Noname manuscript No.
(will be inserted by the editor)

Using Game Description Language for Mediated Dispute Resolution

Dave de Jonge · Tomas Trescak · Carles Sierra · Simeon Simoff · Ramon López de Mántaras

the date of receipt and acceptance should be inserted later

Abstract Mediation is a process in which two parties agree to resolve their dispute by negotiating over alternative solutions presented by a mediator. In order to construct such solutions, the mediator brings more information and knowledge, and, if possible, resources to the negotiation table. In order to do so, the mediator faces the challenge of determining which information is relevant to the current problem, given a vast database of knowledge. The contribution of this paper is the automated mediation machinery to resolve this issue. We define the concept of a Mediation Problem and show how it can be described in Game Description Language (GDL). Furthermore, we present an algorithm that allows the mediator to efficiently determine which information is relevant to the problem and collect this information from the negotiating agents. We show with several experiments that this algorithm is much more efficient than the naive solution that simply takes all available knowledge into account.

D. de Jonge
School of Computing, Engineering and Mathematics,
Western Sydney University, Australia
E-mail: d.dejonge@westernsydney.edu.au,

T. Trescak
School of Computing, Engineering and Mathematics,
Western Sydney University, Australia
E-mail: t.trescak@westernsydney.edu.au,

C. Sierra
IIIA-CSIC, Barcelona, Spain
E-mail: sierra@iiia.csic.es,

S. Simoff
School of Computing, Engineering and Mathematics,
Western Sydney University, Australia
E-mail: s.simoff@westernsydney.edu.au

R. López de Mántaras
IIIA-CSIC, Barcelona, Spain
E-mail: mantaras@iiia.csic.es

1 Introduction and Motivation

Dispute resolution is a complex process that depends on the will of involved parties to reach consensus. Any involved agent would only accept a solution if that solution allows it to partially or completely fulfill its own individual goals with the available resources. In many cases, such negotiation depends on searching for alternative solutions which requires extensive knowledge about the disputed matter. If such information is not available to the negotiating parties then negotiation fails. In such cases a mediator can assist both parties to come to an agreement.

This paper presents a logic-based mediation system inspired by previous work in the fields of mediation and argumentation-based negotiation. It builds on some ideas presented in (Parsons et al, 1998). In that work agents had all the knowledge and resources needed to resolve their dispute—a relatively strong assumption in the context of real-world negotiations. In the real world, lacking knowledge or resources may lead to unsuccessful negotiations. In many cases, such knowledge or even alternative resources may be available, but agents are not aware of them.

This paper proposes a mediator that possesses extensive knowledge. The mediator also has access to various resources that may help to resolve the dispute. Using this knowledge and resources, as well as knowledge and resources obtained from the negotiating agents, the mediator creates alternative solutions, which become subject to further negotiation. We assume this mediator is neutral and considered trustworthy by all interested parties.

Although the problem we aim to tackle is inspired by (Parsons et al, 1998) we are taking an entirely different approach for the implementation of the mediation

algorithm. Key to our approach is the observation that for any mediation scenario the underlying problem that one aims to solve is essentially a game, since there are several parties with conflicting goals. This means that Mediation Problems can be described using Game Description Language (GDL) (Love et al, 2006).

Most works in the field of Automated Mediation represent the domain of interest either in some ad-hoc format, such as a feature-vector representation, or using an ontological language. Ontological languages are very useful to describe facts about a static world, but are less suitable to describe more complex planning domains in which one needs to specify the pre-conditions and post-conditions of the agents' actions. We argue that for such domains GDL is more natural choice. GDL allows us to describe domains in which agents' utilities depend on sequences of actions which are subject to complex constraints, and in which different agents have different goals. Furthermore, it is an established language already used by many researchers in the field of General Game Playing and an extensive code base that facilitates rapid development of algorithms is publicly available.

From a more formal point of view, GDL has te advantage that its restricted rule-based syntax in combination with 'minimal model semantics' and 'negation-by-failure' makes it easy to determine relationships between the formulas in the agents' knowledge bases. This is essential in order to determine which knowledge is relevant to the problem and which knowledge is not. Furthermore, a set of GDL rules is always consistent, and determining whether a formula is true or not, given a set of rules, is decidable.

General Game Playing (GGP) deals with the problem of implementing agents that can play any kind of game, of which the rules are only known at run-time. A General Game Playing agent is able to interpret the rules of a game at run-time and devise a strategy for it without any human intervention. One important difference between GGP and our Mediation Problems is that in GGP it is assumed that the rules of the game are known to all players. In our case, however, we assume that each agent only has partial knowledge of the game's rules, so the agents need to exchange knowledge. Since this knowledge may be of strategic importance, the players may not be willing to share all knowledge with each other, and therefore require a neutral mediator to find a solution for them.

We note that the mediator faces two major challenges:

1. The mediator needs to determine which information to request from the players in order to fill the gaps in its knowledge.
2. Given this information, the mediator needs to search for a suitable action plan that is acceptable to both players.

Naively, one could argue that the mediator can get rid of the first challenge by simply requesting the players to provide all knowledge they have about the given problem. The problem with this approach, however, is that in many real-world situations agents may possess very large databases of knowledge that allow them to deal with many different problems, and therefore the majority of the information in such databases is irrelevant to the problem they are currently trying to solve.

Instead, the mediator needs to apply intelligent reasoning to carefully select which information should be requested from the negotiating agents, and which information should be ignored. After all, taking into account irrelevant information would unnecessarily increase the size of the mediator's search space. Moreover, agents may prefer not to give away too much information for privacy reasons, and instead prefer to only reveal those pieces of knowledge that are essential to solving the problem.

In this paper, we only focus on the first of these two challenges. Our proposed mediation algorithm systematically works backwards from the agents' final goals, and in this way determines step by step which knowledge is required to obtain those goals. For the second challenge one can apply existing techniques from Planning, GGP, CBR, or Automated Mediation, so we will largely ignore this.

We should note that in this paper, although we are dealing with imperfect information, we describe our problems in pure GDL, rather than its extension GDL-II, even though GDL-II was specifically designed for games with imperfect information (Thielscher, 2010). The reason for this is that in our case the imperfect information is of an inherently different nature. GDL-II is used to define games in which the players do not have perfect information about the *state* of the game (e.g. they cannot see each others' playing cards), while the rules of the game are perfectly known to all players. In this paper, on the other hand, the game itself is a game of perfect information, but the players do not have complete information about the *rules* of the game. Therefore, we can restrict ourselves to GDL.

The rest of this paper is organized as follows. In the next section, we summarize existing work related to Automated Mediation, Argumentation-Based Negotiation and Game Description Language. In Section 3 we introduce the Home Improvement problem, which is a toy-world problem that introduces the main ideas behind mediation. In Section 4 we formally define the concept of a Mediation Game, which is a type of game

involving two competing agents and a neutral mediator. In Section 5 we give a short introduction to GDL. Then, in Section 6 we are finally ready to formalize the problem we aim to solve (a Mediation Problem). In Section 7 we give a full description of the Home Improvement example in GDL. In Section 8 we present our mediation algorithm and in Section 9 we present the results of our experiments. In Section 10 we summarize our conclusions and finally, in Section 11, we discuss future work.

2 Previous Work

Many approaches to mediation focus mainly on how to find solutions, given the domain knowledge, but pay less attention to the question of how the knowledge about the current problem is acquired. They assume a closed world in which every participant has all information about the problem to solve, and only about that specific problem. In such cases, providing the mediator with the relevant information to solve the problem is trivial. They often do not take into account that agents in reality may have large knowledge bases containing many pieces of information that are actually irrelevant to the current problem, and that an agent alone may not be able to determine whether a specific piece of information is relevant to the problem or not.

Some Case-Based Reasoning (CBR) approaches do assume the mediator has access to a large database of knowledge, but this is used only to retrieve solutions to similar problems (Baydin et al, 2011). In our case, on the other hand, the knowledge bases contain the information that defines a planning problem, rather than the solutions to such problems. These CBR approaches can still be used perfectly in combination with our approach, but we focus on a different aspect of mediation (gathering relevant information, rather than finding solutions).

The `MEDIATOR` (Kolodner and Simpson, 1989) focused on case-based reasoning as a single-step for finding a solution to a dispute resolution problem. The mediation process was reduced to a one-step case-based inference, aimed at selecting an abstract "mediation plan". The work did not consider the value of the actual dialog with the mediated parties. The `PERSUADER` (Sycara, 1991) deployed mechanisms for problem restructuring that operated over the goals and the relationships between the goals within the game theory paradigm, applied to labor management disputes. To some extent this work is a precursor of another game-theoretic approach to mediation, presented in (Wilkenfeld et al, 2004) and the interest-based negotiation approach in (Rahwan et al, 2009). Notable are recent game-theoretic computational mediators `AutoMed` (Chalamish and Kraus, 2012) and `AniMed` (Lin et al, 2011) for multi-issue bilateral negotiation under time constraints. They operate within known solution space, offering either specific complete solutions (`AutoMed`) or incremental partial solutions (`AniMed`). In these cases the domains were described by sets of feature-vectors, and the agents' preferences over these vectors. Each negotiating agent had perfect knowledge about the domain (except its opponent's utility function). The only information exchanged between the negotiators and the mediator, was partial information about the negotiators' preference relations. The 'Curious Negotiator' (Simoff and Debenham, 2002) was proposed as a general infrastructure for mediation in which knowledge gathering plays an essential. However, it did not specify any detailed algorithms or mention any language to define the mediation problems. The `Family_Winner` (Bellucci and Zeleznikow, 2005) manipulative mediator aimed at modifying the initial preferences of the parties in order to converge to a feasible and mutually acceptable solution. In this work human users introduced their knowledge about the domain through a user interface, so the mediator itself did not have to worry about filtering out any irrelevant information, as this was already done by the humans.

