than the current minimum, it refrains from offering anything new and instead repeats one of its prior offers at random.

This behavior becomes problematic in symmetric cases where all agents are waiting for others to act. For example, suppose agents MiCRO 1, 2, and 3 have proposal counts $n_1 < n_2 = n_3$, and it is currently MiCRO 3's turn. Since $n_1 < n_3$, MiCRO 3 will not generate a new offer. Instead, it randomly repeats one of its previous offers, hoping that others will respond. In a case scenario where MiCRO 1 accepts the offers from MiCRO 2 and MiCRO 3 but the other agent are rejecting them (MiCRO 3 rejects MiCRO 2 offer and viceversa), no new proposals are going to be made. This is because MiCRO 1 is accepting so is not going to propose, and the other agents will not propose as they already made more offers than MiCRO 1. Therefore the negotiation enters a deadlock.

The Resolution. To solve this, we modified the internal counting logic of MiCRO so that

acceptances are also treated as contributing to an agent's activity. Specifically, let n_i represent the number of offers *proposed or accepted* by agent i. Previously, only proposals counted. Now, when an agent accepts a new offer that it had not seen or proposed before, that acceptance is treated as equivalent to a proposal in terms of progress tracking. This prevents the agent from being indefinitely blocked due to a lack of new proposals.

Why this works. Consider again the scenario where $n_1 < n_2 = n_3$, and it is MiCRO 3's turn. MiCRO 3 randomly repeats one of its past offers. Eventually, this repeated offer is either:

- Accepted by MiCRO 1, increasing n_1 and breaking the deadlock. Because at some point MiCRO 1 will have accepted all the other agents offrrs and now the scenario will be $n_1 = n_2 = n_3$.
- Rejected by MiCRO 1, which then triggers MiCRO 1 to make a new proposal, also increasing n_1.

In either case, n_1 increases, and the inequality that caused the deadlock is resolved. Over time, this guarantees that negotiation progresses and avoids stalling.

Significance. This discovery shows that the original multi-agent adaptation of MiCRO, while structurally sound, could not reliably function without this fix. By introducing this acceptance-based adjustment, we preserved MiCRO's reactive, minimal design while extending its robustness to multilateral settings. This modification is not just a patch, but a necessary theoretical contribution that ensures correctness and liveness in the MiCRO-Multi strategy.

6.5 Implementation Environment

To implement and evaluate these new versions of MiCRO, we employed the NegMAS. According to its official documentation:

> "NegMAS is a Python library for developing autonomous negotiation agents embedded in simulation environments. The name NegMAS stands for either *NEGotiation Multi-Agent System* or *NEGotiations Managed by Agent Simulations* (your pick). The main goal of NegMAS is to advance the state of the art in situated simultaneous negotiations. Nevertheless, it can-and has—been used in modeling simpler bilateral and multi-lateral negotiations, preference elicitation, etc."

The library provides comprehensive tools and the necessary documentation for the development and testing of new negotiation strategies in both bilateral and multi-lateral contexts. Futhermore, in order to use other agents to test MiCRO against them, usually the Genius Bridge was used, from NegMAS oficial documentation:

> "Genius is a commonly used automated negotiation research platform implemented in Java. It was used to run ANAC and has hundreds of implemented negotiation strategy for the AOP and SAOP negotiation protocols. It supports other protocols but few existing negotiation strategies are available for any protocols other than AOP and SAOP. NegMAS provides a bridge allowing you to run Genius agents and NegMAS negotiators. This bridge does not come bundled with NegMAS"

Using that tool we use other agents previously developed as a benchmark for the MiCRO multi-agent strategy. Running a bridge it is easy as importing the desired agents, for example

```
from negmas.genius.gnegotiators import Atlas3, PonPokoAgent,
    CaduceusDC16, AgreeableAgent2018
```

Listing 1: Example of Genius Import

Then, by running the following two commands in the terminal, the negotiators are ready to use in the experiments.

```
negmas genius-setup
```

Listing 2: Download Genius using NegMAS

```
negmas genius
```

Listing 3: Run Genius integration via NegMAS

6.6 Hypothesis

Based on the previously observed performance of MiCRO in bilateral negotiations, we hypothesize that these new multi-lateral variants will maintain similar strategic advantages. In particular, the **Minimum** variant is expected to yield higher utility, as it adopts the least conceding behavior by progressing more conservatively through its offer list. Conversely, the **Maximum** variant is expected to concede more quickly and therefore may achieve lower utility. The **Mean** variant is anticipated to perform between these two extremes, offering a balanced trade-off between responsiveness and concession. Based on the discoveries by Jonge [10] we expect the new algorithm, no matter the version, to perform as good as other ANAC strategies winners from previous years on ANAC domains. In other words, we expect that it's behavior can be generalized to multilateral negotiation environments.

However, preliminary analysis suggests that the **Maximum** variant may be particularly vulnerable in adversarial or collusive settings. This version behaves highly reactively: it always tracks the agent with the highest number of proposals and advances its own concessions accordingly. In environments where other agents strategically flood the negotiation with frequent offers, either to manipulate or to coordinate among themselves, the Maximum variant may be tricked into accelerating its concessions prematurely. As a result, it can reach agreements quickly but at the cost of utility, settling for outcomes far below its potential optimum. This behavioral weakness, if exploited, could undermine its effectiveness and represents an important point of analysis in evaluating the robustness of MiCRO variants. The same could be applied to undermine **Mean** variant performance, even though it would be slower also behaves highly reactively and therefore the same attack could be made onto an agent using the MiCRO-mean strategy.

6.7 Code

The implementation of the new strategy is as it follows.

First of all, in order to be able to use a negotiator that follows a certain strategy, we need to create a MAPNegotiator, which is a class of negotiator in NegMAS. A negotiator must have an Offering Policy and an Acceptance policy, which defines the way a negotiator makes offers and accepts offer made by others.

```
from __future__ import annotations
from ..components.acceptance import MiCROAcceptancePolicyMulti
from ..components.offering import MiCROOfferingPolicyMulti
from .modular.mapneg import MAPNegotiator

__all__ = ["MiCRONegotiatorMulti"]

class MiCRONegotiatorMulti(MAPNegotiator):
    def __init__(self, *args, accept_same: bool = True, **kwargs):
        kwargs["offering"] = MiCROOfferingPolicyMulti()
        kwargs["acceptance"] = MiCROAcceptancePolicyMulti(kwargs["offering"
            ], accept_same)
        super().__init__(*args, **kwargs)
```

Listing 4: MiCRONegotiatorMulti class definition

Then we need to define the Offering Policy. An Offering Policy in NegMAS is full of functions that are needed by other methods in order to conduct a session properly. The structural version of the Micro Multi offering Policy looks is the following:

```
@define
class MiCROOfferingPolicyMulti(OfferingPolicy):
```

```
    next_indx: int = 0
    sorter: PresortingInverseUtilityFunction | None = field(repr=False,
        default=None)
    _received: dict[str, set[Outcome]] = field(factory=dict)
    _sent: set[Outcome] = field(factory=set)

    def on_preferences_changed(self, changes: list[PreferencesChange]):
        pass

    def sample_sent(self) -> Outcome | None:
        pass

    def ensure_sorter(self):
        pass

    def next_offer(self) -> Outcome | None:
        pass

    def best_offer_so_far(self) -> Outcome | None:
        pass

    def ready_to_concede(self) -> bool:
        pass

    def __call__(self, state: GBState) -> Outcome | None:
        pass

    def on_partner_proposal(
        self, state: GBState, partner_id: str, offer: Outcome
    ) -> None:
        pass

    def on_partner_response(
        self,
        state: GBState,
        partner_id: str,
        outcome: Outcome | None,
        response: ResponseType,
    ) -> None:
        pass

    def add_accepted_offer(self, offer: Outcome) -> None:
        pass
```

Listing 5: `MiCROOfferingPolicyMulti` Class Structure

The code related to the behaviour of the algortihm inside the MicroOfferingPolicy class is the following.

- ready to concede: this function is the one that decides if our agent should make another offer, and thus conceding, or not. This is the part where $m \leq n$ is done. So for each MiCRO version this will be different.
  - Minimum Version:

```
def ready_to_concede(self) -> bool:
    offers_sent = len(self._sent)

    min_received = (
        min(len(offers) for offers in self._received.values()) if
            self._received else 0
    )
```

```
    return offers_sent <= min_received
```

Listing 6: `ready_to_concede` Minimum Version Method

This method compares the number of offers MiCRO has sent to the minimum number received from any partner. If the number sent is less than or equal to this minimum, MiCRO is ready to concede.

