Rules of Engagement for Automated Negotiation

Alan H. Karp
Intelligent Enterprise Technologies Laboratory
HP Laboratories Palo Alto
HPL-2003-152
July 28th , 2003*

E-mail: alan.karp@hp.com

web services, automated negotiation, ontologies

An important motivation for moving business processes to the web is to reduce the cost associated with having them done by people. One such process is negotiation. Automating negotiations requires that we be able to agree on what we're negotiating for, quantify the value of different possible outcomes, and have a set of rules that specify how the negotiation proceeds. This paper presents a unified approach to address these issues. Contributions include a form of presenting offers that makes it easier to search a large space of potential deals and a negotiation protocol that is guaranteed to terminate in a finite number of steps.

* Internal Accession Date Only
Approved for External Publication

Rules of Engagement for Automated Negotiation

Alan H. Karp
Hewlett-Packard Laboratories
alan.karp@hp.com

August 4, 2003

Abstract

An important motivation for moving business processes to the web is to reduce the cost associated with having them done by people. One such process is negotiation. Automating negotiations requires that we be able to agree on what we're negotiating for, quantify the value of different possible outcomes, and have a set of rules that specify how the negotiation proceeds. This paper presents a unified approach to address these issues. Contributions include a form of presenting offers that makes it easier to search a large space of potential deals and a negotiation protocol that is guaranteed to terminate in a finite number of steps.

1 Introduction

Negotiating contracts is a key step in forming business relationships. Today, people negotiate by mail, telephone, and fax. The unfortunate effect is that the time to reach an agreement can stretch out over weeks or months, costing companies both time and money. As we move business processes to the web, such delays become a limiting factor in the services that can be offered. Automating all, or even some, of the negotiation process promises large efficiency gains.

Software for automated negotiation must be able to locate potential business partners, those with offerings that might satisfy the need, negotiate terms and conditions, and form contracts. Negotiation and contract formation are actually a single activity, because negotiation without contract formation is just playing games. Combining negotiation and contract formation may also make it easier to extend a one-to-one protocol to many parties.

The word negotiation often causes confusion because it is used to denote very different activities, auctioning and bargaining. Auctioning, either direct (buyers bid) or reverse (sellers bid), is the form of negotiation with simpler rules, but it naturally includes multiple parties. Bargaining, involving multiple rounds of offers and counteroffers, has more complex rules, and the extension to multiple parties is less obvious. This work uses the word negotiation in the sense of bargaining only. Further, we assume that each of the parties to the negotiation prefers deals with

larger payoff without concern for the other party's. There is no concept of global efficiency; the parties are purely selfish.

Before starting the discussion it's important to know how we say what we're negotiating for. Section 2 describes the method used in this work, and Appendix A discusses one controversial feature of this approach. As used in e-commerce, the term *contract* has a special meaning. One commonly agreed to formulation is described in Section 3. The negotiation requires the specification of a protocol so that the parties know when to make an offer and what constitutes a legal counteroffer. One negotiation protocol is described in Section 4. Section 5 shows some examples of negotiations run with this set of rules.

2 Ontology

Key to the negotiation mechanisms described in this report is the concept of an ontology that provides the semantic content of the terms in the negotiation. Basically, an ontology provides a context for describing something. For example, the keyword "size" may be used to advertise an item, but is it a shirt size or a shoe size? If the term appears in an ontology related to shoes, we know. Of course, true semantic content depends on human understanding. After all, the word "size" has meaning only to someone who speaks English.

There are many ways to represent ontologies; here we use vocabularies. A vocabulary provides the semantic information in the form of attribute-value pairs. The values may be single values, numeric ranges, or lists of acceptable values. A vocabulary doesn't describe anything; it is just a particular way of describing things. A description assigns values to the attributes of the vocabulary. The analogy is to objects in programming languages. The class corresponds to the vocabulary; the instance, to the description. For example, a **Shoe** vocabulary might have the attributes *Style* and *Color*, while a particular shoe is described in that vocabulary as *Style=loafer* and *Color=brown*.

A vocabulary consists of a set of keywords called *attributes*. Each attribute has a number of properties. More detail on each is given in the sub-sections that follow. Some of these properties are included because of the protocol described in Section 4.

Name: A string different from that of any other attribute in this vocabulary.
Type: The value type of any description using this attribute, *e.g.* integer, string.
Multiplicity: One of *single, multiple, range*.
Value: Zero or more values of the specified type.
Matching rule: A function defining what constitutes a match.