In real settings it is not sufficient to only have information about negotiation issues to derive the outcome preferences (Visser et al, 2011). An exploratory study (Schei and Rognes, 2003) of a multiple (three) issue negotiation setting suggests the need for developing integrative (rather than position-based) negotiation processes which take into account information about the motivational orientation of negotiating parties. Incorporation of information beyond negotiation issues has been the focus of a series of works related to information-based agency (Debenham, 2004; Debenham and Simoff, 2006; Sierra and Debenham, 2007). Value-based argumentation frameworks (Bench-Capon, 2003), interest-based negotiation (Rahwan et al, 2009) and interest-based reasoning (Visser et al, 2011) considers the treatment of any kind of motivational information that leads to a preference in negotiation and decision making.

The field of General Game Playing is a relatively new field. Although it already existed earlier, it only started to draw widespread attention with the development of GDL and the first edition of the annual GGP competition, at the AAAI Conference in 2005 (Genesereth et al, 2005). Common search techniques that are applied by many players are minimax (von Neumann, 1959) search, alpha-beta pruning (Knuth and Moore, 1975) and Monte Carlo Tree Search (MCTS) (Koc-

sis and Szepesvári, 2006; Genesereth and Thielscher, 2014). Although GDL was mainly intended to be used for GGP, it has also been used to represent knowledge in other fields, such as Automated Negotiations (Jonge and Zhang, 2016).

3 The Home Improvement Problem

In this section we give an informal description of the problem that we will use as a running example througout this paper. It is an adaptation of the home improvement problem from (Parsons et al, 1998).

In this example, agent α is trying to hang a picture on the wall. Agent α knows that to hang a picture it needs a nail and a hammer. Agent α only has a screw and a hammer, but it knows that agent β owns a nail. Agent β is trying to hang a mirror on the wall. β knows that it needs a nail and a hammer to hang the mirror, but β currently possesses only a nail and a screw, and also knows that α has a hammer. Luckily, there is also a mediator μ which owns a screwdriver and knows that a mirror can be hung using a screw and a screwdriver. This mediator does not have any goals of its own, other than that it desires α and β to both achieve their respective goals.

Since agents α and β have individual interests we assume they are not willing to share their information directly with each other. We do, however, assume that they both trust the mediator and are therefore willing to share information with the mediator.

The difference with the example in (Parsons et al, 1998) is that here the mediator owns some of the knowledge and resources needed to resolve the dispute and not just the agents. This reflects reality, when clients seek advice of an expert to resolve their problem.

The task of the mediator can be split up into two subtasks. Firstly, the mediator needs to gather all the relevant information necessary to find a solution. Secondly, once the mediator has all that information, he needs to use it to find the solution. In this paper we focus on the first subtask, as the second subtask can be considered a standard planning problem. The difficulty in the first task lies in the fact that the mediator and the agents do not only have information about the current problem they are aiming to solve, but also about many other topics, most of which are unrelated to the Home Improvement problem. Furthermore, for any given piece of information it may not be obvious to the agent owning that information whether it is relevant to the problem or not, because it depends on certain knowledge owned by another agent. Therefore, the mediator needs an intelligent algorithm to identify the relevant pieces of information and collect them from the other agents.

4 Mediation Games

In the previous section we have informally described an example of a Mediation Problem. Before we can give a formal definition of a Mediation Problem we first define what we call a *Mediation Game*, which is very similar to the traditional notion of a game, except that it involves a neutral mediator. Then, we will use this in Section 6 to define a *Mediation Problem* as a Mediation Game for which each agent only knows part of the rules.

It is important to understand that the concepts of a Mediation Game and a Mediation Problem are only used to describe the problem that the agents aim to solve. They do not describe the actual process of mediation.

A Mediation Game consists of two players and a mediator. The players and the mediator each have a set of actions to their disposal which they can execute to change the state of the world. While the two players each aim to maximize their individual utility values, the mediator is neutral and only aims to find Pareto-optimal solutions.

Definition 1 (Mediation Game) A Mediation Game consists of:

- A set of **agents:** $Ag = \{\alpha, \beta, \mu\}$, where α and β are also called **players** and μ is called the **mediator**.
- Three non-empty sets of **actions**: $\mathcal{A}_\alpha, \mathcal{A}_\beta, \mathcal{A}_\mu$ (one for each agent) which are mutually disjoint.
- A finite, non-empty set of **states** W.
- An **initial state** $w_1 \in W$.
- A non-empty set of **terminal states** $T \subset W$.
- For each player $i \in \{\alpha, \beta\}$ a **utility function**: $U_i : T \to [0, 100]$ that maps each terminal state to a utility value between 0 and 100.
- For each agent a **legality function**: $L_i : (W \setminus T) \to 2^{\mathcal{A}_i}$ (with $i \in \{\alpha, \beta, \mu\}$) that defines for each non-terminal state which actions are legal for that agent.
- An **update function** $u : (W \setminus T) \times \mathcal{A}_\alpha \times \mathcal{A}_\beta \times \mathcal{A}_\mu \to W$ that maps each non-terminal state and action profile to a new state.

The definition of a Mediation Game is very similar to a standard game-theoretical definition of an extensive form game for 3 players. The main difference is that in our case the mediator does not have its own utility function.

It seems worthy to remark here that according to this definition utility is only defined on the terminal states. This means that the agents only care about *which* terminal state is achieved, but do not care about *how* it is achieved. This assumption may be different from the assumptions made in some planning domains,

but it is standard practice in GGP.[1] Also note that utility is defined to be a value between 0 and 100. This is because GDL does not define fractional numbers, so utility must be an integer.

Definition 2 (Binary Mediation Game) We say a Mediation Game is binary or that a Mediation Game has binary utility functions, if for each terminal state $t \in T$ and each player $i \in \{\alpha, \beta\}$ we have either $U_i(t) = 0$ or $U_i(t) = 100$

Informally, a mediation game being binary means that for each player the outcome is either success or failure, but there is nothing in between.

Definition 3 (Action Profile) An action profile $\mathbf{a}$ is a triple consisting of one action for each agent.

$\mathbf{a} = (a_\alpha, a_\beta, a_\mu) \in \mathcal{A}_\alpha \times \mathcal{A}_\beta \times \mathcal{A}_\mu$

An action profile $\mathbf{a}$ is **legal** in state w iff:

$\mathbf{a} \in L_\alpha(w) \times L_\beta(w) \times L_\mu(w)$

Definition 4 (History) A history $\mathcal{H}$ is a sequence:

$(w_1, \mathbf{a}_1, w_2, \mathbf{a}_2 \dots w_n)$

where each $\mathbf{a}_t$ is an action profile and each w_t is a state, such that for each $t \geq 1$ we have that $\mathbf{a}_t$ is legal in w_t and we have:

$w_{t+1} = u(w_t, \mathbf{a}_t)$.

A **terminal history** is a history for which the last state w_n is a terminal state.

That is: each state w_{t+1} in the sequence is the result from the agents playing the action profile $\mathbf{a}_t$ in the previous state w_t, and each action profile $\mathbf{a}_t$ consists only of actions that are legal for the respective agents in the state w_t.

Definition 5 (Plan) A plan is a finite sequence of action profiles:

$(\mathbf{a}_1, \mathbf{a}_2, \dots \mathbf{a}_n)$

Given an initial state w_1 a plan is a **legal plan** if there exists a sequence of states $(w_1, w_2, \dots w_n)$ such that the sequence $(w_1, \mathbf{a}_1, w_2, \mathbf{a}_2 \dots w_n)$ is a history. Note that if such a sequence of states exists it is unique. A legal plan is a **terminal plan** if the corresponding history is a terminal history. A legal plan is **cycle-free** if the sequence of states does not contain any state more than once; i.e.: for all $j, k \in [1, \dots n]$ if $j \neq k$ then $w_j \neq w_k$.

[1] Technically speaking, GDL does allow you to define utility over non-terminal states, but these utility values do not really have any meaning, as in the end the utility of the terminal state is the only thing that 'counts'.

Lemma 1 *For any Mediation Game G the set of cycle-free plans is finite.*

Proof This follows directly from the restriction that the set of world states W of G must be finite.

We note that each player has its own individual utility function, and that both players are considered selfish: each player is only interested in realizing a terminal state that maximizes its own utility function. Neither of the two players is interested in the value of other player's utility, and therefore the players' intentions may be conflicting. We do assume, however, that the mediator is willing to help executing a plan that is acceptable to both players.

The presence of a mediator has two advantages: firstly, the mediator may have knowledge the agents do not have. Secondly, the agents may share their knowledge with the mediator. This allows the agents to find plans using their combined knowledge, without having to reveal their knowledge to one another. The mediator is assumed to be a trusted party to which the agents can reveal information in confidence. The goal of the mediator is to propose a legal and terminal plan such that each player is willing to execute its part of that plan.