– Max Version:

```
def ready_to_concede(self) -> bool:
    offers_sent = len(self._sent)

    max_received = (
        max(len(offers) for offers in self._received.values()) if
            self._received else 0
    )
    return offers_sent <= max_received
```

Listing 7: `ready_to_concede` Max Version Method

This method uses the maximum number of unique offers received from any partner. If the number of offers MiCRO has sent is less than or equal to this maximum, it is deemed ready to concede.

– Mean Version:

```
def ready_to_concede(self) -> bool:
    offers_sent = len(self._sent)

    mean_received = (
        sum(len(offers) for offers in self._received.values()) /
            len(self._received)
        if self._received else 0
    )
    return offers_sent <= mean_received
```

Listing 8: `ready_to_concede` Mean Version Method

This method calculates the average number of unique offers received per partner and checks if the number of offers MiCRO has sent is less than or equal to that average. If so, MiCRO is considered ready to concede by proposing a new offer.

- on partner proposal: This method is triggered when a partner proposes a new offer. It:
  - Initializes an offer set for that partner if none exists.
  - Records the newly received offer.
  - Passes the event up to the parent class.

```
    def on_partner_proposal(
        self, state: GBState, partner_id: str, offer: Outcome
    ) -> None:
        if partner_id not in self._received:
            self._received[partner_id] = set()
        self._received[partner_id].add(offer)
        return super().on_partner_proposal(state, partner_id, offer)
```

Listing 9: `on_partner_proposal` Method

- on partner response: This method is triggered when a partner responds to a MiCRO offer. If the partner's response is not an acceptance (i.e., response != 1), it assumes the partner still acknowledges the offer and records it.

```
    def on_partner_response(
        self,
        state: GBState,
        partner_id: str,
        outcome: Outcome | None,
        response: ResponseType,
    ) -> None:
        if partner_id not in self._received:
            self._received[partner_id] = set()
        if response != 1:
            self._received[partner_id].add(outcome)
        return super().on_partner_response(state, partner_id, outcome,
            response)
```

Listing 10: `on_partner_response` Method

The last block of code that is needed in order to have a functional agent is an Acceptance Policy.

```
@define
class MiCROAcceptancePolicyMulti(AcceptancePolicy):

    offering_strategy: MiCROOfferingPolicyMulti
    accept_same: bool = True

    def __call__(self, state, offer, source):
        if not self.negotiator or not self.negotiator.ufun:
            return ResponseType.REJECT_OFFER

        mine = (
            self.offering_strategy.next_offer()
            if self.offering_strategy.ready_to_concede()
            else self.offering_strategy.best_offer_so_far()
        )

        if self.negotiator.ufun.is_better(None, mine):
            mine = None

        is_acceptable = (
            self.negotiator.ufun.is_not_worse
            if self.accept_same
            else self.negotiator.ufun.is_better
        )

        if is_acceptable(offer, mine):
            self.offering_strategy.add_accepted_offer(offer)
            return ResponseType.ACCEPT_OFFER

        return ResponseType.REJECT_OFFER
```

Listing 11: `MiCROAcceptancePolicyMulti` Class Definition

7 Experimental Setup and Test Definitions

To rigorously evaluate the performance of various MiCRO agent versions, a series of comprehensive tests were designed and executed. All experiments were conducted within the NegMAS environment, leveraging its capabilities for simulating complex negotiation scenarios and managing agent interactions. A crucial aspect of this empirical evaluation was the selection of appropriate opponent agents and negotiation domains.

7.1 Selection of Opponent Agents

The primary criterion for selecting opponent agents was their demonstrated proficiency in multilateral negotiation settings. Consequently, we turned to the historical records of the Automated Negotiating Agents Competition (ANAC). Specifically, we focused on ANAC editions that featured tournaments for agents operating in multi-agent (three or more participants) negotiation scenarios, as opposed to purely bilateral encounters. Our selection process involved reviewing winners and notable participants from relevant ANAC years[6], [9], [16]:

- **ANAC 2015:** Atlas3 was selected as it was the winner of the individual utility category in a multilateral league. This agent is well-regarded for its robust performance in such settings.
- **ANAC 2016:** Cauduceus was the winner of the individual utility category. However, this agent was not readily available through the Genius-NegMAS bridge at the time of our experiments, precluding its inclusion.
- **ANAC 2017:** PonPokoAgent, the winner of the ANAC Repeated Multilateral Negotiation League (individual utility), was chosen for its demonstrated success in similar competitive environments.
- **ANAC 2018:** AgreeableAgent2018, also a winner of the ANAC Repeated Multilateral Negotiation League (individual utility), was included to represent another distinct and successful negotiation strategy.

Subsequent to ANAC 2018, the competition's focus shifted significantly towards bilateral negotiation scenarios. Therefore, to maintain relevance to multilateral testing environments, our final roster of opponent agents comprised: Atlas3, PonPokoAgent, and AgreeableAgent2018. These agents represent a diverse set of strategies developed by different research groups.

7.2 Negotiation Scenarios

For the negotiation scenarios (domains and preference profiles), we adopted the suite of scenarios utilized in the ANAC 2015 competition. This set of scenarios is well-documented and has been extensively used in prior research, providing a solid benchmark for evaluating negotiation agent performance in multilateral contexts. These scenarios define the issues under negotiation and the utility functions for each participating agent role.

7.3 Test Methodologies

1. **Comprehensive Agent Combination Tests:** This battery of tests involved running iterations across all negotiation scenarios from the ANAC 2015 suite using a set of three agents. For each scenario, every possible combination of agents was tested. It was specified that agents could be repeated within a session (for instance a session could consist of three MiCRO agents) and that the order of agents within a combination did not matter for defining a unique test case. This describes a selection process corresponding to combinations with repetition.

   The number of unique combinations for selecting k items from n types with repetition allowed and where order does not matter is given by the formula:

   $$C(n+k-1,k) = \binom{n+k-1}{k}$$

   For $n = 4$ distinct agent types and $k = 3$ agent slots per session, the number of unique combinations is:

   $$\binom{4+3-1}{3} = \binom{6}{3} = \frac{6\times 5\times 4}{3\times 2\times 1} = 20$$

2. **Utility Function Coverage Tests:** The objective of this second set of tests was to ensure the robustness of the observed behaviors from the comprehensive combination tests. Specifically, we aimed to mitigate the possibility that strong or weak performances were merely artifacts of particular utility functions aligning favorably or unfavorably with certain agent strategies. To achieve this, iterations were conducted ensuring that, for each scenario, every participating agent type was assigned each of the available utility functions defined within that scenario at least once. Due to the significant computational overhead associated with exhaustively testing all utility function permutations across all agent combinations, these tests were conducted using sessions composed of one agent of each of the primary selected types, so the four strategies were tested at the same time. This approach allowed for a focused verification of agent performance across the spectrum of utility profiles without the combinatorial explosion of the first test type.

All test results were logged and analyzed to compare the performance of MiCRO agent versions against the established ANAC agents across these diverse scenarios and utility distributions.

7.4 All agents combination test

The code for this section is the following:

```
def create_agents():
    return [
        AgreeableAgent2018(name="Agreeable"),
        PonPokoAgent(name="PonPoko"),
        Atlas3(name="Atlas3"),
        MiCRONegotiatorMulti(name="Micro"),
    ]
```

Listing 12: Creating a list of agents for testing

This function returns a list of initialized agents used in all test sessions. Including the three metioned agents and `MiCRONegotiatorMulti`.

```
def ANAC2015_scenarios_test():
    scenario_names = [
        "group1-university", "group2-dinner", "group2-politics", "group3-
            bank_robbery",
        "group5-car_domain", "group6-tram", "group8-holiday",
        "group9-killer_robot", "group9-vacation", "group11-car_purchase"]

    utilities = {"Agreeable": [], "PonPoko": [], "Atlas3": [], "Micro": []}
    utilities_on_agreement = {"Agreeable": [], "PonPoko": [], "Atlas3": [],
         "Micro": []}
    total_sessions = {"Agreeable": 0, "PonPoko": 0, "Atlas3": 0, "Micro":
        0}
    agreements_reached = {"Agreeable": 0, "PonPoko": 0, "Atlas3": 0, "Micro
        ": 0}
    performance_against_pairs = {}

    for scenario_name in scenario_names:
        print(f"\nRunning scenario: {scenario_name}")
        scenario_path = Path.home() / "negmas" / "scenarios" / "ANAC2015" /
             scenario_name
        try:
            scenario = Scenario.load(scenario_path)
        except (ET.ParseError, OSError, ValueError) as e:
            print(f"Skipping scenario '{scenario_name}' due to error: {e}")
            continue

        agents = create_agents()
        agent_combinations = list(combinations_with_replacement(agents, 3))
```

```
        for index, (agent1_class, agent2_class, agent3_class) in enumerate(
            agent_combinations, start=1):
            print(f"Iteration {index}/{len(agent_combinations)}: Running
                session for {agent1_class.name}, {agent2_class.name}, {
                agent3_class.name}")

            agent1 = type(agent1_class)(name=agent1_class.name)
            agent2 = type(agent2_class)(name=agent2_class.name)
            agent3 = type(agent3_class)(name=agent3_class.name)

            session = scenario.make_session(time_limit=35)
            session.add(agent1, preferences=scenario.ufuns[0].scale_max
                (1.0))
            session.add(agent2, preferences=scenario.ufuns[1].scale_max
                (1.0))
            session.add(agent3, preferences=scenario.ufuns[2].scale_max
                (1.0))

            print(session.run())
            process_session_results(session, utilities,
                utilities_on_agreement, total_sessions, agreements_reached,
                 performance_against_pairs)

    print_utility_statistics(utilities, utilities_on_agreement,
        total_sessions, agreements_reached)
    print_performance_against_pairs(performance_against_pairs)
    return performance_against_pairs
```