Three other properties are not needed by the protocol, but including them has some benefits in the form of faster or fewer futile negotiations.

Designation: One of *ordered* or *unordered*.
Required: True if at least one value must be supplied for this attribute.
Must-match: True if a look-up will succeed only if this attribute matches.

The vocabulary developer decides on these properties. Anyone using the vocabulary, either to advertise a service or to discover one, must follow the rules of the vocabulary being used. Alternatively, these properties could be determined by a meta-negotiation. However, allowing these properties to be defined by the vocabulary user may lead to unnecessary confusion and should be avoided.

2.1 Name

Attributes in a vocabulary are identified by their names and must be unique within a vocabulary. Hence, if I wish to represent my shoe size and my shirt size, I can either use two different vocabularies, each with an attribute *size*, or I can use one vocabulary with attributes having different names, say *shirtsize* and *shoesize*.

2.2 Value Type

All values specified for a particular attribute must be of the same type.[1] All the basic types - `string`, `int`, `float`, *etc.*, should be supported. In addition, new value types can be introduced as long as their matching rules are expressed in terms of the base types. For example, a value type of `Bigint` can be defined as consisting of two `ints`. A matching rule could then specify equality when the corresponding `ints` of two `Bigints` are equal.

2.3 Value

The value associated with an attribute is a list of zero or more values, except that a *range* attribute must specify exactly two values and an attribute with multiplicity *single* may have no more than one. These values must be of the specified value type.

2.4 Multiplicity

Some attributes may only take on a single value. For example, when listing availability, it makes sense to advertise an item as in stock or not, but not both. Other times, it is reasonable to specify a set of values. A shoe seller may want to list all colors available for a particular style. The protocol described in Section 4 treats range-valued attributes differently than it does multi-valued attributes.

2.5 Matching Rule

Many systems provide flexibility in the matching rules, but these rules are invariably based on the type of the attribute. Often this approach is too limiting. For example, the waist and inseam of a pair of pants may both be represented by integers, but the vocabulary developer may know that the waist attribute requires an exact match while inseams that differ by one unit may still fit. Hence, attribute-based matching

[1] Relaxing this condition would not have a dramatic effect on the negotiation process, but it would make the semantics of the attribute harder to understand.

rules allow more flexibility. Certain denial of service attacks against the party doing the matching can be avoided by having no loops in the language used to specify the matching rules.[2]

2.6 Designation

The protocol in Section 4 defines a valid offer for a multivalued attribute as one having one or more values for that attribute. If the attribute is defined to be *ordered*, we assume that the attribute values are specified in preference order, from most preferred to least preferred. There is no indication of the strength of the preference. The order in which unordered attribute values appear does not indicate a preference. See Appendix A for more discussion.

2.7 Required

Some attributes must be specified or there's no point in advertising or looking up the service. For example, the vocabulary developer may decide that there is no point in negotiating unless the acceptable payment methods are included in the negotiation. This property is not necessary for the protocol described in Section 4. However, defining this property constrains the initial offer in a way that may shorten the negotiation by making the intent of the offerer clearer.

2.8 Must-match

It is often the case that there is no point in reporting a match if a particular attribute value was not included in the constraint expression, *e.g.*, payment method. After all, if we can't agree on means of payment, there's no point in trying to negotiate. Such an attribute can also be used as a simple form of password protection. If you don't know the password, you can't find the service.

3 Contract

Reaching a deal is the point of any negotiation; a contract is the specification of that deal. This work assumes that the parties are negotiating in some marketplace. A marketplace is defined by a set of rules, including such things as who can participate. One of the things a marketplace provides is a set of contract templates. Each contract template is made up of one or more sections. Each section is expressed in terms of a specific vocabulary.[3] The contract is a template because the values of the attributes in the vocabulary of each of the sections are not determined; that is the goal of the negotiation. This framework is not unique to this work[1].

A typical marketplace will provide templates made up of different sections, each section representing an aspect of a deal. For example, there will almost certainly be

[2] If your program has no loops, then it's done. – D. Knuth

[3] We may want to allow a section to contain multiple vocabularies. That way we could express such things as different payment methods for the product and the shipping costs with a single payment vocabulary.

a section on payment method and another on delivery terms, as well as one or more specific to the product. Thus, a shoe-buying marketplace would specify a section with a vocabulary including attributes for size, width, style, material, *etc.* There may be other sections, as well, including, perhaps, sections for return policy and warranty terms.