For a given mediation game G we can formalize the agents' objectives by defining a preference relation $\succeq_i$ for each agent $i \in \{\alpha, \beta, \mu\}$ over the set of terminal plans. Let p and q be two terminal plans and w_p and w_q the terminal states of their respective corresponding histories. Then the agents have the following preference relations:

$$
\begin{aligned}
p \succeq_\alpha q \quad &\Leftrightarrow \quad U_\alpha(w_p) \geq U_\alpha(w_q) \\
p \succeq_\beta q \quad &\Leftrightarrow \quad U_\beta(w_p) \geq U_\beta(w_q) \\
p \succeq_\mu q \quad &\Leftrightarrow \quad U_\alpha(w_p) \geq U_\alpha(w_q) \ \wedge \ U_\beta(w_p) \geq U_\beta(w_q)
\end{aligned}
$$

Of course, one could define a utility function for the mediator simply as the sum of the two utility functions of the players. However, we are intentionally not doing that. The first reason for this is that in real-world scenarios you may not always have *exact* utility functions. Often it is more reasonable to just assume that an agent has a preference order over the possible outcomes, without being able to assign exact quantitative values to them. Therefore, whenever we have a state w_a with utility 100 and a state w_b with utility 50 we should interpret this merely as "w_a is better than w_b" rather than as the stronger statement "w_a is twice as good as w_b". This means that it does not really make sense to take the sum of the utility values of two different players, since their precise values do not have any real meaning. The second reason is that, even if the utility functions are exact, a player may lie to the mediator

about its true utility in order to enforce a better outcome for himself (a very common strategy in real-life negotiations). Again, this means that it does not really make sense for the mediator to evaluate the sum of the players' utility functions.

Definition 6 (Solution) A solution to a Mediation Game is a terminal, cycle-free plan such that its resulting terminal state t yields positive utility for both players: $U_\alpha(t) > 0$ and $U_\beta(t) > 0$

A Mediation Game does not necessarily have a solution, but it only makes sense to apply mediation to Mediation Games that do have at least one solution.

The restriction that a solution be cycle-free is not strictly necessary. However, it does not make much sense to propose any solutions that do contain cycles, since any cycle could trivially be removed from the solution. On the other hand, the restriction of being cycle-free has the advantage that it ensures that the set of solutions is finite.

5 Game Description Language

In this section we give a short introduction to GDL. For more details we refer to (Love et al, 2006).

GDL is a logical language designed to describe game rules. Specifically, it was invented for the research area of General Game Playing. Although our Mediation Problems may just as well be specified in any other language we feel that GDL is a natural choice, because a Mediation Problem is a planning problem in which the agents have conflicting goals, and therefore it is essentially a game. Moreover, a big advantage of GDL is that it is designed specifically to be used in actual implementations (rather than mere theoretical descriptions) and an extensive Java framework is readily available that allows us to quickly implement working algorithms.

5.1 Syntax

Predicates in GDL are composed of relation symbols, constants, variables, function symbols and the logical connectives $\wedge$, $\neg$ and $\rightarrow$. GDL is similar to Datalog (Ceri et al, 1989), but it gives special meaning to the following relation symbols:[2] *init*, *true*, *next*, *legal*, *goal*, *terminal*, *does* and *distinct*, which are related to games.

Definition 7 (Rule) A GDL **rule** is an expression of the following form:

$$s_1 \wedge s_2 \wedge \ldots s_n \rightarrow h$$

where each s_i is a positive or negative literal, and h is a positive literal. The atom h is called the **head** of the rule and the s_i's are called the **subgoals** of the rule. The conjunction of subgoals is called the **body** of the rule. The body of a rule may be an empty conjunction, in which case the rule is also referred to as a **fact**, and is denoted as: $\rightarrow h$.

In GDL the relation symbols *true*, *does*, and *distinct* are not allowed to appear in the head of any rule, while the other keywords *goal*, *terminal*, *legal*, *next*, and *init* are not allowed to appear in the body of a rule. Apart from these eight keywords, GDL literals may also contain user-defined relation symbols. An expression is said to be *ground* if it does not contain any variables.

Ground atoms with the relation symbol *true* are called **base propositions**. In this paper, if $\mathbf{a} = (a_\alpha, a_\beta, a_\mu)$ is an action profile of some (mediation) game G, then we use the notation $D_\mathbf{a}$ to denote the following set of propositions:

$$D_\mathbf{a} = \{does(\alpha, a_\alpha), does(\beta, a_\beta), does(\mu, a_\mu)\}.$$

We now define the notion of a proposition s being *derivable*. Informally, if s is derivable from $(\mathcal{R}, \mathcal{B}, \mathbf{a})$ it means that, if all propositions in $\mathcal{B}$ and $D_\mathbf{a}$ are considered 'true' then, according to the rules in $\mathcal{R}$, the proposition s must also be considered 'true'.

Definition 8 (Derivable) Let $\mathcal{R}$ be a set of rules, $\mathcal{B}$ a set of base propositions, and $\mathbf{a}$ an action profile. Then we say that a ground atom s is derivable from $(\mathcal{R}, \mathcal{B}, \mathbf{a})$ iff either $s \in \mathcal{B} \cup D_\mathbf{a}$, or all of the following are true:

- There is a rule $\phi \in \mathcal{R}$ and a substitution θ such that $\phi[\theta]$ is ground, and
- s is the head of $\phi[\theta]$, and
- all positive subgoals of $\phi[\theta]$ are derivable from $(\mathcal{R}, \mathcal{B}, \mathbf{a})$, and
- none of the negative subgoals of $\phi[\theta]$ is derivable from $(\mathcal{R}, \mathcal{B}, \mathbf{a})$.

We see that the notion of being derivable is defined recursively. This recursion may terminate either because the rule ϕ is a fact (i.e. has an empty body), or because its subgoals contain the relation symbol *does* or *true*. This recursion is not always guaranteed to terminate. If the recursion does not terminate because it involves a cycle of positive literals, then all literals in this cycle are considered non-derivable. Note that it is not possible that such a cycle contains any negative literals, because a valid GDL description must satisfy a property called *stratification*. For more information about this we refer to (Love et al, 2006), because we do not want to get into the technical details of GDL here.

[2] GDL defines more relations symbols, but we will not discuss them here because they are not relevant for this paper.

5.2 Semantics (informal)

We here give an informal description of the semantics of GDL. A formal description is given in the next subsection. We should note that these semantics are defined for *games* rather than *Mediation Games*. However, as we noted before a Mediation Game is just a game for three players, except that the mediator does not have a utility function. Therefore, we only need to add the syntactical restriction that the first argument of the *goal* relation cannot be the mediator. With this restriction the semantics of GDL for games directly carries over to Mediation Games.

In order to link a game to a set of GDL rules we need a *valuation function* V, which maps every state w of G injectively to a set of base propositions. That is, each state is uniquely identified with a finite set of atoms of the form $true(p)$, where p can be any ground term. For example, in Tic-Tac-Toe the state in which the the center cell contains the marker X and the left upper cell contains the marker O could be represented as:

$$V(w) = \{\ true(cell(2,2,X))\ \ ,\ \ true(cell(1,1,O))\ \}$$

A game can then be defined completely as a set of GDL rules. For example, if the game description contains the following rule:

$$true(p) \wedge does(\alpha, a) \rightarrow next(q)$$

it means that if the game is in a state w for which $true(p) \in V(w)$ and player α plays action a then in the next round the game will be in a state w' for which $true(q) \in V(w')$ holds.

In Tic-Tac-Toe, for example, one could have the rule:

$$does(\alpha, mark(2,2)) \rightarrow next(cell(2,2,X))$$

meaning "If α plays the action $mark(2,2)$ then in the next turn the cell with coordinates (2,2) will contain an X". Similarly:

$$true(p) \rightarrow terminal$$

means that any state w for which $true(p) \in V(w)$ holds is a terminal state.

The terms in these rules may contain variables, which we denote with a question mark. For example, the rule

$$does(\alpha, mark(?m, ?n)) \rightarrow next(cell(?m, ?n, X)$$

means that for any pair of coordinates (m, n) it is true that if player α plays the action $mark(m, n)$ then in the next state the cell with those coordinates will be marked with an X. The fact

$$\rightarrow init(p)$$

means that for the initial state w_1 we have that $true(p) \in V(w_1)$ holds.

$$true(p) \rightarrow legal(\alpha, a)$$

means that for any state in which $true(p) \in V(w)$ holds it is legal for player α to play the move a.

$$true(p) \rightarrow goal(\alpha, 100)$$

means that in any state w for which $true(p) \in V(w)$ holds α receives a utility value of 100.

5.3 Semantics (formal)

We will now present the semantics of GDL in a more formal way.