Listing 13: Main test function for all ANAC2015 scenarios

This function loads ANAC2015 scenarios, iterates through combinations of agents, and runs multi-agent negotiation sessions for each. It tracks and stores utility outcomes, agreement status, and performance metrics.

```
def process_session_results(session, utilities, utilities_on_agreement,
                            total_sessions, agreements_reached,
                                performance_against_pairs):
    for negotiator in session.negotiators:
        agent_name = negotiator.name
        utility = negotiator.ufun(session.agreement) if session.agreement
            else negotiator.reserved_value

        other_agents = [a.name for a in session.negotiators if a.id !=
            negotiator.id]
        pair = tuple(sorted(other_agents))
        key = (pair, agent_name)

        performance_against_pairs.setdefault(key, []).append(utility)
        total_sessions[agent_name] += 1

    proposer = session.state.current_proposer

    if session.agreement is not None:
        print("    Final Agreement:", session.agreement)
        for negotiator in session.negotiators:
            role = "Proposer" if negotiator.id == proposer else "Responder"
            ufun_value = negotiator.ufun(session.agreement)
            utilities[negotiator.name].append(ufun_value)
            utilities_on_agreement[negotiator.name].append(ufun_value)
            agreements_reached[negotiator.name] += 1
```

```
            print(f"    {role} - Negotiator '{negotiator.name}' (ID: {
                negotiator.id}) Utility: {ufun_value:.4f}")
    else:
        print("    No agreement reached in the session.")
        for negotiator in session.negotiators:
            utilities[negotiator.name].append(negotiator.reserved_value)
```

Listing 14: Processing results from each negotiation session

After each session, this function calculates individual utilities, identifies if agreements were reached, and updates overall performance statistics.

```
def print_utility_statistics(utilities, utilities_on_agreement,
    total_sessions, agreements_reached):
    print("\n### Utility Statistics ###")
    for name, values in utilities.items():
        if values:
            mean_utility = statistics.mean(values)
            utility_on_agreement = statistics.mean(utilities_on_agreement[
                name]) if name in utilities_on_agreement else 0
            std_dev = statistics.stdev(values) if len(values) > 1 else 0
            std_error = std_dev / math.sqrt(len(values)) if len(values) > 1
                else 0
            agreement = (agreements_reached[name] / total_sessions[name]) *
                100 if total_sessions[name] > 0 else 0
        else:
            mean_utility, utility_on_agreement, std_error, agreement = 0,
                0, 0, 0

        print(f"{name}: Mean Utility = {mean_utility:.4f},
            Utility_on_agreement = {utility_on_agreement:.4f},"
              f" Std Error = {std_error:.4f}, Agreement = {agreement:.4f}%"
                )
```

Listing 15: Displaying utility statistics

This function computes and prints the following summary statistics: mean utility, standard error and agreement rate for each agent across all sessions.

```
def print_performance_against_pairs(performance_against_pairs):
    print("\n### Sorted Performance Against Pairs ###")
    pair_to_agents = {}

    for (pair, agent), values in performance_against_pairs.items():
        mean_utility = sum(values) / len(values)
        pair_to_agents.setdefault(pair, []).append((agent, mean_utility))

    for pair in sorted(pair_to_agents.keys()):
        print(f"\nAgainst pair {pair}:")
        sorted_agents = sorted(pair_to_agents[pair], key=lambda x: x[1],
            reverse=True)
        for agent, mean_utility in sorted_agents:
            print(f"  Agent {agent}     average utility: {mean_utility:.4f}
                ")
```

Listing 16: Print sorted performance by agent pairs

This function sorts and displays the average utility that each agent achieved against every other pair of agents. It provides a detailed view of competitive performance.

```
def get_best_agents_against_pairs(performance_against_pairs):
    best_agents = {}
    pair_to_agents = {}
```

```
    for (pair, agent), values in performance_against_pairs.items():
        mean_utility = sum(values) / len(values)
        pair_to_agents.setdefault(pair, []).append((agent, mean_utility))

    for pair, agent_utils in pair_to_agents.items():
        agent_utils.sort(key=lambda x: x[1], reverse=True)
        best_agents[pair] = agent_utils[0][0]

    return best_agents
```

Listing 17: Identify the best-performing agent against each pair

Given the performance data, this function identifies which agent performed best (on average) against each possible pair of opponents. This is used for simulating best-response dynamics.

```
def simulate_best_response_dynamics(agent_names, best_agents_by_pair,
    max_steps=20, seed=None):
    if seed is not None:
        random.seed(seed)

    state = tuple(random.choices(agent_names, k=3))
    seen_states = set()
    transitions = []

    for step in range(max_steps):
        seen_states.add(state)
        current_state = list(state)

        for player in range(3):
            others = tuple(sorted([current_state[i] for i in range(3) if i
                != player]))
            best_response = best_agents_by_pair.get(others)

            if best_response and best_response != current_state[player]:
                next_state = current_state.copy()
                next_state[player] = best_response
                next_state = tuple(next_state)
                transitions.append((state, next_state, player))
                state = next_state
                break
        else:
            break

    return transitions, state
```

Listing 18: Simulate strategy updates until Nash Equilibrium

Simulates how agent strategies evolve over time by switching one agent at a time to its best response against the current pair. The process continues until convergence known as nash-equilibrium.

```
def plot_best_response_graph(agent_names, best_agents_by_pair):
    G = nx.DiGraph()
    edge_colors = []
    edge_labels = {}
    player_color = {0: 'blue', 1: 'green', 2: 'red'}

    all_states = list(combinations_with_replacement(agent_names, 3))
    out_edges = defaultdict(set)

    for state in all_states:
        sorted_state = tuple(sorted(state))
        for player in range(3):
```

```
            current_state = list(sorted_state)
            others = current_state[:player] + current_state[player+1:]
            others_sorted = tuple(sorted(others))
            best_response = best_agents_by_pair.get(others_sorted,
                current_state[player])

            if best_response != current_state[player]:
                new_state = tuple(sorted(others + [best_response]))
            else:
                new_state = tuple(sorted(current_state))

            G.add_edge(sorted_state, new_state, label=f"P{player + 1}")
            edge_labels[(sorted_state, new_state)] = f"P{player + 1}"
            edge_colors.append(player_color[player])
            out_edges[sorted_state].add(new_state)

    equilibria = [node for node, targets in out_edges.items() if len(
        targets) == 1 and node in targets]

    pos = nx.shell_layout(G, scale=3.0)
    node_colors = ['lightgreen' if node in equilibria else 'lightblue' for
        node in G.nodes()]
    nx.draw_networkx_nodes(G, pos, node_size=900, node_color=node_colors,
        edgecolors='black')

    nx.draw_networkx_edges(
        G,
        pos,
        connectionstyle="arc3,rad=0.1",
        arrows=True,
        arrowstyle='-|>',
        arrowsize=10,
        width=1.5,
        edge_color=edge_colors,
        min_source_margin=15,
        min_target_margin=15
    )

    nx.draw_networkx_labels(G, pos, font_size=6)
    nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size
        =5, label_pos=0.5)

    from matplotlib.lines import Line2D
    legend_elements = [
        Line2D([0], [0], color='blue', lw=2, label='Player 1'),
        Line2D([0], [0], color='green', lw=2, label='Player 2'),
        Line2D([0], [0], color='red', lw=2, label='Player 3'),
        Line2D([0], [0], marker='o', color='w', label='Nash Equilibrium',
               markerfacecolor='lightgreen', markersize=10, markeredgecolor
                   ='black'),
    ]
    plt.legend(handles=legend_elements, loc='upper right', fontsize=8)
    plt.title("Best-Response Dynamics", fontsize=12)
    plt.axis('off')
    plt.tight_layout(pad=2.0)
    plt.margins(x=0.2, y=0.2)
    plt.show()
```

Listing 19: Visualize best-response transitions as a graph

Visualizes the dynamics of best-response transitions as a directed graph, with nodes representing agent trios and edges showing improvements by one agent.

```
if __name__ == "__main__":
    start_time = datetime.now()
    print(f"Start Time: {start_time}")

    performance_against_pairs = ANAC2015_scenarios_test()

    agent_names = ["Agreeable", "PonPoko", "Atlas3", "Micro"]
    best_agents_by_pair = get_best_agents_against_pairs(
        performance_against_pairs)

    transitions, final_state = simulate_best_response_dynamics(agent_names,
         best_agents_by_pair)
    plot_best_responde_graph(transitions, final_state)

    end_time = datetime.now()
    print(f"End Time: {end_time}")
    print(f"Execution Time: {end_time - start_time}")
```

Listing 20: Main execution block

This is the entry point of the script. It tracks execution time, runs the scenario tests, simulates best-response learning, and plots the results. This script was ran 5 times in order to have the least std. error as possible.