Clearly, in the most general case, we need a negotiation to determine which sections are included in the contract template. Since this negotiation is most likely identical to the negotiation for the product itself, we'll assume the marketplace specifies the sections. This assumption may not be warranted, making this a topic for further work.

The terms specified by a section are represented by a vocabulary. Any of the attributes in the vocabulary may be included in the negotiation. Terms that are not included in the negotiation are considered irrelevant. For example, shipping cost need not be included in the negotiation if I'm taking the shoes home with me. Also, delivery time may be set to a default of zero, indicating that the customer will take the merchandise at the time of purchase. Negotiation on this term is only needed if its value is to be changed.

4 Protocol

The negotiation protocol used here consists of a number of steps. First, the product is advertised by specifying attributes in a vocabulary or set of vocabularies appearing in a particular contract template. Next, a potential customer finds the advertisement by performing a look-up based on a constraint expression built up using attributes from the vocabulary or vocabularies. Finally, the parties negotiate the terms of a contract by agreeing to specific values of attributes in the contract template. The last step involves committing to these terms in a contract.

The protocol has a number of goals. Some steps are designed to reduce the likelihood of a negotiation failing. Secondly, a negotiation is guaranteed to terminate in a finite number of steps with each party having to store only a modest amount of information. There is no need to use heuristics, such as the number of rounds or amount of time, and no need to worry about cycles, returning to an offer that was made before.

4.1 Finding Possible Deals

We assume that an item is advertised in a particular set of vocabularies, and that the item is found by specifying a constraint expression in those vocabularies. For example, a contract template for buying shoes may have a shoe vocabulary with attributes for style, color, and manufacturer; another vocabulary specifying payment method and price; and a third for delivery options. A search may ask for black or brown wingtips in size 10 to be paid for by credit card from a merchant who has the shoes in stock.

This simple example shows some of the key elements of the negotiation. First, there are a finite number of attributes in the contract template, only some of which

are used for the advertisement or the look-up. Those terms not included in either may not be added to the negotiation, and the corresponding attributes are effectively removed from the contract template.[4] If an attribute is not in either the advertisement or the lookup, we assume neither party cares about it. For example, a shoe store in a shopping mall expects customers to walk out with what they buy; customers of the store expect the same. Thus, even if the contract template specifies clauses for shipping, neither party cares to negotiate their values. The marketplace determines suitable default values.

4.2 Two-party Protocol

The protocol deals with two parties that we'll call the *listener* and the *initiator*. Normally, the seller advertises wares and waits to be approached by a potential buyer. In this case, the seller is the listener and the buyer is the initiator. However, it may be that the buyer has advertised a need, say in a Request for Proposal. In this case, the roles are reversed.

The protocol involves the following steps.

1. The listener advertises in the vocabularies of the contract template.
2. The initiator does a lookup by specifying a constraint expression in those vocabularies.
3. The initiator sends to the listener an offer consisting of values, often numeric ranges or sets of values, for a (proper) subset of the attributes used in the lookup.
4. The listener and initiator take turns sending counteroffers consisting of values, often numeric ranges or sets of values, for a (proper) subset of the attributes used in the advertisement, lookup, and the offer.
5. Either party may send a *negotiation failed* message at any time.

An attribute is *settled* if it appears with a single value in an offer and the same value in the counter to that offer. A settled attribute becomes a clause in the contract and is binding should all the terms in the contract template be settled. A section is said to be *closed* once all its attributes that have been included in the negotiation are settled. The contract is *closed* when all of its sections are closed. At this point the parties have reached a binding contract.

A counteroffer is valid if it satisfies a few conditions.

1. Any attribute that was included in the advertisement (lookup) may be introduced by the listener (initiator) at any time subject to rule 4.
2. The numeric range or set of values of at least one attribute must be narrowed, or a new attribute introduced.
3. No attribute that has been settled may be reintroduced into the negotiation.

[4] Relaxing this restriction would not have a significant effect on the negotiation, but it would complicate some strategies.

4. If all the attributes from a section that were introduced into the negotiation are settled, no more attributes from that section may be introduced.[5]

An invalid offer results in a failed negotiation.

4.3 Dealing with Attribute Designations

Ordered and unordered attributes are handled somewhat differently. Revealing preferences of competitive attributes can put you in a bad bargaining position, but that is less of an issue for cooperative attributes. (See Appendix A.) We use this fact to accelerate the negotiation by assuming that cooperative attributes are rank ordered and competitive ones are not. For example, a specification "style = wingtips, loafers, sandals" might be ordered while "quality = low, medium, high" is not. The strength of the preference is not indicated, and misrepresentations are difficult, but not impossible, to detect. It is possible to extract the values corresponding to the Pareto optimal set if both parties provide their rank orderings [7].