Definition 9 (Valuation Function) Given a game G and a set of Base propositions $\mathcal{B}$, a valuation function is an injective map that maps each world state w of G to a finite set of base propositions $V(w) \subset \mathcal{B}$.

$$V : W \rightarrow 2^{\mathcal{B}}$$

Given a game G, a valuation function V for that game, a state w and an action profile $\mathbf{a} = (a_\alpha, a_\beta, a_\mu)$ the entailment of a ground atom for which its relation symbol is a GDL keyword is defined as follows:

- $G, V \vDash_{(w,\mathbf{a})} true(p)$ iff $true(p) \in V(w)$
- $G, V \vDash_{(w,\mathbf{a})} does(i, b)$ iff $a_i = b$ with $i \in \{\alpha, \beta, \mu\}$
- $G, V \vDash_{(w,\mathbf{a})} distinct(p, q)$ iff $p \neq q$
- $G, V \vDash_{(w,\mathbf{a})} init(p)$ iff $true(p) \in V(w_1)$
- $G, V \vDash_{(w,\mathbf{a})} next(p)$ iff $true(p) \in V(u(w, \mathbf{a}))$
- $G, V \vDash_{(w,\mathbf{a})} legal(i, b)$ iff $b \in L_i(w)$ with $i \in \{\alpha, \beta, \mu\}$
- $G, V \vDash_{(w,\mathbf{a})} terminal$ iff $w \in T$
- $G, V \vDash_{(w,\mathbf{a})} goal(i, n)$ iff $U_i(w) = n$ with $i \in \{\alpha, \beta\}$

Here, all terms are ground. The equality $a_i = b$ means that the two terms are syntactically equal, and $p \neq q$ means that p and q are syntactically different. Note that the entailment of $true(p)$, $legal(i, b)$, $terminal$, and $goal(i, n)$ does not depend on the action profile, so in those cases we may write $G, V \vDash_w$ instead of $G, V \vDash_{(w,\mathbf{a})}$. In the case of *distinct* and *init* their entailment does not even depend on the state, so we may even write $G, V \vDash$.

Logical connectives are defined as usual:

- $G, V \vDash_{(w,\mathbf{a})} \neg s$ iff $G, V \nvDash_{(w,\mathbf{a})} s$
- $G, V \vDash_{(w,\mathbf{a})} s_1 \wedge s_2$ iff $G, V \vDash_{(w,\mathbf{a})} s_1$ and $G, V \vDash_{(w,\mathbf{a})} s_2$

5.4 Negation-by-Failure

The fact that the head of a GDL rule must always be a *positive* atom, is because negation is to be interpreted as *negation-by-failure*. This means that an atom s is considered true only if it is derivable, and is considered false if it is not derivable. Equivalently, a literal $\neg s$ is considered true if and only if s is not derivable. As a consequence, we can never have that s and $\neg s$ are true at the same time, so a set of GDL rules can never be inconsistent.

5.5 Irrelevant Rules

In the rest of this paper we assume that agents may have knowledge bases that contain much more information than just information about the current game they are playing. In order to formalize this we introduce the notion of a GDL rule being *irrelevant*. This concept does not appear in the existing literature, because normally in GGP one is not interested in such rules.

Informally, a rule ϕ is irrelevant in $\mathcal{R}$, if it does not affect (either directly or indirectly) the goals or the termination conditions of the game that $\mathcal{R}$ is supposed to describe. This means that ϕ may just as well be removed from $\mathcal{R}$, because $\mathcal{R} \setminus \{\phi\}$ still forms a proper description of that game.

Definition 10 (Relevant) Let $\mathcal{R}$ be a set of GDL rules. An atom s is said to be **relevant** in $\mathcal{R}$ iff at least one of the following three conditions is true:

- s contains the relation symbol 'terminal' or 'goal', or
- There is a rule $\phi \in \mathcal{R}$ such that either s or $\neg s$ is unifiable with a subgoal of ϕ, and the head of ϕ is relevant in $\mathcal{R}$, or
- s is of the form $init(t)$ or $next(t)$, while $true(t)$ is relevant in $\mathcal{R}$,
- s is of the form $legal(i, a)$, while $does(i, a)$ is relevant in $\mathcal{R}$.

A rule $\phi \in \mathcal{R}$ is said to be relevant in $\mathcal{R}$ iff its head is relevant in $\mathcal{R}$. An atom or rule is said to be **irrelevant** in $\mathcal{R}$ iff it is not relevant in $\mathcal{R}$.

Note that in this definition for any given atom s no more than one of these conditions can ever be true at the same time, so we could equivalently require that *exactly* one of them be true. We will sometimes simply say that a rule or atom is (ir)relevant without specifying the set of rules, if the set of rules is clear from the context.

The following definition will be useful to prove some of the lemmas in the rest of this paper.

Definition 11 (Relevance Depth) If s is a relevant atom, then we define the relevance depth $rd(s)$ of s as follows: if s contains the relation symbol 'terminal' or 'goal' then $rd(s) = 0$. If s is of the form $next(t)$ or $init(t)$ then $rd(s) = rd(true(t))$. If s if of the form $legal(i, a)$ then $rd(s) = rd(does(i, a))$. Otherwise, if s or $\neg s$ is unifiable with a subgoal of some rule with relevant head h, then $rd(s) = rd(h)+1$ (if there is more than one such rule then take the one for which $rd(h)$ is minimal).

Lemma 2 *If a rule ψ is irrelevant in $\mathcal{R}$, then it is irrelevant in any subset of $\mathcal{R}$.*

Proof This follows immediately from the definition. If h is the head of ψ, and all conditions of Def. 10 are false for h, then they are still false if any number of rules is removed from $\mathcal{R}$.

Lemma 3 *If ϕ is relevant in $\mathcal{R}$, and ψ is irrelevant in $\mathcal{R}$, then ϕ is also relevant in $\mathcal{R} \setminus \{\psi\}$*

Proof Let h be the head of ϕ. The proof goes by induction on relevance depth of h. If h contains the relation symbol 'terminal' or 'goal' then the lemma is trivial, so it holds for $rd(h) = 0$. Now assume the lemma holds for all rules for which the head s has $rd(s) < n$, and that we have $rd(h) = n$. We first prove the lemma for the case h is not of the form $init(t)$, $next(t)$ or $legal(i, a)$. Since h is relevant in $\mathcal{R}$, there must be a relevant rule $\phi' \in \mathcal{R}$ such that h or $\neg h$ is unifiable with one of the subgoals of ϕ'. By definition of relevance depth, we know that for the head h' of ϕ' we have $rd(h') < n$, and therefore, by the induction hypothesis, we know that h' and ϕ' are relevant in $\mathcal{R} \setminus \{\psi\}$. This, however, means that h and ϕ are also relevant in $\mathcal{R} \setminus \{\psi\}$, by Def. 10. We can repeat this argument for $true(t)$ or $does(i, a)$ instead of h, and therefore conclude that the lemma also holds if h is of the form $init(t)$ or $next(t)$ or $legal(i, a)$.

Proposition 1 *Let $\mathcal{R}$ be a set of rules, and let S be the subset of $\mathcal{R}$ consisting of exactly all the rules that are irrelevant in $\mathcal{R}$. Then all rules $\phi \in \mathcal{R} \setminus S$ are relevant in $\mathcal{R} \setminus S$.*

Proof Let us write $T = \mathcal{R} \setminus S$, and $S = \{\psi_1, \psi_2, \ldots \psi_n\}$. We need to prove that all $\phi \in T$ are relevant in T. Clearly, all $\phi \in T$ are relevant in $\mathcal{R}$. Now, note that according to Lemma 3 we have that all $\phi \in T$ are relevant in $\mathcal{R} \setminus \{\psi_1\}$. Furthermore, because of Lemma 2, we know that ψ_2 is irrelevant in $\mathcal{R} \setminus \{\psi_1\}$. This means that we can use Lemma 3 again, to prove that all $\phi \in T$ are relevant in $\mathcal{R} \setminus \{\psi_1, \psi_2\}$. It is now clear that if we continue applying these two lemmas we finally obtain that all $\phi \in T$ are relevant in T.

Lemma 4 *Let ψ be an irrelevant rule in $\mathcal{R}$. Then for any relevant ground atom s, state w and action profile $\mathbf{a}$ we have:*

$$s \text{ is derivable from } (\mathcal{R}, V(w), \mathbf{a}) \Leftrightarrow s \text{ is derivable from } (\mathcal{R} \setminus \{\psi\}, V(w), \mathbf{a}) \quad (1)$$

Proof For this proof we define the derivation depth $dd(s)$ of a ground atom s as follows: $dd(s) = 0$ iff $s \in \mathcal{B} \cup D_{\mathbf{a}}$. Otherwise, if there is a rule ϕ as in Def. 8, then we define $dd(s)$ as $dd(s^*) + 1$ where s^* is the subgoal of ϕ with the highest derivation depth. If there is more than one such rule, then take the one that yields the highest value of $dd(s)$. The proof now goes by induction on $dd(s)$. Clearly, if $dd(s) = 0$, s is derivable regardless of whether ψ is in $\mathcal{R}$. Now suppose that the lemma is true for any relevant ground atom s' with $dd(s') < n$.

Left-to-right: if $dd(s) = n$ and s is derivable from $(\mathcal{R}, V(w), \mathbf{a})$, then there is some rule $\phi \in \mathcal{R}$ that satisfies the conditions of Def. 8. However, this means that the head of ϕ is unifiable with s, which is a relevant atom and therefore ϕ is relevant. This means $\phi \neq \psi$, which means $\phi \in \mathcal{R} \setminus \{\psi\}$. Since for all subgoals s' of ϕ we have $dd(s') < n$, and all these s' are relevant because ϕ is relevant, we know that all positive subgoals and none of the negative subgoals is derivable from $(\mathcal{R}\setminus\{\psi\}, V(w), \mathbf{a})$ and therefore s is derivable from $(\mathcal{R} \setminus \{\psi\}, V(w), \mathbf{a})$.