7.5 All utility functions test

This function iterates over multiple scenarios and runs negotiation sessions ensuring that all the agents uses at least once every preference set of each scenario,it cycles through all available utility functions in each scenario to test robustness across configurations.

Again the script was ran 5 times in order to have the least std. error as possible.

```
def ANAC2015_ufuns_combinatios():
    scenario_names = [...]

    utilities = {"Agreeable": [], "PonPoko": [], "Atlas3": [], "Micro": []}
    utilities_on_agreement = {"Agreeable": [], "PonPoko": [], "Atlas3": [],
         "Micro": []}
    total_sessions = {"Agreeable": 0, "PonPoko": 0, "Atlas3": 0, "Micro":
        0}
    agreements_reached = {"Agreeable": 0, "PonPoko": 0, "Atlas3": 0, "Micro
        ": 0}
    performance_against_pairs = {}
    for i in range(5):
        print("ITERATION:", i)
        for scenario_name in scenario_names:
            print(f"\nRunning scenario: {scenario_name}")
            scenario_path = Path.home() / "negmas" / "scenarios" / "
                ANAC2015" / scenario_name
            try:
                scenario = Scenario.load(scenario_path)
            except (ET.ParseError, OSError, ValueError) as e:
                print(f"Skipping scenario '{scenario_name}' due to error: {
                    e}")
                continue

            for preference_set in range(len(scenario.ufuns)):
                print(f"  Testing preference set {preference_set + 1} of {
                    len(scenario.ufuns)}")

                session = scenario.make_session(time_limit=45)
```

```
                agent1 = MiCRONegotiatorMulti(
                    preferences=scenario.ufuns[preference_set % len(
                        scenario.ufuns)].scale_max(1.0),
                    name="Micro")
                agent2 = Atlas3(
                    preferences=scenario.ufuns[(preference_set + 1) % len(
                        scenario.ufuns)].scale_max(1.0),
                    name="Atlas3")
                agent3 = PonPokoAgent(
                    preferences=scenario.ufuns[(preference_set + 2) % len(
                        scenario.ufuns)].scale_max(1.0),
                    name="PonPoko")
                agent4 = AgreeableAgent2018(
                    preferences=scenario.ufuns[(preference_set + 3) % len(
                        scenario.ufuns)].scale_max(1.0),
                    name="Agreeable")

                session.add(agent1)
                session.add(agent2)
                session.add(agent3)
                session.add(agent4)

                print(session.run())
                process_session_results(session, utilities,
                    utilities_on_agreement, total_sessions,
                    agreements_reached, performance_against_pairs)
    print_utility_statistics(utilities, utilities_on_agreement,
        total_sessions, agreements_reached)
    print_performance_against_pairs(performance_against_pairs)
    return performance_against_pairs
```

Listing 21: Testing multiple utility function combinations in various ANAC2015 scenarios

7.6 Problems faced

Here is a list of the main challenges encountered during the development and evaluation of this work, along with the solutions adopted to address them.

- **Access to ANAC domains:** The negotiation domains used in the ANAC competitions hard to find and difficult to use and implement at the time this thesis was made. I was granted access to the ANAC 2015 domain set through direct collaboration with Dave de Jonge, who shared the relevant data for research purposes. As a result, all the experiments and evaluations in this thesis have been conducted exclusively using the 2015 domains.
- **Limitations of NegMAS for multilateral tournaments:** The NegMAS framework, while powerful for bilateral negotiation simulations, did not offer at the time the experiments were conducted a built-in support for multilateral tournament execution. Although it contains some preliminary support for multi-agent sessions, I encountered significant technical limitations and unexpected behaviors. After weeks of experimentation and direct communication with the creator of NegMAS, Yasser Farouk Mohammad, the issues persisted. In the end, I decided to manually implement the tournament structure by scripting the required combinations, iterations, and metric-logging mechanisms. This solution allowed full control over the simulation logic and data collection.
- **Incorrect early testing with KakeSoba:** Initially, I conducted my strategy evaluations using the KakeSoba agent, the second-place finisher in ANAC 2019. I mistakenly assumed that the 2019 edition of ANAC featured multilateral negotiation, when in fact it was a bilateral competition. As a result, the tests I performed during that period used a non-compatible benchmark agent and led to invalid conclusions. Once this misunderstanding was

identified, I re-ran all experiments using appropriate multi-agent configurations and updated the corresponding analyses accordingly.

- **Deadlock in MiCRO-Multi with the Minimum strategy:** As discussed in Section 6.4 early in the testing of the extended MiCRO in multilateral scenarios using the *Minimum* variant, we encountered a critical deadlock situation. This issue revealed a fundamental limitation in the original multi-agent design of MiCRO: agents using the Minimum strategy would refuse to make new proposals if they had already made more offers than any other participant. When all agents adopt this behavior, the negotiation can completely stall, with none of them initiating further proposals.

  Resolving this deadlock required modifying MiCRO's core logic. Specifically, we adjusted the algorithm so that accepted offers would also count toward an agent's number of actions, not just explicit proposals. Although this change was introduced solely to prevent deadlocks, it affected the behavior of MiCRO in all negotiation settings since agents now appeared to act more frequently under the new rule.

  To ensure that our evaluation remained consistent and fair, we re-ran all experiments using this updated version of MiCRO. This ensured the validity of performance comparisons across all strategies and preserved the integrity of our experimental results.

8 Results

This section presents the results of two experimental setups: the **All Agents Combination Test** and the **All Utility Functions Test**. These experiments were designed to evaluate and compare the performance of the three MiCRO variants (Minimum, Mean, and Maximum) against three well-established agents: AgreeableAgent2018, PonPokoAgent, and Atlas3, each a top-performing multilateral agent from past ANAC competitions.

The experiments aim to assess MiCRO's adaptability to multilateral environments and to explore whether, as in bilateral settings [10], its simplicity exposes limitations in current benchmarking domains such as those from ANAC 2015.

How to Interpret the Tables

Each result table in this section displays four key metrics per agent:

- **Mean Utility**: This is the primary performance indicator and represents the agent's average utility across all negotiation sessions. It is the main value used to compare agents' effectiveness.
- **Utility on Agreement**: This metric reports the agent's average utility only in those sessions where an agreement was reached. It helps explain whether an agent concedes more or less when it does reach agreement, but it should not be used by itself to assess performance since it ignores failed negotiations.
- **Agreement Rate**: This value indicates the percentage of negotiation sessions that ended in an agreement. Like utility on agreement, it is not a direct measure of quality, but it helps to contextualize the mean utility. For example, a low mean utility may be due to frequent negotiation failures, while a high utility on agreement may show that the agent only agrees under favorable terms.
- **Standard Error**: This metric reflects how accurately we have measured the mean utility. The smaller it is, the more reliable the mean utility is.

In summary, the **Mean Utility** is the most important metric for ranking agents. The other values, Utility on Agreement and Agreement Rate, are useful for understanding the underlying behavior and reasons behind the agent's performance, but should not be interpreted as standalone measures of strategy quality.

Note: Mean Utility, Std Error and Utility on Agreement, have been all multiplied by a factor of 100 for facilitate its readability on the tables.

8.1 All Agents Combination Test

This test evaluates how each agent performs when paired with different combinations of the other agents. It provides insight into the agent's robustness across varying strategic environments.

Minimum Version:

Agent	Mean Utility ±Std Error	Utility on Agreement	Agreement Rate
MiCRO-Min	81.82 ±0.91	88.87	91.7333%
Agreeable	80.36 ±0.80	81.92	97.8667%
PonPoko	78.08 ±1.03	87.78	88.0000%
Atlas3	75.77 ±0.76	76.30	99.2000%

Table 2: All agents combination test - Minimum Version

- MiCRO shows the highest **mean utility** (0.8182) and the highest **utility on agreement** (0.8887), outperforming all other agents.
- The **agreement rate** (91.73%) is slightly below Atlas3 (99.2%) and Agreeable (97.87%), suggesting that although MiCRO Minimum is more selective, its agreements tend to be highly favorable.
- PonPoko, while achieving strong utility on agreement (0.8778), suffers a lower agreement rate (88%), which slightly reduces its overall mean utility.