There are special rules for attributes with numeric types that depend on whether the attributes are ordered or unordered. Ordered numeric attributes are represented by a numeric range. For example, if I want to buy in bulk, I might want no more than 500 pairs of shoes, and I don't want to bother with buying unless I can get at least 100 pairs. In addition, I can list my preference since there's no reason the smaller value must come first. Similarly, the seller might have different terms and conditions for different size orders. For example, the minimum order size might be 250 pairs and a maximum size 1,000 pairs. On each round, either party may choose to narrow the range until agreement is reached on a specific value.

Unordered attributes of numeric types are handled differently. For example, we know the buyer wants the lowest possible price, and the seller, the highest. There is no point is specifying a range on such an attribute.[6] In fact, giving a range would be revealing the least acceptable value, giving too much information to the other party. Hence, for unordered range attributes, one party introduces a single value, and the other does the same, either higher or lower. These first two values specify a range that must be narrowed on subsequent rounds.

We can see how this works with an example. The buyer might offer $50 per pair, and the seller could counter with $300. On subsequent rounds, either party can specify a new value, as long as it narrows the existing range. For example, the initial offer might be

Style=wingtip,loafer; Color=black;brown; Quantity=100:500; Price=50,

and the counter offer might be

Style=wingtip; Color=black; Quantity=500:250; Price=50:300.

[5]This simplification is not strictly needed to meet the goals of the protocol. However, closing sections reduces the number of attributes that may be introduced into the negotiation, thereby simplifying the analysis needed by the strategy.

[6]There is an implicit assumption here that the dependence is monotonic. However, subsequent offers can restrict the values to a range in which the utility is monotonic.

Note that the seller's range for the quantity is 250-1,000, but the counter offer can't widen the specified range. Subsequent rounds might be

Style=wingtip; Color=black; Quantity=500:250; Price=50:100
Style=wingtip; Color=black; Quantity=400:400; Price=70:100
Style=wingtip; Color=black; Quantity=400:400; Price=70:70

There is nothing in the rules that prevents one party from changing the other's end of the range. For example, if the price range is 50:100, the seller could offer 60:100. In effect, the seller is asking if the buyer will raise the bid. This strategy may not be a good one, since the buyer may declare a failed negotiation rather than let the seller control the offer in this way.

4.4 Disjunctions

While what has been described can be used, it is not expressive enough. Look at the style and color from the example. The seller might have only black wingtips and brown loafers, but the protocol does not provide a way to say that. We might go quite far in a negotiation before finding out that the buyer wants black loafers.

The solution is to allow disjunctions. In the above example, the first counteroffer could be

Style=wingtip; Color=black; Quantity=250:300; Price=50:90
Style=loafer; Color=brown; Quantity=400:500; Price=50:70

Each disjunction must separately follow the rules spelled out in Section 4.2, including being declared failed. A deal is reached when all the terms in one disjunction have been agreed upon, and only that one disjunction is left.

Why not just try black wingtips first and then brown loafers if the first negotiation fails? That would be inefficient, particularly if there's a substantial cost for a negotiation. Disjunctions allow multiple negotiations to be carried out at the same time, amortizing the costs.

4.5 Summary of Protocol

We can now see why the negotiation terminates in a finite number of steps. There are a finite number of sections in a contract template, and a finite number of attributes in the vocabulary for each section. Each step removes at least one attribute value or narrows a numeric range. Each party need only remember the previous offer to compare with the current counteroffer to determine if the rules are being followed.

Numeric ranges appear to be a problem. After all, I can increase the price I'm willing to pay by $0.01 on each round. While the negotiation will end in a finite number of steps, that number may be so large as to be effectively infinite as far as the parties are concerned. While we could impose heuristics on the amount by which numeric ranges must be narrowed, we rely instead on the behavior of the parties. We assume negotiations carry a cost that depends on their duration. Should either party not negotiate in good faith, the other can always declare a failed negotiation.

This protocol has some shortcomings. For example, you can't change your mind. You may have agreed to buy a pair of shoes from a top of the line manufacturer, but the seller wants more money than you're willing to pay. With a more general protocol, you could change the manufacturer to a less expensive brand. This protocol doesn't give you that option. You must declare a failed negotiation and start over.