Right-to-left: suppose that $dd(s) = n$ and that s is derivable from $(\mathcal{R} \setminus \{\psi\}, V(w), \mathbf{a})$. Again, let ϕ denote any rule that satisfies the conditions of Def. 8. We know that all positive subgoals and none of the negative subgoals of ϕ is derivable from $(\mathcal{R} \setminus \{\psi\}, V(w), \mathbf{a})$, and by the induction hypothesis the same holds for $(\mathcal{R}, V(w), \mathbf{a})$. Again, using the fact that the head of ϕ is unifiable with s, and therefore that ϕ must be relevant, it follows that s must be derivable from $(\mathcal{R}, V(w), \mathbf{a})$.

Definition 12 (Correct Description) We say a set of GDL rules $\mathcal{R}_G$ is a correct description of G if there exists a valuation function V such that for any relevant ground atom s for which the relation symbol is a GDL keyword, we have that

$$\text{s is derivable from } (\mathcal{R}_G, V(w), \mathbf{a}) \quad \Rightarrow \quad G, V \vDash_{(w,\mathbf{a})} s$$

Definition 13 (Complete Description) We say a set of GDL rules $\mathcal{R}_G$ is a complete description of G if there exists a valuation function V such that for any ground atom s for which the relation symbol is a GDL keyword other than 'distinct', we have that

$$G, V \vDash_{(w,\mathbf{a})} s \quad \Rightarrow \quad \text{s is derivable from } (\mathcal{R}_G, V(w), \mathbf{a})$$

Note that even if a description of G is correct and complete, it still may contain irrelevant rules, because the definition of 'correct' is limited to relevant atoms.

Proposition 2 *Given a correct and complete description of a Mediation Game, an agent with unbounded computational resources is able to find all solutions to that Mediation Game.*

Proof The description allows the agent to derive the initial state, for each state the set of legal actions, and for each state and each legal action, the next state. Therefore, the agent can systematically explore the space of plans. Furthermore, given that the set of cycle-free plans is finite, the agent can explore the space of cycle-free plans exhaustively, in a finite amount of time. Again, using the description, the agent can determine for each cycle-free plan its terminal state, and the players' utility values for that state, and therefore it can detect whether the plan is a solution or not.

Proposition 3 *Let $\mathcal{R}_G$ be a complete and correct description of some game G, and let S be a subset of $\mathcal{R}_G$ such that every rule in S is irrelevant in $\mathcal{R}_G$, then the set of rules $\mathcal{R}_G \setminus S$ is also a complete and correct description of G.*

Proof According to Lemma 4 the question whether an atom s is derivable or not does not change if we remove an irrelevant rule from $\mathcal{R}_G$. Furthermore, because of Lemma 2, this continues to hold if we remove all the rules in S, one by one, from $\mathcal{R}_G$. Finally, the entailment of s does not depend on the rules in $\mathcal{R}_G$. Therefore, if we look at Defs. 12 and 13 we see that nothing changes about the correctness or completeness of $\mathcal{R}_G$.

Definition 14 (Minimal Description) Let $\mathcal{R}_G$ be a complete and correct description of a game G. We say it is minimal iff it does not contain any irrelevant rules.

Just like the notion of an 'irrelevant' rule, the notion of a 'minimal' description does not appear in existing literature, because in GGP there is usually no reason to consider any non-minimal descriptions.

6 Mediation Problems

In Section 4 we have defined a Mediation Game as a type of game, and in Section 5 we have explained how such a game can be described as a set of GDL rules. However, in this paper we want to tackle a problem in which the agents do not have complete knowledge of the game they are playing. That is, they are playing some Mediation Game G which is completely described

by some set of GDL rules $\mathcal{R}_G$, but each agent is only aware of a subset of those rules.

Given a mediation game G we can define a *Mediation Problem* as follows:

Definition 15 (Mediation Problem) A Mediation Problem is a tuple $\langle G, B_\alpha, B_\beta, B_\mu \rangle$ where G is a Mediation Game and B_α, B_β, and B_μ are sets of GDL rules such that the union $B_\alpha \cup B_\beta \cup B_\mu$ forms a correct and complete description of G.

The set B_i (with $i \in Ag$) is called the *knowledge base* of agent i. These knowledge bases represent the knowledge the respective agents have about the Mediation Game they are playing. Each agent only has access to its own knowledge base, but this knowledge base does not necessarily form a complete description of the Mediation Game, so the agents have incomplete information about the problem they are aiming to solve. Therefore, individual agents may not be able to find a solution to the problem, unless they exchange knowledge.

For example, if an agent knows the current world state w_t, and it knows which actions are chosen by all the agents, it may still not know which will be the next state w_{t+1} until these actions are executed, because it does not know all the rules that describe the update function u.

Especially, this means that the mediator generally will not be able to propose any plan before it has requested all relevant information from the players to gather a complete description of the Mediation Game.

We do not pose any restrictions on how the rules that describe G are divided among the three initial knowledge bases. Either of the three could be empty, or could be a complete description of G. Furthermore, they do not need to be disjoint (but they could be). Also, it essential to understand that the union $B_\alpha \cup B_\beta \cup B_\mu$ will in general not be minimal. We consider the three knowledge bases as static (i.e. nothing will be added or removed from them during the execution of the mediation algorithm). Instead, as we will explain later, we assume the mediator will collect its information received from the players in a separate *working knowledge* base.

We say a Mediation Problem is *trivial* if both B_α and B_β already contain enough information that α and β both find the solutions to the game individually.

Definition 16 (Trivial Mediation Problem) A Mediation Problem is said to be trivial if both B_α and B_β form a correct and complete description of the mediation game.

If a Mediation Problem is trivial then any proposal made by the mediator could also have been found by the players themselves, and therefore the mediator is essentially useless. In other words, the application of a mediation algorithm only makes sense for non-trivial Mediation Problems.

Clearly, our problem is different from GGP, because we are assuming the players only know part of the rules of the game. Another important difference is that in GGP one is interested in designing an algorithm for one of the players, that aims to find the sequence of actions that maximizes the utility for that player. In our case, however, we are implementing an algorithm for the mediator, which does not have its own utility function, but instead is only interested in finding solutions that are acceptable to both α and β.

Finally, let us remark that we do not expect the agents to reason about the knowledge of other agents. Each agent simply knows exactly those rules that are in its own knowledge base, and has no idea about the contents of the other agents' knowledge bases. Our mediation algorithm does not attempt to reason about the contents of the players' knowledge bases. For this reason we do not need to apply any form of epistemic logic.

7 The Home Improvement Domain in GDL

In this section we give a minimal description of the home improvement domain in GDL, distributed over three knowledge bases B_α, B_β and B_μ.

These three knowledge bases partially overlap, and therefore, in order that we do not have to repeat the same rules more than once, we will divide the rules into four sets B_s, B'_α, B'_β and B'_μ, where B_s is the set of shared knowledge:

$$B_s := B_\alpha \cap B_\beta \cap B_\mu,$$

and B'_i is defined as the knowledge known to agent i apart from the shared knowledge.

$$B'_i := B_i \setminus B_s \quad \text{for all } i \in \{\alpha, \beta, \mu\}$$

7.1 Shared Knowledge

The shared knowledge B_s consists of the following rules, which are explained below:

$$\rightarrow role(\alpha) \tag{2}$$
$$\rightarrow role(\beta) \tag{3}$$
$$\rightarrow role(\mu) \tag{4}$$

$$\rightarrow resource(nail) \tag{5}$$
$$\rightarrow resource(screw) \tag{6}$$
$$\rightarrow resource(hammer) \tag{7}$$
$$\rightarrow resource(screwdriver) \tag{8}$$
$$\rightarrow resource(mirror) \tag{9}$$
$$\rightarrow resource(picture) \tag{10}$$

$$role(?i) \rightarrow input(?i, hang\_mirror) \quad (11)$$
$$role(?i) \rightarrow input(?i, hang\_picture) \quad (12)$$
$$role(?i_1) \wedge role(?i_2) \wedge resource(?r) \rightarrow input(?i_1, give(?r, ?i_2)) \quad (13)$$
$$role(?i) \rightarrow input(?i, noop) \quad (14)$$

$$true(have(?i_1, ?r)) \rightarrow legal(?i_1, give(?r, ?i_2)) \quad (15)$$
$$role(?i) \rightarrow legal(?i, noop) \quad (16)$$

$$does(?i_1, give(?r, ?i_2)) \rightarrow next(have(?i_2, ?r)) \quad (17)$$
$$does(?i_1, give(?r, ?i_2)) \rightarrow lose(?i_1, ?r) \quad (18)$$
$$true(have(?i_1, ?r)) \wedge \neg lose(?i_1, ?r) \rightarrow next(have(?i_1, ?r)) \quad (19)$$
$$does(?i_1, hang\_mirror) \rightarrow next(mirror\_hanging) \quad (20)$$
$$does(?i_1, hang\_picture) \rightarrow next(picture\_hanging) \quad (21)$$

$$\rightarrow init(first\_round) \quad (22)$$
$$true(first\_round) \rightarrow next(second\_round) \quad (23)$$
$$true(second\_round) \rightarrow next(last\_round) \quad (24)$$
$$true(last\_round) \rightarrow terminal \quad (25)$$

Facts 2-4 define the roles in this game. Facts 5-10 define that the constants 'nail', 'screw', 'hammer', 'screwdriver', 'mirror' and 'picture' all represent resources (note that 'resource' is not a keyword of GDL, but a predicate that we have defined ourselves).