Maximum Version:

Agent	Mean Utility ±Std Error	Utility on Agreement	Agreement Rate
Agreeable	82.95 ±0.72	83.42	99.3333%
PonPoko	82.44 ±0.89	89.22	91.7333%
MiCRO-Max	81.51 ±0.66	82.74	98.4000%
Atlas3	77.37 ±0.77	77.82	99.3333%

Table 3: All agents combination test - Maximum Version

- Agreeable and PonPoko show a marginal edge in **mean utility** (0.8295 and 0.8244, respectively) compared to MiCRO Maximum (0.8151), likely due to their aggressive concession tactics.
- Atlas3 consistently trails in both utility and agreement rate, reaffirming its relatively weaker performance in these multilateral configurations.
- MiCRO Maximum balances a high agreement rate (98.4%) with stable utility values, confirming its alignment with high-frequency negotiation environments.

Mean Version:

Agent	Mean Utility ±Std Error	Utility on Agreement	Agreement Rate
MiCRO-Mean	82.48 ±0.82	87.99	93.4667%
Agreeable	80.49 ±0.79	81.59	98.5333%
PonPoko	79.85 ±0.97	88.37	89.4667%
Atlas3	76.27 ±0.77	76.70	99.3333%

Table 4: All agents combination test - Mean Version

- MiCRO once again achieves the **highest mean utility** (0.8248) and strong utility on agreement (0.8799).
- AgreeableAgent2018 has higher agreement rates but gains slightly lower utility from those agreements.

- PonPoko shows strong utility on agreement but remains less consistent across sessions due to lower agreement rates (89.47%).

Summary: In this test, MiCRO's performance is competitive across all versions. The Minimum version secures the most valuable agreements, while the Mean and Maximum versions trade off some utility for improved agreement frequency.

8.2 All Utility Functions Test

This test changes the agents' utility functions across multiple negotiation sessions, simulating variability in domain and preference configurations while keeping the opponent agents fixed.

Minimum Version:

Agent	Mean Utility ±Std Error	Utility on Agreement	Agreement Rate
MiCRO-Min	71.80 ±1.77	90.08	79.0361%
PonPoko	65.86 ±1.63	82.21	79.0361%
Agreeable	61.75 ±1.69	76.78	79.0361%
Atlas3	53.80 ±1.52	67.30	79.0361%

Table 5: All utility functions test - Minimum Version

- MiCRO achieves the highest **mean utility** (0.7180) and **utility on agreement** (0.9008), consistent with its conservative and selective behavior.
- All agents show an identical agreement rate (79.03%), this makes sense as all of the agents took part on the same number of negotiations together.
- Despite equal opportunity, MiCRO consistently secures better outcomes when agreements occur.

Maximum Version:

Agent	Mean Utility ±Std Error	Utility on Agreement	Agreement Rate
PonPoko	76.89 ±1.35	85.80	89.1566%
Agreeable	73.96 ±1.42	82.66	89.1566%
MiCRO-Max	67.40 ±1.37	75.34	89.1566%
Atlas3	057.52 ±1.35	64.13	89.1566%

Table 6: All utility functions test - Maximum Version

- PonPoko leads in both **mean utility** (0.7689) and utility on agreement (0.8580), likely due to its aggressive, high-concession approach.
- MiCRO Maximum scores slightly below (0.6740 mean utility), reflecting its susceptibility to over-concession in adversarial or unbalanced domains.
- The agreement rate is still identical across all the agents, due to the nature of the tests. This time is higher than with the minimum version due to over-concession from this version of Micro.

Mean Version:

Agent	Mean Utility ±Std Error	Utility on Agreement	Agreement Rate
MiCRO-Mean	74.06 ±1.45	85.25	86.5060%
PonPoko	73.38 ±1.42	84.16	86.5060%
Agreeable	67.79 ±1.50	77.75	86.5060%
Atlas3	56.11 ±1.37	64.38	86.5060%

Table 7: All utility functions test - Mean Version

- MiCRO Mean achieves the highest **mean utility** (0.7406), confirming its robustness across diverse preference profiles.
- PonPoko follows closely with 0.7338, while Agreeable and Atlas3 lag behind.
- All agents exhibit an identical agreement rate (86.51%), the Agreement Rate is balanced between the Minimum and the Maximum version, showing again the balanced nature of the Mean version.

Summary: Across varied utility functions, MiCRO variants remain competitive, with the Mean version emerging as the most consistently effective variant in terms of mean utility and agreement quality.

8.3 Results Analysis and Interpretation

Performance of MiCRO Variants

MiCRO demonstrates a strong capacity to compete with and even outperform more complex agents like PonPoko and AgreeableAgent2018. Notably:

- The **Minimum version** excels in *utility on agreement*, confirming its strength in achieving highly favorable deals, though often at the expense of fewer agreements.
- The **Maximum version** consistently performs well in *mean utility* and *agreement rate*, indicating its value in scenarios where speed and frequency are prioritized over individual gain.
- The **Mean version** offers the best overall trade-off, with strong results in both agreement rate and utility measures, particularly under diverse preference scenarios.

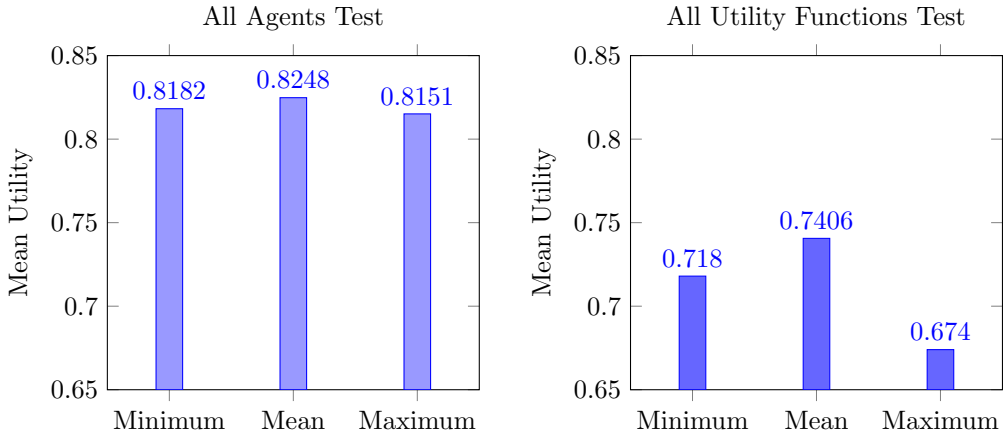

Figure 6: Comparison of MiCRO Mean Utility across versions and tests

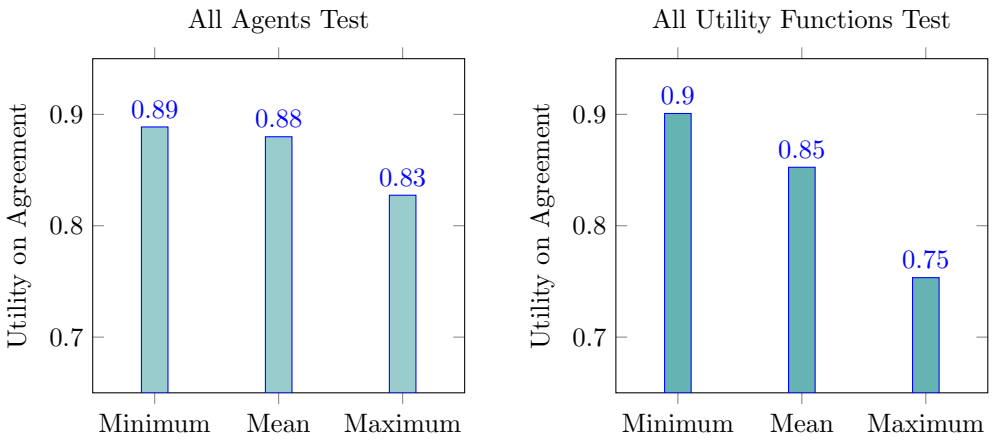

Figure 7: Comparison of MiCRO Utility on Agreement across versions and tests

Implications for ANAC as a Benchmark

As shown by Jonge [10], MiCRO's success in bilateral negotiation exposed weaknesses in ANAC domains that allowed simple, model-free strategies to outperform sophisticated agents. These results extend that argument into the multilateral setting.