This rule means that you must save more than just the previous offer and current counteroffer to avoid cycles; you must also remember what negotiations failed. The hypothesis is that all you need remember is the list of offers that were declared *failed*. If you must remember the entire history of failed negotiations, the arguments about the amount of state needed are wrong.

One unfortunate effect of the strict rules of this protocol is that there may be deals that are possible that you don't find. We know that there are strategies almost certain to miss deals. For example, take it or leave it, in which the offer consists of a single value for each attribute, is likely to miss many possible deals. Therefore, the strategy plays an even more important role than it does in other protocols.

5 Examples

This section contains some examples illustrating the various parts of the rules of engagement presented in the preceding sections. It also contains the results of some negotiations run with simple strategies.

5.1 Contract Template

We will assume a very simple contract template having only three sections. The first section will describe the product; the second, the payment method; and the third, delivery options. The goal is to show the linkage between attributes in different sections of the contract. Each section of the contract template requires a specific vocabulary. For this example, we'll use very simple vocabularies.

Table 1: Shoe Vocabulary

Attribute	Value Type	Multiplicity	Matching Rule	Designation
Style	String	M	==	Ordered
Color	String	M	==	Ordered
Maker	String	M	Soundex[10]	Ordered

The Shoe vocabulary is shown in Table 1. The *Maker* attribute doesn't appear in the negotiations shown below.

In Table 2 a simple payment vocabulary shows some features that must be supported for attribute properties. We see in this example how the vocabulary designer can specify legal values for attributes. The only possible payment methods

are cash, check, or credit; no purchase orders. We also see that prices must be non-negative. Matching rules are specified as functions to be evaluated at run time.[7] In this example, we have a match for payment method if any of the strings in the lookup match any of the strings in the advertisement. Prices match if their numeric ranges overlap.

The type of the payment method is ordered because the seller has costs for each type of payment and would like to provide preferences. We're familiar with the fees charged by credit card companies, but check validation services also charge the merchant. What's less well known is that taking cash also has a cost. The obvious problem is theft, but another significant cost is the additional insurance premium charged for businesses that deal with large amounts of cash. Similarly, we don't know the sign of the payment method term in the buyer's utility function. Some buyers prefer to use a credit card because of the float; others want to avoid running up a large bill. Again, the buyer would like to express preferences.

Table 3 is another very simple vocabulary. In declaring its one attribute ordered, we are assuming that both parties benefit with a shorter delivery time. Perhaps the buyer wants the merchandise as soon as possible, and the seller only gets paid once the goods are delivered. This attribute could also be unordered if, for example, the delivery charge is included in the price, and faster delivery costs more.

5.2 Initial Match

In Section 4.1 we saw that the negotiation starts with an advertisement and a lookup request. In our case, we'll assume that the seller advertises shoes and the buyer searches for them. Thus, the seller has the role of listener; the buyer, that of initiator.

Let's assume that the seller posts the following advertisement.

Shoe Vocabulary
Style = sandal, loafer, wingtip
Color = black, brown, tan
Manufacturer = Florscheim, Dexter

Payment Vocabulary
Payment = cash, credit, check

[7] In the prototype system, these rules are strings evaluated in Python.

Table 2: Payment Vocabulary

Attribute	Value Type	Multiplicity	Matching Rule	Designation
Method	String	M:[Cash, Check, Credit]	F1(value)	Ordered
Price	Float	R:[0,]	F2(value)	Unordered

Price = 50

Delivery Vocabulary
Delay = 0:3

The buyer might specify a constraint expression

Shoe Vocabulary
Style = wingtip OR loafer
Color = black OR brown

Payment Vocabulary
Payment = cash OR credit
Price < 300

Delivery Vocabulary
Delay < 5

These two are compatible, so the buyer will find this seller and can enter into negotiation.

Haven't the buyer and seller revealed too much information? After all, the seller has specified a minimum acceptable price, and the buyer has given a maximum. This information doesn't present a problem for two reasons. Firstly, in many middleware systems, the matching is done by a disinterested third party, such as the marketplace. In such an environment, the initiator doesn't see the advertisement, and the listener doesn't see the search expression. Secondly, these numbers need not be the actual bounds. By expanding the available range, both buyer and seller conceal their actual limits but risk more failed negotiations. Mutually anonymous lookups based on zero knowledge proofs are also possible[9].