Rules 11-14 define the possible actions the agents can take. Specifically, rules 11 and 12 state that any agent can perform the action *hang_mirror* or the action *hang_picture*. rule 13 states that any agent i_1 can perform the action $give(r, i_2)$ if r is a resource and i_2 is an agent. We will later see that this has the interpretation of giving the resource r to agent i_2. Rule 14 states that an agent i can also simply do nothing (the constant *noop* does not have any formal meaning, but is commonly used as a dummy to represent non-action).

Rules 15-16 define when it is legal (or possible) to perform these actions. Rule 15 states that it is possible for an agent i_1 to give a resource r to an agent i_2 if i_1 currently owns that resource. Rule 16 says that it is always possible for any agent i to do nothing. Note that there are no rules here to define when *hang_mirror* and *hang_picture* are possible. That is because the rules here only represent the shared knowledge and in this example the knowledge when you can hang a mirror or a picture on the wall is not known to all agents.

Rules 17-21 define how the game state evolves depending on the actions chosen by the agents. Rule 17 states that if an agent i_1 gives a resource r to an agent i_2 then in the next state agent i_2 will own that resource and rule 18 states that in that case i_1 will lose the resource. Rule 19 states that if an agent i_1 currently owns a resource r and it is not losing it, then in the next state that will still own r. We note here that in GDL every predicate is by default considered false in each round of the game, unless for that round it is explicitly stated that the predicate is true. Therefore, we need rule 19 to define that an agent keeps a resource unless he gives it away.

7.2 Rules Known to α

The private knowledge base B'_α of α consists of the following rules:

$$\rightarrow init(have(\alpha, hammer)) \quad (26)$$
$$\rightarrow init(have(\alpha, picture)) \quad (27)$$
$$\rightarrow init(have(\alpha, screw)) \quad (28)$$
$$\rightarrow init(have(\beta, nail)) \quad (29)$$

$$true(have(?i, hammer)) \wedge true(have(?i, nail)) \wedge true(have(?i, picture)) \rightarrow legal(?i, hang\_picture)) \quad (30)$$

$$true(picture\_hanging) \rightarrow goal(\alpha, 100)) \quad (31)$$

Rules 26-29 state that agent α knows it has a hammer, a picture, and a screw, and that it knows that β has a nail. Furthermore, rule 30 states that if you have a hammer, a nail and a picture then you can hang the picture on the wall. The last rule indicates that α's goal is to have the picture hanging on the wall.

7.3 Rules Known to β

The private knowledge base B'_β of β consists of the following rules:

$$\rightarrow init(have(\beta, nail)) \quad (32)$$
$$\rightarrow init(have(\beta, mirror)) \quad (33)$$

$$true(have(?i, hammer)) \wedge (true(have(?i, nail)) \wedge true(have(?i, mirror)) \rightarrow legal(?i, hang\_mirror)) \quad (34)$$

$$true(mirror\_hanging) \rightarrow goal(beta, 100)) \quad (35)$$

Rules 32 and 33 state that agent β knows he initially has a nail and a mirror. Rule 34 states that he knows that with a hammer a nail and mirror you can hang the mirror on the wall. The last rule indicates that his goal is to have the mirror hanging on the wall.

7.4 Rules (Initially) Known to μ

The initial knowledge base B'_μ of μ consists of the following rules:

$$\rightarrow init(have(\mu, screwdriver)) \tag{36}$$

$$true(have(?i, hammer)) \ \wedge \ true(have(?i, nail)) \wedge true(have(?i, mirror)) \rightarrow legal(?i, hang\_mirror)) \tag{37}$$

$$true(have(?i, screw)) \ \wedge \ true(have(?i, screwdriver)) \wedge true(have(?i, mirror)) \rightarrow legal(?i, hang\_mirror) \tag{38}$$

This means that the mediator knows he owns a screw driver, and that the mediator knows two ways of hanging a mirror: either using a nail and a hammer, or using a screw and a screwdriver.

7.5 Resources

In the home improvement domain, the main question is how the agents can exchange resources, such as a hammer, a nail or a screw, in such a way that both players can be satisfied. GDL does not have any explicit notion of a resource. However, as demonstrated in the example above, we can easily model resources in GDL. For example, we can define a constant 'nail' and define the fact that α initially owns a nail as follows:

$$init(have(\alpha, nail))$$

Furthermore we have defined a nail to be a resource, by stating:

$$\rightarrow resource(nail)$$

Then, in combination with rules 13, 15, 17, 18, and 19, we are sure that it behaves exactly like we expect a resource to behave.

8 Mediation Algorithm

We will now present the algorithm that we have implemented for the mediator. Throughout this section we assume some fixed Mediation Problem $\langle G, B_\alpha, B_\beta, B_\mu \rangle$ with binary utility functions. The algorithm takes the mediator's knowledge base B_μ as input. When the mediator agent is started it proceeds as follows:

1. The mediator initializes a new, empty, database B_w, which we call its *working knowledge*.
2. The mediator runs its Information Gathering algorithm (Sec. 8.1) to collect all relevant rules from $B_\alpha \cup B_\beta \cup B_\mu$ and copies them into its working knowledge B_w.
3. The mediator starts a planning algorithm to find a solution to the problem, defined by the GDL rules in B_w.
4. If the mediator cannot find any (new) solution, the algorithm returns unsuccessfully.
5. If the mediator does find a solution, he proposes it to the players.
6. If both players accept the solution, the algorithm returns successfully.
7. If at least one of the players does not accept the solution, the algorithm goes back to step 3 to find a new solution.

8.1 Information Gathering

In step 2 of the algorithm above, the mediator collects rules from the players' knowledge bases and from its own knowledge base, which it will then use to find solutions.

One naive implementation of this Information Gathering algorithm, would be that the mediator simply requests *all* knowledge from the players' knowledge bases and copies it, together with all the contents of its own knowledge base into its working knowledge. However, in practice this would be very inefficient, because the players and the mediator may have very large knowledge bases containing lots of irrelevant information. In a real-world setting this may occur because the agents are not implemented to solve one specific problem, but instead have information that is useful for a whole range of various problems.

One could try to solve this by simply giving each agent a separate database for each problem it has knowledge about, but this is unrealistic in practice, since many pieces of information may be related to more than one problem. For example, the knowledge that in order to hit a nail you need a hammer is useful for hanging a picture on the wall, but is equally relevant to the problem of building a wooden piece of furniture.

Instead, we here present an algorithm that allows the mediator to request specific information, and that allows the mediator to determine precisely which kind of information to request.

We should note that an agent generally cannot determine by itself whether a rule is relevant or not. Suppose for example that α has the following two rules in its knowledge base:

$$q \wedge s \rightarrow t \qquad p \rightarrow goal(\alpha, 100)$$

while agent β has the rule

$$t \rightarrow p$$

in its knowledge base. From the point of view of α his two rules are completely unrelated. However, in combination with the rule from agent β, we see that α's first rule is relevant to satisfy the subgoal of α's second rule.

We assume a protocol that allows the mediator to request information from each of the three knowledge bases. When the mediator 'requests' a GDL atom s from a knowledge base B_i with $i \in \{\alpha, \beta, \mu\}$, the mediator will receive a response that consists of the list of all rules from B_i for which the head is unifiable with s. Of course, the mediator does not have direct access to the players' knowledge bases, but we can imagine the mediator sending a message containing the request to the player and then the player will execute the request and send a message back to the mediator containing the response.

The mediator's Information Gathering algorithm works as follows:

1. Initialize an empty list called the 'request list'.
2. Request the atoms $goal(?i, ?x)$, and $terminal$ from each of the three knowledge bases B_α, B_β, and B_μ.
3. For each response received, copy its contents into the working knowledge base B_w.
4. For every new rule in B_w extract the atoms that appear in its body. Then for each such atom s:
   - If s is of the form $does(i, a)$, add the atom $legal(i, a)$ to the request list.
   - Else, if s is of the form $true(t)$, add the atoms $init(t)$ and $next(t)$ to the request list.
   - Else, add s itself to the request list.
5. If the request list is empty, then return.
6. Remove the first atom s from the request list.
7. Check if s has been requested before. If yes, then go back to step 5.
8. Request the atom s from each of the three knowledge bases B_α, B_β, and B_μ.
9. Go back to step 3.

For the following lemmas and theorems we always assume that the players respond faithfully to the mediator's queries.

Lemma 5 *If an atom s is relevant in $B_\alpha \cup B_\beta \cup B_\mu$, and does not contain the relation symbols 'does' or 'true' then at some point during the execution of the algorithm the mediator will request it.*

Proof The proof now goes by induction on relevance depth. If $rd(s) = 0$ then indeed s is requested at step 2 of the algorithm. Now suppose that all relevant atoms s' with $rd(s') < n$ have been requested and that we have a relevant atom s with $rd(s) = n$. Let us first prove the lemma for the case that s is not of the form $legal(i, a)$, $init(t)$, or $next(t)$. In this case, by the definition of relevance and relevance depth, there must be a rule ϕ with a subgoal that is unifiable with s or $\neg s$, and such that for the head h of ϕ we have $rd(h) = n - 1$. Since the head of a rule cannot contain the relation symbols 'does' or 'true', it follows from the induction hypothesis that h is requested by the algorithm. This in turn means that the mediator must at some point receive ϕ in some reply message, and (unless s is of the form $true(t)$ or $does(i, a)$) s will be put on the request list. Now suppose that s is of the form $legal(i, a)$, and define s' to be $does(i, a)$. We then know that s' is relevant, and applying the same reasoning as above to s', we conclude that the mediator must at some point receive a message containing a rule ϕ that has a subgoal that is unifiable with s'. Upon receiving this message the algorithm will put $legal(i, a)$ on the request list. In case s is of the form $init(t)$ or $next(t)$ the same reasoning applies, but with s' being $true(t)$.