Despite MiCRO's reactive and non-adaptive nature it was able to challenge, and in some configurations even to outperform, top agents like PonPoko, Atlas3, and AgreeableAgent2018. However, it is important to note that in many cases, the differences in mean utility between MiCRO and the best-performing agents, particularly Agreeable, fall within the range of their respective standard errors. This suggests that such differences are not statistically significant, and caution should be exercised before drawing strong conclusions about MiCRO's superiority in these settings.

Nonetheless, the fact that a conceptually minimal and model-free agent can match the performance of top competition entries across multiple domains raises concerns about the benchmarking capabilities of the ANAC framework. It implies that current ANAC domains may still lack the necessary strategic richness or adversarial dynamics to truly differentiate between sophisticated learning-based strategies and well-tuned heuristics.

Therefore, these experiments support the argument that:

- **ANAC domains can still be exploited by simple heuristics** in multilateral negotiation.
- Benchmarking systems must include richer, more adversarial, and less predictable domains to better differentiate strategy sophistication.

Conclusion of the Experiments

MiCRO's variants, particularly the Mean and Minimum versions, show that the strategy can achieve high performance in complex multi-agent environments. However, while these performances are competitive, the proximity of utility scores to those of leading agents, in light of their standard errors, suggests that such outcomes may not reflect statistically significant differences. Therefore, although MiCRO demonstrates strong potential, claims of clear outperformance must be tempered by statistical considerations.

The experiments nonetheless validate MiCRO's adaptability and emphasize a broader concern: the continuing limitations of existing benchmarks like ANAC 2015 in evaluating negotiation strategies. The results underscore the need for evaluation protocols that can distinguish between genuine strategic sophistication and reactive efficiency.

8.4 Concluding Remarks

The results presented in this section provide several valuable insights into both the behavior of the MiCRO strategy in multilateral negotiation scenarios and the suitability of ANAC domains as

benchmarking tools.

From a performance standpoint, MiCRO demonstrated competitive results against top-performing ANAC agents such as AgreeableAgent2018, AgentAtlas3 and PonPokoAgent. While it does not dominate any single metric across the board, its consistent performance across configurations confirms its robustness and versatility. In particular, the three variants of MiCRO allow it to adapt to different negotiation priorities: the **Minimum** version excels in achieving highly favorable agreements, the **Maximum** version ensures a high agreement rate, and the **Mean** version offers a pragmatic middle ground suitable for balanced settings.

A key insight is the effectiveness of MiCRO's internal decision logic. By basing its concession behavior on the proposal activity of other agents, MiCRO indirectly captures strategic intent without relying on explicit opponent modeling or learning. This approach is computationally lightweight, adaptable, and resistant to certain forms of adversarial manipulation. The results confirm that even simple, reactive mechanisms, when carefully designed, can rival more complex, learning-based agents in practice.

Most importantly, these findings reaffirm one of the central hypotheses of this thesis: that the ANAC benchmarking framework, especially its negotiation domains, may not always provide sufficient complexity to differentiate truly advanced strategies from simple heuristics. The fact that a strategy like MiCRO, model-free and conceptually minimal, can match or surpass state-of-the-art agents in many scenarios suggests that ANAC domains, even in multilateral form, can still be **exploited** by carefully tuned reactive strategies. This observation extends the conclusions of Jonge [10] from bilateral to multilateral contexts.

Therefore, this section not only highlights the value of MiCRO as a competitive and tunable strategy, but also raises important questions about the continued use of traditional ANAC domains as definitive benchmarks for negotiation strategy evaluation. Future work should aim to introduce richer, more dynamic domains and consider more adversarial or cooperative environments that can better reveal the limits of both reactive and adaptive strategies.

The empirical evidence presented here supports the broader thesis argument: **evaluating negotiation strategies solely on current ANAC-style benchmarks may lead to overestimating their general effectiveness**, particularly when such benchmarks can be dominated by non-adaptive heuristics like MiCRO.

While MiCRO does not unequivocally outperform top agents like AgreeableAgent2018 or PonPoko in isolated metrics, its design shows a strong emphasis on strategic adaptability. The tunable behavior from conservative (Minimum) to aggressive (Maximum) allows MiCRO to fit different deployment contexts.

Furthermore, the internal logic based on reactive proposal counts introduces a novel and effective mechanism for indirect opponent modeling, without requiring explicit profiling or learning. This not only reduces computational complexity but also enhances robustness in variable or adversarial settings.

The insights from these results suggest promising directions for future negotiation agents particularly in how they balance responsiveness, robustness, and efficiency through dynamically modulated concession policies.

Aspect	Minimum Version	Mean Version	Maximum Version
Mean Utility	Moderate	High	High
Utility on Agreement	Highest	High	Lowest
Agreement Rate	Lowest	Moderate	Highest
Robustness	Highest	Moderate	Lowest
Vulnerability to Opponent Behavior	Low (resistant to spamming)	Moderate	High (reactive to active opponents)
Recommended Context	Adversarial environments, low trust	Balanced or mixed-agent systems	High-frequency markets, fast convergence needed

Table 8: Summary of MiCRO Variant Characteristics

8.5 Best MiCRO Version

Each of the three MiCRO variants, Minimum, Mean and Maximum, offers distinct trade-offs between utility, agreement rate, and robustness. While all versions are competitive in multilateral settings, as demonstrated by the experiments, selecting the "best" variant depends strongly on the context in which the negotiation takes place.

The **Maximum version** exhibits high agreement rates and performs well in environments where the value of frequent agreements outweighs the importance of maximizing individual utility. However, its behavior is highly reactive: it mirrors the most active proposer in the negotiation. This means that if one or more agents continuously submit offers, the Maximum version will also be forced to accelerate its concessions potentially far beyond what would be considered strategically optimal. As discussed in Section 6.6 in adversarial or collusive settings, this can be exploited to push MiCRO into premature and suboptimal agreements.

The **Mean version**, while more balanced, is still susceptible to the same form of manipulation. Since it responds based on the average number of proposals made by other agents, a coordinated or hyperactive subset of agents can inflate that average and indirectly pressure MiCRO to make concessions earlier than necessary. Although less extreme than the Maximum variant, this vulnerability makes the Mean version inappropriate for high-risk or untrusted negotiation environments.

In contrast, the **Minimum version** offers a robust and stable behavior pattern that is inherently **resistant to manipulation**. Its decision rule, only proposing a new offer once all other agents have made at least the same number of proposals or acceptances, prevents external agents from unilaterally dictating its concession rate. Even if multiple agents attempt to overwhelm MiCRO with a flood of proposals, the Minimum version will simply refrain from moving until all participants are equally active. This makes it significantly harder to exploit.

Given this resistance to coordinated manipulation, the Minimum version stands out as the most strategically safe option for deployment in open, unknown, or adversarial environments. While it may result in lower agreement rates in scenarios with limited communication, the agreements it does reach tend to be highly favorable, and its stability ensures consistent behavior regardless of the surrounding agents' strategies.

Therefore, the Minimum version should be considered the default and recommended variant of MiCRO for general use, especially in contexts where opponent behavior is not fully known or cannot be controlled.

8.6 Game-Theoretic Analysis of MiCRO as a Strategic Choice

To gain deeper insight into the strategic robustness and adaptability of each MiCRO variant, we conducted a game-theoretic analysis grounded in best-response dynamics. Game-theoretical evaluations are particularly important in negotiation research because they reveal how strategies

perform not just in isolation, but in interactive, competitive contexts where each agent's success depends on the strategies chosen by others. This allows us to assess the stability, exploitability, and equilibrium behavior of negotiation strategies and key dimensions that are often obscured by purely performance-based metrics.

In our analysis, we simulate a setting in which three players must simultaneously select a strategy from the set {Agreeable, PonPoko, Atlas3, MiCRO variant}. The expected utility of each strategy combination is derived from the **average utility** scores obtained in the All Agents Combination Test, which provides a reliable estimate of strategic performance in a multilateral context.

For each triplet of players, we fix two strategies and allow the third player to unilaterally deviate in search of a higher expected utility. This models the concept of a *best response*, a central idea in game theory where a player adjusts its strategy to maximize its payoff given the choices of others.

The results are visualized as directed graphs, one for each MiCRO variant (Minimum, Mean, and Maximum).

In these graphs:

- **Nodes** represent a specific combination of strategies selected by the three players, for example: `(Player 1: Atlas3, Player 2: PonPoko, Player 3: MiCRO)`.
- **Directed Edges** (colored arrows) illustrate a player's *best response*. An arrow from one node to another signifies that the corresponding player (P1: blue, P2: green, P3: red) can increase their utility by unilaterally changing their strategy, thus moving the game to the new node.
- **Nash Equilibrium** (highlighted in green) are nodes with no outgoing edges. In these states, no single player has an incentive to change their strategy, given the strategies of the others. These represent stable outcomes of the game.