5.3 Sample Utility Functions

We need to specify the buyer's and seller's utility functions in order to understand the outcome of the examples shown in Section 5.5. If we assume the seller doesn't care what shoes are sold, prefers cash to credit regardless of delivery method, and wants the highest price, and won't take less than $100, then the utility [5] is

Table 3: Shipping Vocabulary

Attribute	Value Type	Multiplicity	Matching Rule	Designation
Delay	Integer	R:[0,7]	F[value]	Ordered

$$\vec{V}_{seller} = \max\left\{\begin{bmatrix} b_{wingtips} \\ b_{loafers} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\} + \max\left\{\begin{bmatrix} b_{black} \\ b_{brown} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\} + 3\max\left\{\begin{bmatrix} b_{cash} \\ b_{credit} \\ b_{check} \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}\right\} + 2\max\left\{\begin{bmatrix} b_{carry} \\ b_{delivery} \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix}\right\} + H(100, P, \infty)/100 - t/5, \tag{1}$$

where $b_x = 0$ if the attribute value x appears in the offer and $-\infty$ otherwise. Here, $H(a, x, b) = x$ if $a \leq x \leq b$ and $-\infty$ otherwise. The buyer's utility function is

$$\vec{V}_{buyer} = \max\left\{\begin{bmatrix} b_{black,wingtips} \\ b_{brown,loafers} \\ b_{absent} \end{bmatrix} + \begin{bmatrix} 3 - t/25 \\ 2 - t/10 \\ 0 \end{bmatrix}\right\} + \max\left\{\begin{bmatrix} b_{wingtips} \\ b_{loafers} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\} + \max\left\{\begin{bmatrix} b_{black} \\ b_{brown} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\} + \max\left\{\begin{bmatrix} b_{credit,delivery} \\ b_{cash,carry} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\} + H(0, 200 - P, \infty)/25 - t/10 \tag{2}$$

where $b_{absent} = 0$ if no values for the attribute are in the offer and $-\infty$ otherwise. At time $t = 0$ the buyer's payoff for black wingtips is 5 (3 from the combination of black with wingtips and 2 from the terms for color and style), and 4 for brown loafers. These terms are time dependent. In the example, the buyer's preference changes to brown loafers at $t = 4$.[8]

The buyer doesn't have a preference between the other two combinations, so we can assign each one a value of 1. The buyer wants the lowest price, but won't pay more than $200; a higher price violates a constraint. The buyer also insists on paying cash if carrying the shoes out of the store and by credit card if they will be delivered. No other combinations will be accepatble to the buyer.

Time has been included as a linear penalty term having a value equivalent to $2.50 per round for the buyer and $20 per round for the seller. If a deal isn't reached fast enough, the time penalty will make its value negative for either or both parties, resulting in no deal being reached.

5.4 Negotiator's Parameters

There are still a few more things we need to know about each negotiator that have to do with the handling of range-valued attributes, namely price and delivery delay [8]. Each negotiator must specify an initial offer for each such attribute and a final value that won't be passed. In addition, each negotiator specifies the minimum

[8] My brown suit was at the cleaners.

Table 4: Range parameters for sample negotiation

	Price		Delay	
Negotiator	Buyer	Seller	Buyer	Seller
First	50	300	3	2
Final	250	100	0	0
MinInc	53	-57	-2	-1
MaxInc	151	-153	-3	-2

and maximum concession allowed, which may be time dependent or may depend on the size of the interval. Not every strategy uses all these parameters.

Table 4 shows the parameters used for the examples that follow. The buyer's first offer will be $50, and the buyer won't offer more than $250. The buyer's minimum price concession will be $53, and the maximum price concession is $151.[9] We see that both parties want a shorter delivery time.

5.5 Sample Negotiations

A few test cases were run to illustrate some features of the process. All negotiations start from the same advertisement of the seller.

delivery delay [0, 3]
shoe color [black, brown]
shoe style [wingtip, loafer]
payment price [150.0, 300.0]
payment method [cash, credit, check]

In the first experiment, both parties run a random strategy, which involves changing an attribute value at random. No attempt is made to avoid offers that violate the parties' constraints. By trying a several random number seeds, we got the following negotiation:

Time	Negotiator	Vocabulary	Attribute	New Value
0	B	delivery	delay	[0, 0]
1	S	shoe	color	[brown]
2	B	shoe	style	[wingtip]
3	S	payment	price	[150.00, 192.16]
4	B	payment	price	[192.16, 192.16]
5	S	payment	method	[credit, cash]
6	B	payment	method	[cash]

with a final deal of

[9] These values were chosen to make it easy to see how a given offer was reached. Using commensurate values, *e.g.*, $50 for each, might obscure who conceded how much.

delivery delay [0, 0])
shoe color [brown])
shoe style [wingtip])
payment price [192.16, 192.16])
payment method [cash])

which has a value of 7.2 to the buyer and 11.5 to the seller. We can't conclude from these payoffs that the seller got a better deal. These numbers are not directly comparable since the two utility functions can be in entirely different units.