Lemma 6 *Any atom requested by the mediator is relevant in $B_\alpha \cup B_\beta \cup B_\mu$.*

Proof The first atoms requested contain the relation symbols 'terminal' and 'goal', and are therefore relevant by definition. Now, suppose that at some point during the execution of the algorithm all previously requested atoms were relevant. Let s be any atom on the request list. First consider the case that s is not of the form $legal(i, a)$, $next(t)$ or $init(t)$. Then we know that s was in the body of some rule ϕ which was in the content of a reply message from a player. This means that the head h of ϕ must be unifiable with a previously requested atom, and therefore h is relevant. This in turn means that s itself is also relevant. Now let us look at the case that s is of the form $legal(i, a)$. In that case we know that the mediator must have received a message containing a rule ϕ with $does(i, a)$ in its body. Again, this means that the head of ϕ is relevant and therefore that $does(i, a)$ is relevant. Then, by definition of relevance, this means that $legal(i, a)$ is also relevant. The proof goes similarly when s is of the form $next(t)$ or $init(t)$.

Lemma 7 *Let S denote the set of all irrelevant rules in $B_\alpha \cup B_\beta \cup B_\mu$. Then, when the Information Gathering algorithm has finished we have:*

$$B_w = B_\alpha \cup B_\beta \cup B_\mu \setminus S$$

Proof According to the previous lemmas, an atom s which is not of the form $true(t)$ or $does(t)$ is requested if and only if it is relevant in $B_\alpha \cup B_\beta \cup B_\mu$. Furthermore, since a rule ϕ is contained in a reply message iff its head was requested, and since the head of a rule cannot be of the form $true(t)$ or $does(t)$, we have that ϕ is contained in a reply message iff its head is relevant in $B_\alpha \cup B_\beta \cup B_\mu$,

which means that ϕ itself is relevant in $B_\alpha \cup B_\beta \cup B_\mu$. Finally, since B_w will consist of exactly the rules that were contained in any reply message, the lemma follows.

We are now ready to state the main theorem of this paper.

Theorem 1 *When the Information Gathering algorithm has finished B_w will be a complete, correct, and minimal description of G.*

Proof The set $B_\alpha \cup B_\beta \cup B_\mu$ was assumed to be a complete and correct description of G. Combining Lemma 7 with Proposition 3 we conclude that B_w is complete and correct. Furthermore, from Lemma 7 and Proposition 1 we also conclude that B_w is minimal.

8.2 Finding Solutions

Once the Information Gatheting algorithm has finished the mediator can start searching for solutions. In order to perform a search through the state space of the game defined by a set of GDL rules one needs a so-called *State Machine*. This is a data structure that is initialized with a set of GDL rules $\mathcal{R}$, and then for any input consisting of a world state w, an action profile $\mathbf{a}$ and an atom s it outputs 'true' if s is derivable from $(\mathcal{R}, V(w), \mathbf{a})$, and 'false' otherwise.

The standard GGP code base comes with two different implementations of a State Machine: one based on a first-order backward chaining algorithm, called the AIMA prover, and one based on a forward chaining algorithm for propositional logic, called a PropNet. The PropNet is usually much faster than the AIMA prover, but has the disadvantage that it requires a grounding step that replaces all rules containing variables with grounded rules, and therefore takes more time to initialize.

In order to provide evidence that our mediation algorithm works correctly we have implemented a simple brute-force exhaustive tree search algorithm to find solutions, based on the AIMA prover. We first apply the above described knowledge gathering algorithm to collect the relevant rules in the mediator's working knowledge B_w and then use the rules in B_w to initialize the AIMA prover. We then use this to generate a search tree. Each node ν_w in the tree represents a state w and each arc between a node ν_w and its child node $\nu_{w'}$ represents an action profile $\mathbf{a}$ that is legal in ν' and for which $u(w, \mathbf{a}) = w'$. The tree is generated by a depth-first search, until a node ν^* is generated corresponding to some terminal state w^* that yields a utility value of 100 for both players.

We have tested this algorithm on the Home Improvement domain, and indeed it managed to find the correct solution.

Of course, in domains that are more complicated than the Home Improvement domain this approach may not work because the domain could be too large for exhaustive search. There are many other search techniques that can be used in such cases, such as heuristic search and Monte Carlo Tree Search (MCTS). However, the goal of this paper is to present a smart knowledge gathering algorithm that is able to collect all relevant information, rather than to present the best planning/GGP algorithm to find solutions using that information.

9 Experiments

In order to test our algorithm we have conducted a number of experiments. The idea is that we compare our 'smart' mediator algorithm with a 'naive' mediator. The smart mediator uses the Information Gathering algorithm described in Section 8.1. The naive mediator, on the other hand, simply requests the entire knowledge base of each agent and does not distinguish between relevant and irrelevant rules. In other words, it uses the entire set of rules $B_\alpha \cup B_\beta \cup B_\mu$ to initialize its State Machine.

9.1 Setup

We have used three quantities to measure the efficiency of both implementations. Firstly, we measure how much time it takes them to collect the rules into the working knowledge B_w. Secondly, we have measured how much time it takes to initialize the AIMA Prover and the PropNet. Note that here we are not actually *using* the AIMA Prover or the PropNet. We are only *initializing* them so that we can measure how much time that takes.

The games we have used for our experiments are Tic-Tac-Toe, Breakthrough, Connect-4, Free-For-All, and Qyshinsu. The descriptions of the games were downloaded from the GDL repositories at `http://games.ggp.org/`. Note that these games are ordinary games, rather than Mediation Games. This is not important, however, because we are only testing the mediator's Information Gathering algorithm, and not its ability to find a plan that satisfies the players' goals.

In each experiment we have taken the GDL description of one game and randomly distributed its rules among the three knowledge bases. For the largest of the tested games, Qyshinsu, the description contains 223 relevant rules. We then augmented each of these

knowledge bases with up to 200,000 randomly generated irrelevant rules.

All algorithms were implemented in Java, and all experiments were performed on a HP Z1 G2 workstation with Intel Xeon E3 4x3.3GHz CPU and 8 GB RAM.

9.2 Results

As explained, we have used five games for our experiments. However, since the results for all these games are virtually identical we here only discuss the experiments with Tic-Tac-Toe. The results are displayed in Tables 1, 2, and 3. For the the results obtained with the other games we refer to the Appendix.

In each of the tables the first row displays the number of irrelevant rules added to each knowledge base. The second and third row respectively show the measured times for the smart and naive mediator. The results are averaged over 500 repetitions of the experiments.

From Tables 2 and 3 we see that the time the smart mediator needs to initialize the AIMA Prover and PropNet are independent of the number of irrelevant rules, whereas for the naive algorithm these times clearly increase. This makes sense, because the smart mediator only uses the relevant rules to initialize the State Machines.

From Table 1 we see that the smart mediator is also faster when it comes to gathering the rules, which is perhaps less obvious. After all, the naive mediator only needs to send one request to both players and then immediately receives all rules, whereas the smart mediator needs to go through several iterations to determine the relevant rules, and for each request from the mediator the agents need to search again for the right rules within their respective knowledge bases. However, searching for a rule with a given head can be done in constant time (if the rules are stored in a hash table), so the large size of the databases is not very important. Furthermore, the smart mediator has the advantage that it only needs to copy a small number of rules to its working knowledge, which is independent of the number of irrelevant rules, whereas the naive algorithm must copy the complete knowledge bases to its working knowledge, which is linear in the number of irrelevant rules. Indeed, we see that the results of the smart algorithm remain constant while the results of the naive algorithm are (roughly) linearly increasing.

irr. rules/agent:	0	40,000	80,000	120,000	160,00	200,000
smart:	1	1	2	1	0	1
naive:	0	108	209	310	534	718

Table 1 Time required to collect the rules from the players, in milliseconds.

irr. rules/agent:	0	40,000	80,000	120,000	160,00	200,000
smart:	0	0	0	0	0	0
naive:	0	337	698	943	1,369	2,555

Table 2 Time required to initialize the AIMA Prover, in milliseconds.

irr. rules/agent:	0	40,000	80,000	120,000	160,00	200,000
smart:	4	9	20	11	6	15
naive:	17	5,722	12,123	17,949	25,272	31,720

Table 3 Time required to initialize the PropNet, in milliseconds.

10 Conclusions

In this paper we have presented the notion of a Mediation Problem as a multi-agent problem in which each agent has a large data base containing knowledge, but only a small part of that knowledge is relevant to the problem at hand, and the knowledge of each agent is incomplete. Furthermore, there is a third agent, known as the mediator, which is neutral and helps the players to find a plan that is acceptable to both of them.