Analysis and Conclusions

An examination of the best-response dynamics for each MiCRO variant reveals significant differences in their strategic stability.

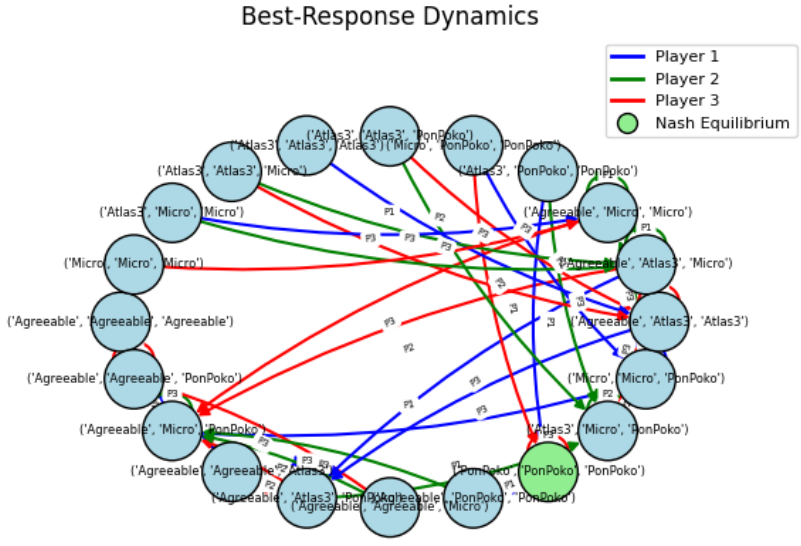

Figure 8: A graph showing the best response dynamics for each 20 possible states for MiCRO-max.

MiCRO-Maximum (Max variant) The analysis shows a single, dominant *Pure-Strategy Nash Equilibrium*: `('PonPoko', 'PonPoko', 'PonPoko')`. The dynamics of the graph indicate that, regardless of the initial combination of strategies, the players are consistently incentivized to switch their strategies, eventually converging on the state where all three adopt PonPoko. The MiCRO-Max strategy does not appear in any stable equilibrium. This suggests that, despite any potential high scores in specific matchups, it is strategically unstable and ultimately driven out in an adaptive environment where opponents select their best responses. This it is like this due to the fact that MiCRO-Max is the MiCRO version which concedes the most, so will have a low average utility against pairs thus making it not a good option for players to choose.

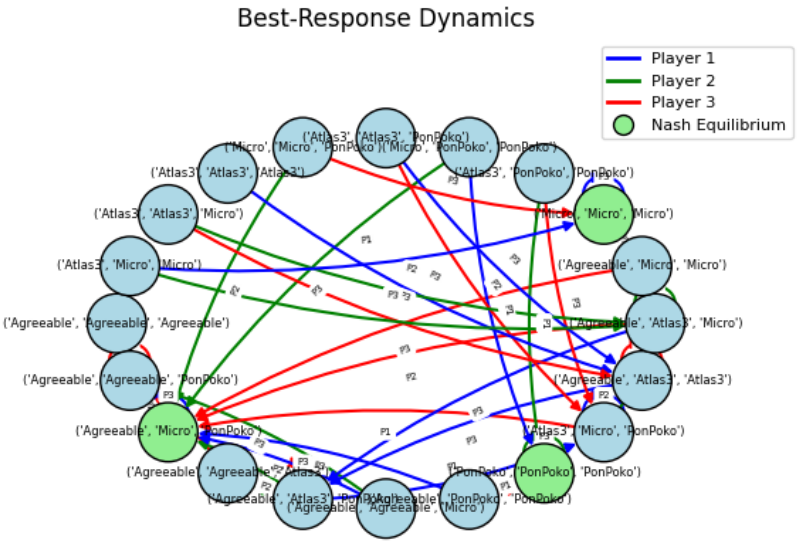

Figure 9: A graph showing the best response dynamics for each 20 possible states for MiCRO-mean.

MiCRO-Mean (Mean variant) In contrast, the introduction of the MiCRO-Mean variant fundamentally alters the strategic landscape. The analysis identifies three distinct Nash Equilibria:

1. The pure `('PonPoko', 'PonPoko', 'PonPoko')` equilibrium persists, highlighting the robust nature of the PonPoko strategy.
2. A new equilibria emerge, each being a permutation of `('Agreeable', 'MiCRO-Mean', 'PonPoko')`.
3. `('MiCRO-Mean', 'MiCRO-Mean', 'MiCRO-Mean')` constitudes a nash equilibrium by itself.

The existence of these additional equilibrium is a critical finding. It demonstrates that MiCRO-Mean is a *conditionally robust strategy*. It can be a player's optimal choice and part of a stable outcome, coexisting with other two different strategies Agreeable and PonPoko and also it reaches nash equilibrium when the three players chooses MiCRO-Mean as their strategy.

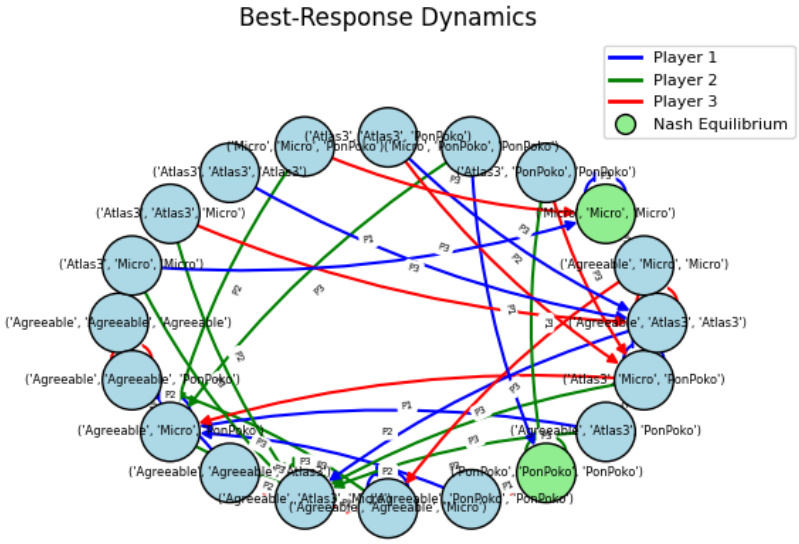

Figure 10: A graph showing the best response dynamics for each 20 possible states for MiCRO-min.

MiCRO-Minimum (Min variant) The results for the MiCRO-Minimum variant are very similar to those of the Mean variant. But this analysis only reveals two Nash Equilibrium: the persistent `('PonPoko', 'PonPoko', 'PonPoko')` state and the three permutations of `('MiCRO-Min', 'MiCRO-Min', 'MiCRO-Min')`.

In conclusion, the game-theoretic analysis shows that while the MiCRO-Max variant is strategically fragile, both the **MiCRO-Mean** and **MiCRO-Minimum** variants demonstrate significant strategic robustness. Their ability to form stable equilibrium in combination with other diverse strategies indicates a higher level of adaptability. They do not simply survive in a competitive environment but actively contribute to creating new, stable social outcomes, making them more viable and versatile for complex, multilateral negotiations.

Strategic Implications

These graphs collectively reinforce the conclusion that MiCRO is a good strategy that can perform well against most of the state-of-art strategies. It frequently appears in equilibrium configurations and resists displacement by best-response adaptations.

In summary, from a game-theoretic perspective, MiCRO consistently constitutes part of an equilibrium in multilateral settings. This finding provides a formal justification for recommending MiCRO as a benchmark when testing new strategies in a multi-agent context.

9 Conclusions

This thesis explored the development and evaluation of automated negotiation strategies in multilateral settings, with a specific focus on the MiCRO strategy: a minimal, model-free, reactive algorithm originally designed for bilateral negotiation domains. Building on prior work by de Jonge [10], this study aimed to determine whether MiCRO's simplicity and effectiveness could generalize to multilateral negotiation environments, and what this would reveal about the benchmarks used to evaluate such agents.

To this end, three variants of MiCRO were implemented and tested: Minimum, Mean, and Maximum. Each variant adapted MiCRO's core idea to multilateral contexts by adjusting the

concession behavior in relation to the number of proposals made by other agents. The variants were evaluated against top-performing agents from past ANAC multilateral competitions, using both fixed agent combinations and randomized utility functions.

The results demonstrate that:

- MiCRO remains highly competitive in multilateral negotiation, often matching or outperforming more complex and adaptive agents who have won previous ANAC competitions.
- The Minimum variant, while more conservative and slower to reach agreements, achieves the highest utility on agreement and exhibits robust behavior that is resistant to manipulation.
- The Maximum and Mean variants can perform well in collaborative environments but are vulnerable to exploitative behavior when other agents manipulate offer pacing.
- Despite its simplicity, MiCRO frequently participates in states of equilibrium under best-response dynamics, reinforcing its theoretical solidity and practical reliability.