Here we see the value of declaring attributes to be *ordered*. Since the strategy doesn't take into account the payoff corresponding to a counteroffer, it is as likely to pick a bad value as a good one. However, since the values appear in preference order, we can avoid some bad cases by eliminating the least preferred value. In this example, the seller would have picked black at time 1 if the strategy considered the preference ordering. As we will see below, the result would have been worse for the buyer, but there would be no way to know that without looking ahead in the negotiation.

We next ran a case in which both participants examine the attribute values and pick the change that results in the highest payoff. This strategy is equivalent to a hill climbing algorithm with no lookahead. Naturally, both parties make the minimum concession on range-valued attributes. The negotiation was

Time	Negotiator	Vocabulary	Attribute	New Value
0	B	delivery	delay	[0, 1]
1	S	delivery	delay	[0, 0]
2	B	shoe	color	[black]
3	S	shoe	style	[wingtip]
4	B	payment	method	[check, cash]
5	S	payment	method	[cash]
6	B	payment	price	[203, 300]
7	S	payment	price	[203, 243]
8	B	payment	price	[243, 243]

This example reached the deal

delivery delay [0, 0]
shoe color [black]
shoe style [wingtip]
payment price [243.0, 243.0]
payment method [cash]

with a value of 3.6 to the buyer and 11.6 to the seller. Here the seller did marginally better than with a random strategy, and the buyer did considerably worse, paying a higher price and not getting the preferred shoes. That's because a local strategy can't see the consequences of the length of the negotiation. At the time the buyer picks black, there's a higher payoff for black wingtips than brown loafers. On the buyer's next turn, the situation is reversed, but it's too late to change the color.

The next case uses the random strategy for the buyer and the local strategy for the seller. In addition, both parties make the minimum concession on range-valued attributes.

Time	Negotiator	Vocabulary	Attribute	New Value
0	B	delivery	delay	[0, 1]
1	S	delivery	delay	[0, 0]
2	B	shoe	color	[black]
3	S	shoe	style	[loafer]
4	B	payment	price	[203, 300]
5	S	payment	method	[cash, credit]
6	B	payment	price	[250, 300]
7	S	payment	method	[cash]
8	B	payment	price	[250,250]

The resulting deal is

delivery delay [0, 0]
shoe color [black]
shoe style [loafer]
payment price [250,250]
payment method [cash]

which has a value of 5.1 to the buyer and 11.7 to the seller.

Note the last offer by the buyer. Although we normally expect the buyer to raise the low end of the price range, here the buyer lowers the high end. That's because the buyer's upper limit is $250, so the buyer makes the only legal offer that will result in a deal. This example illustrates why the protocol allows this behavior; it enables us to reach deals that would otherwise be missed.

6 Summary

There is a vast literature on automated negotiation[4], far more than can be reviewed here, even limiting the discussion to one-to-one bargaining by selfish parties.

The primary difference between this work and that reported elsewhere is in the form of the offers. In a recent example[2], a contract is represented as a vector of 0s and 1s, representing the absence or presence, respectively, of each term in a contract. The problem with this approach is the difficulty of searching the deal space, which affects the strategies available to negotiators. A similar approach is taken by FIPA[3].

The search problem is greatly simplified using the form for counteroffers presented in Section 4.2. The analogy is to convex hulls. Instead of exploring a distinct set of points in a high-dimensional space, we shrink the size of a convex hull. Hence, instead of examining an offer for black wingtips, then one for brown wingtips, then one for black loafers, and one for brown loafers, we look at an offer for wingtips or loafers in black or brown.