The mediator faces two major challenges. The first challenge is to collect all knowledge from the players and from its own database that is relevant to the problem they are aiming to solve. The second challenge is to use that knowledge in order to find a plan that solves the problem. We have focused on the first of these two challenges, as the second challenge can be solved using existing planning algorithms or GGP algorithms.

We have provided a simple example of a Mediation Problem, and we have implemented an information gathering algorithm that allows the mediator to request exactly those rules from the agents' knowledge bases that are relevant to the problem.

We have given a formal proof that our algorithm outputs a complete, correct, and minimal description of the Mediation Game, and we have showed with several experiments that this algorithm is much more efficient than the naive solution in which the mediator simply uses all available knowledge, including irrelevant rules.

Furthermore, we have argued that GDL is a natural choice for the description of Mediation Problems, because it allows us to define games, and allows us to distribute the description of the game among the agents such that each only has partial knowledge.

11 Future Work

Currently, our mediation algorithm assumes that the agents are willing to provide all their relevant information to the mediator. It would be interesting to investigate what happens if players could withhold information for strategic reasons. For this we first need to investigate to what extend it is possible to construct plans if the rules known to the mediator do not form a complete description of the game.

Another interesting way to increase the complexity of the domain would be to allow for conflicting information. The mediator may at some point believe some predicate ϕ to be true, but then may need to revise this belief if one of the agents provides new information that is inconsistent with ϕ. The players may then use arguments to convince the mediator to discard certain information to re-establish consistency of its knowledge. Introducing this possibility would lead us into the realm of Argumentation Based Negotiation (ABN).

We would also like to investigate whether our approach can be applied when mediation involves languages other than GDL, such as CANPlan (Sardina and Padgham, 2011).

Finally, we plan to investigate to what extent we could make use of GDL-II to define Mediation Games in which players have imperfect information about the state of the game and in which actions may have indeterministic outcomes.

Acknowledgments

This work was sponsored by Endeavour Research Fellowship 4577_2015 awarded by the Australian Department of Education.

Appendix A

In this section we present the results obtained with a number of other games. All results in this section were averaged over 100 repetitions.

Experimental Results Connect-4

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	0	1	0	0	1	0
naive:	0	44	99	132	189	262

Table 4 Time required to collect the rules from the players, in milliseconds.

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	0	0	0	0	0	0
naive:	0	143	309	415	573	733

Table 5 Time required to initialize the AIMA Prover, in milliseconds.

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	32	36	27	31	35	34
naive:	31	2,848	5,804	8,685	11,873	14,712

Table 6 Time required to initialize the PropNet, in milliseconds.

Experimental Results Breakthrough

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	0	1	1	1	1	1
naive:	0	48	97	145	188	249

Table 7 Time required to collect the rules from the players, in milliseconds.

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	0	0	0	0	0	0
naive:	0	143	290	415	659	853

Table 8 Time required to initialize the AIMA Prover, in milliseconds.

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	2,252	2,225	2,244	2,134	2,302	2,333
naive:	2,076	4,332	6,954	9,352	12,029	14,393

Table 9 Time required to initialize the PropNet, in milliseconds.

Experimental Results Free-For-All

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	0	1	0	0	2	0
naive:	0	43	92	128	193	250

Table 10 Time required to collect the rules from the players, in milliseconds.

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	0	0	0	0	0	0
naive:	0	140	281	467	589	846

Table 11 Time required to initialize the AIMA Prover, in milliseconds.

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	140	137	143	126	144	159
naive:	142	3,127	6,392	9,367	12,007	14,822

Table 12 Time required to initialize the PropNet, in milliseconds.

Experimental Results Qyshinsu

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	5	1	2	3	2	3
naive:	0	42	107	125	193	221

Table 13 Time required to collect the rules from the players, in milliseconds.

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	0	0	1	0	0	0
naive:	0	142	299	405	545	905

Table 14 Time required to initialize the AIMA Prover, in milliseconds.

irr. rules/agent:	0	20,000	40,000	60,000	80,00	100,000
smart:	168	244	162	204	226	233
naive:	138	3,426	6,981	10,511	14,374	17,876

Table 15 Time required to initialize the PropNet, in milliseconds.

References

Baydin AG, López de Mántaras R, Simoff S, Sierra C (2011) CBR with commonsense reasoning and structure mapping: An application to mediation. In: Ram A, Wiratunga N (eds) Case-Based Reasoning Research and Development: 19th International Conference on Case-Based Reasoning, ICCBR 2011, London, UK, September 12-15, 2011. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 378–392, DOI 10.1007/978-3-642-23291-6_28, URL http://dx.doi.org/10.1007/978-3-642-23291-6_28

Bellucci E, Zeleznikow J (2005) Developing negotiation decision support systems that support mediators: case study of the Family_Winner system. Artificial Intelligence and Law 13(2):233–271

Bench-Capon TJM (2003) Persuasion in practical argument using value-based argumentation frameworks. Journal of Logic and Computation 13(3):429–448

Ceri S, Gottlob G, Tanca L (1989) What you always wanted to know about datalog (and never dared to ask). IEEE Transactions on Knowledge and Data Engineering 1(1):146–166, DOI http://doi.ieeecomputersociety.org/10.1109/69.43410

Chalamish M, Kraus S (2012) AutoMed: an automated mediator for multi-issue bilateral negotiations. Autonomous Agents and Multi-Agent Systems 24:536–564, DOI 10.1007/s10458-010-9165-y

Debenham J (2004) Bargaining with information. In: Jennings NR, Sierra C, Sonenberg L, Tambe M (eds) Proceedings Third International Conference on Autonomous Agents and Multi Agent Systems AAMAS-2004, ACM Press, New York, pp 664–671

Debenham JK, Simoff S (2006) Negotiating intelligently. In: Bramer M, Coenen F, Tuson A (eds) Proceedings 26th International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, UK, pp 159–172

Genesereth M, Love N, Pell B (2005) General game playing: Overview of the aaai competition. AI Magazine 26(2):62–72

Genesereth MR, Thielscher M (2014) General Game Playing. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, DOI 10.2200/S00564ED1V01Y201311AIM024

Jonge Dd, Zhang D (2016) Using gdl to represent domain knowledge for automated negotiations. In: Osman N, Sierra C (eds) Autonomous Agents and Multiagent Systems: AAMAS 2016 Workshops, Visionary Papers, Singapore, Singapore, May 9-10, 2016, Revised Selected Papers, Springer International Publishing, Cham, pp 134–153

Knuth DE, Moore RW (1975) An analysis of alpha-beta pruning. Artificial Intelligence 6(4):293 – 326, DOI http://dx.doi.org/10.1016/0004-3702(75)90019-3, URL http://www.sciencedirect.com/science/article/pii/0004370275900193

Kocsis L, Szepesvári C (2006) Bandit based monte-carlo planning. In: Proceedings of the 17th European Conference on Machine Learning, Springer-Verlag, Berlin, Heidelberg, ECML'06, pp 282–293, DOI 10.1007/11871842_29

Kolodner JL, Simpson RL (1989) The mediator: Analysis of an early case-based problem solver. Cognitive Science 13(4):507–549

Lin R, Gev Y, Kraus S (2011) Bridging the gap: Face-to-face negotiations with automated mediator. IEEE Intelligent Systems 26(6):40–47

Love N, Genesereth M, Hinrichs T (2006) General game playing: Game description language specification. Tech. Rep. LG-2006-01, Stanford University, Stanford, CA, http://logic.stanford.edu/reports/LG-2006-01.pdf

von Neumann J (1959) On the theory of games of strategy. In: Tucker A, Luce R (eds) Contributions to the Theory of Games, Princeton University Press, pp 13–42

Parsons S, Sierra C, Jennings N (1998) Agents that reason and negotiate by arguing. Journal of Logic and computation 8(3):261–292

Rahwan I, Pasquier P, Sonenberg L, Dignum F (2009) A formal analysis of interest-based negotiation. Annals of Mathematics and Artificial Intelligence 55(3-4):253–276

Sardina S, Padgham L (2011) A bdi agent programming language with failure handling, declarative goals, and planning. Autonomous Agents and Multi-Agent Systems 23(1):18–70, DOI 10.1007/s10458-010-9130-9, URL http://dx.doi.org/10.1007/s10458-010-9130-9

Schei V, Rognes JK (2003) Knowing me, knowing you: Own orientation and information about the opponent's orientation in negotiation. International Journal of Conflict Management 14(1):43–60

Sierra C, Debenham J (2007) Information-based agency. In: Proceedings of Twentieth International Joint Conference on Artificial Intelligence IJCAI-07, Hyderabad, India, pp 1513—1518

Simoff SJ, Debenham J (2002) Curious negotiator. In: Klusch M, Ossowski S, Shehory O (eds) Proceedings of the Int. Conference on Cooperative Information Agents, CIA-2002, Springer, Heidelberg

Sycara KP (1991) Problem restructuring in negotiation. Management Science 37(10):1248–1268

Thielscher M (2010) A general game description language for incomplete information games. In: Fox M, Poole D (eds) Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010, AAAI Press, URL http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1727

Visser W, Hindriks KV, Jonker CM (2011) Interest-based preference reasoning. In: Proceedings of International Conference on Agents and Artificial Intelligence ICAART2011, pp 79–88

Wilkenfeld J, Kraus S, Santmire TE, Frain CK (2004) The role of mediation in conflict management: Conditions for successful resolution. In: et al ZM (ed) Multiple Paths to Knowledge in International Relations, Lexington Books