In addition, this thesis revealed persistent limitations in the benchmarking standards used by the ANAC competition. Many ANAC domains, despite being multilateral, can still be exploited by non-adaptive strategies like MiCRO, raising concerns about their effectiveness in evaluating true negotiation intelligence. As in bilateral settings, the ability of MiCRO to succeed without opponent modeling or learning indicates that some ANAC environments may still lack the strategic depth required for meaningful assessment.

Overall, this work confirms that a simple strategy like MiCRO can offer significant advantages in multilateral negotiations, especially when robustness, interpretability, and low computational complexity are desired. Among its variants, MiCRO-Minimum emerges as the most strategically safe and general-purpose option, suitable for use in open and unpredictable negotiation environments.

9.1 Future Work

Several directions emerge naturally from the findings of this thesis:

- **Domain complexity analysis:** A systematic study of the structural properties of ANAC negotiation domains would help identify which domains are genuinely challenging and which ones are susceptible to heuristic exploitation.
- **Robustness testing against adversarial agents:** While MiCRO has shown resistance to manipulation, further stress-testing it in environments with colluding agents or adversarial pacing strategies would help quantify its resilience under worst-case conditions.
- **Improved benchmarking platforms:** The limitations observed in ANAC suggest a need for new benchmarking standards that include more dynamic and asymmetric domains, strategic uncertainty, and mechanisms for testing robustness. These could be incorporated into future versions of GENIUS or NegMAS.

In conclusion, this thesis highlights not only the enduring value of simple strategies like MiCRO in automated negotiation but also the ongoing need to critically reassess our evaluation frameworks. The goal is not merely to build better agents, but to ensure that the environments used to test them genuinely reflect the complexity and unpredictability of real-world negotiation.

10 Acknowledgements

This master thesis would not have been possible without the generous support of the **JAE Intro Icu 2025** scholarship and the project: *"New Variants of the MiCRO Negotiation Strategy"*.

I would like to express my deepest gratitude to my mentor, **Dr. Dave de Jonge** (`davedejonge@iiia.csic.es`), for his continuous guidance, insightful suggestions, and unwavering encouragement throughout the course of this research. His expertise in automated negotiation and his development of the MiCRO algorithm served as both the foundation and inspiration for this work.

I am especially thankful for the opportunity to contribute to the exciting and evolving field of automated negotiation. The challenges of generalizing the MiCRO strategy to multilateral scenarios and large-scale solution spaces provided an enriching and intellectually stimulating experience that has greatly enhanced my academic and technical skills.

I would also like to thank the staff and colleagues at the **IIIA-CSIC** for creating a supportive and collaborative research environment. Their feedback and discussions were invaluable in shaping the direction of this thesis.

Finally, I extend my sincere appreciation to **Francisco Javier Larrosa Bondia**, my academic supervisor, for his consistent support and valuable advice throughout this project. His clarity, availability, and commitment made a significant difference in the development of this thesis.

References

[1] S. Vente, A. Kimmig, A. Preece, and F. Cerutti, "The current state of automated negotiation theory: A literature review," *arXiv preprint arXiv:2004.02614v2 [cs.AI]*, 2020. [Online]. Available: https://arxiv.org/abs/2004.02614v2.

[2] D. de Jonge, *Introduction to Automated Negotiation*. Barcelona, Spain: self-published, 2025. [Online]. Available: https://www.iiia.csic.es/~davedejonge/intro_to_nego.

[3] T. Baarslag, M. Kaisers, E. H. Gerding, C. M. Jonker, and J. Gratch, "When will negotiation agents be able to represent us? the challenges and opportunities for autonomous negotiators," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 4684–4690. DOI: 10.24963/ijcai.2017/653. [Online]. Available: https://doi.org/10.24963/ijcai.2017/653.

[4] Y. Mohammad, *Yasser mohammad youtube channel*, 2020. [Online]. Available: https://www.youtube.com/@YasserMohammadElmorsy.

[5] B. M. Renting, H. H. Hoos, and C. M. Jonker, "Multi-Agent Meeting Scheduling: A Negotiation Perspective," en,

[6] T. Baarslag, M. J. C. Hendrikx, K. V. Hindriks, and C. M. Jonker, "Learning about the opponent in automated bilateral negotiation: A comprehensive survey of opponent modeling techniques," *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 5, pp. 849–898, Sep. 2016, ISSN: 1573-7454. DOI: 10.1007/s10458-015-9309-1. [Online]. Available: https://doi.org/10.1007/s10458-015-9309-1.

[7] K. Hindriks and D. Tykhonov, "Opponent modelling in automated multi-issue negotiation using bayesian learning," vol. 1, May 2008, pp. 331–338.

[8] C. Williams, V. Robu, E. Gerding, and N. Jennings, "Using gaussian processes to optimise concession in complex negotiations against unknown opponents," *IJCAI International Joint Conference on Artificial Intelligence*, Jan. 2011. DOI: 10.5591/978-1-57735-516-8/IJCAI11-080.

[9] T. Baarslag, K. Hindriks, C. Jonker, S. Kraus, and R. Lin, "The first automated negotiating agents competition (anac 2010)," in Jan. 2012, vol. 383, pp. 113–135, ISBN: 978-3-642-24696-8. DOI: 10.1007/978-3-642-24696-8_7.

[10] D. de Jonge, "An analysis of the linear bilateral anac domains using the MiCRO benchmark strategy," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Appears to be a technical report or preprint version. Source: User uploaded PDF., 2022.

[11] E. Kalai and M. Smorodinsky, "Other solutions to nash's bargaining problem," *Econometrica*, vol. 43, no. 3, pp. 513–518, 1975, ISSN: 00129682, 14680262. [Online]. Available: http://www.jstor.org/stable/1914280 (visited on 06/21/2025).

[12] X.-B. Hu, M. Wang, and E. Di Paolo, "Calculating complete and exact pareto front for multiobjective optimization: A new deterministic approach for discrete problems," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 1088–1101, 2013. DOI: 10.1109/TSMCB.2012.2223756.

[13] S. Kakimoto and K. Fujita, "Estimating pareto fronts using issue dependency for bilateral multi-issue closed nonlinear negotiations," in *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, 2014, pp. 289–293. DOI: 10.1109/SOCA.2014.30.

[14] X. Luo, N. R. Jennings, N. Shadbolt, H.-f. Leung, and J. H.-m. Lee, "A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments," *Artificial Intelligence*, vol. 148, no. 1, pp. 53–102, 2003, Fuzzy Set and Possibility Theory-Based Methods in Artificial Intelligence, ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(03)00041-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370203000419.

[15] R. Lin, S. Kraus, T. Baarslag, D. Tykhonov, K. Hindriks, and C. M. Jonker, "Genius: An integrated environment for supporting the design of generic automated negotiators," *Computational Intelligence*, vol. 30, no. 1, pp. 48–70, 2014, ISSN: 1467-8640. DOI: 10.1111/j.1467-8640.2012.00463.x. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8640.2012.00463.x.

[16] T. Baarslag, K. Fujita, E. H. Gerding, *et al.*, "Evaluating practical negotiating agents: Results and analysis of the 2011 international competition," *Artificial Intelligence*, vol. 198, pp. 73–103, 2013, ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2012.09.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370212001105.

[17] R. Aydoğan, D. Festen, K. V. Hindriks, and C. M. Jonker, "Alternating offers protocols for multilateral negotiation," in *Modern Approaches to Agent-based Complex Automated Negotiation*, K. Fujita, Q. Bai, T. Ito, *et al.*, Eds. Cham: Springer International Publishing, 2017, pp. 153–167, ISBN: 978-3-319-51563-2. DOI: 10.1007/978-3-319-51563-2_10. [Online]. Available: https://doi.org/10.1007/978-3-319-51563-2_10.

[18] A. Casali, P. Pilotti, and C. Chesñevar, "Assessing communication strategies in argumentation-based negotiation agents equipped with belief revision1," *Argument & Computation*, vol. 7, no. 2-3, pp. 175–200, 2016. DOI: 10.3233/AAC-160012. eprint: https://doi.org/10.3233/AAC-160012. [Online]. Available: https://doi.org/10.3233/AAC-160012.

[19] I. RAHWAN, S. D. RAMCHURN, N. R. JENNINGS, P. McBURNEY, S. PARSONS, and L. SONENBERG, "Argumentation-based negotiation," *The Knowledge Engineering Review*, vol. 18, no. 4, pp. 343–375, 2003. DOI: 10.1017/S0269888904000098.