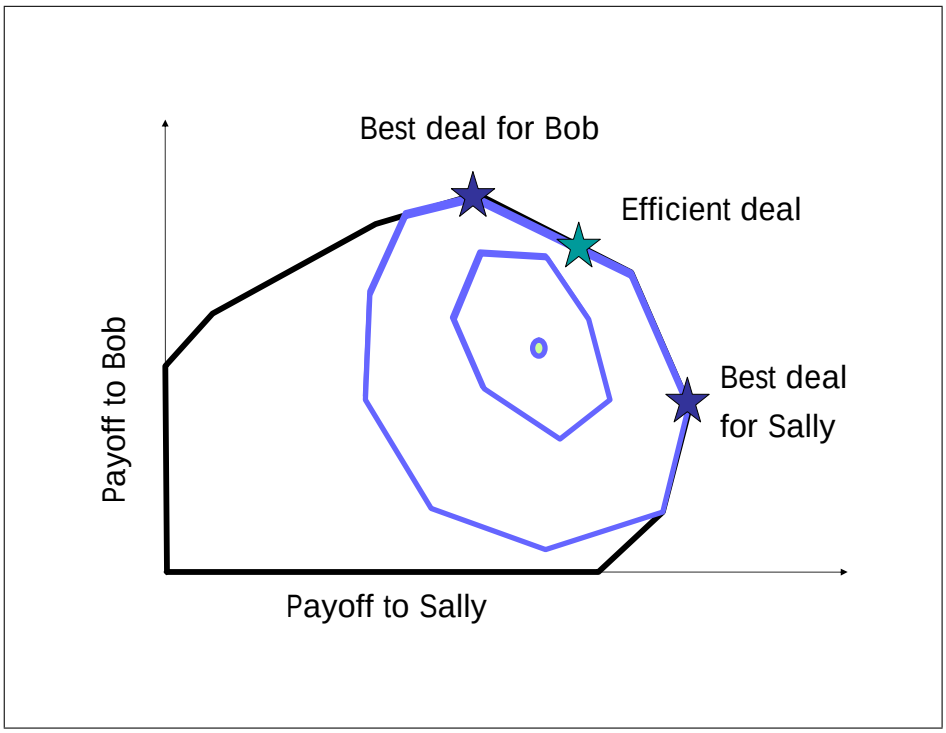

Figure 1: Narrowing the deal space.

Figure 1 illustrates the process. The axes represent the payoffs, of various deals to the negotiators Sally Seller and Bob Buyer. Bob wants the deal highest in the graph; Sally, the one farthest to the right. Points on the segment of the boundary between these two points are Pareto optimal, *i.e.*, what's better for one is worse for the other.

Every possible deal reachable from the initial offer is contained in the outer boundary, but not all points in the interior represent deals. The protocol specifies that a legal counteroffer must narrow the range. We see from the figure that the first counteroffer does not preclude reaching an efficient deal, but that the second one does. The final deal reached is quite far from the efficient frontier. Both players would benefit if either or both used a better strategy.

Another unique feature of the work presented here is the guaranteed termination of the protocol. The advantage is that there is no need to worry about cycles of repeated offers. Of course, there is a price to pay; some possible deals might be missed. A good strategy [6] should minimize the number of missed deals.

Automating negotiation carries risks, so we need to consider where it is an acceptable business solution. Fortunately, a system for automated negotiation can be used in a number of ways.

1. When the risk of a bad deal is less than the cost of involving a person,

2. When the maximum time available to reach a deal is shorter than people can handle,
3. To deal with the easy terms and conditions, leaving the rest to people,
4. As a guide to help people negotiate better.

These options provide more than enough opportunity to warrant further study.

Acknowledgements: I'd like to thank Claudio Bartolini for his careful reading of the draft and for pointing out several useful references.

References

[1] C. Bartolini, C. Preist, and N. R. Jennings, "Architecting for Reuse: A Software Framework for Automated Negotiation", Proc. 3rd Int. Workshop on Agent-Oriented Software Engineering, Bologna, Italy 2002

[2] P. Faratin, M. Klein, H. Sayama, and Y. Bar-Yam, "Simple Negotiating Agents in Complex Games: Emergent Equilibria and Dominant Strategies", in *Proceedings of the 8th Int Workshop on Agent Theories, Architectures and Languages (ATAL-01)*, Seattle, USA, pp. 42–53, 2001, also at http://ccs.mit.edu/peyman/pubs/ATAL-01.pdf

[3] FIPA Iterated Contract Net Interaction Protocol Specification, http://www.fipa.org/specs/fipa00030/SC00030H.html, 2002

[4] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge, "Automated Negotiation: Prospects, Methods, and Challenges", Int. J. of Group Decision and Negotiation, **1**0(2), pp. 199-215, 2001

[5] A. H. Karp, "Representing Utility for Automated Negotiation", HP Labs Tech Report HPL-2003-153, 2003, http://www.hpl.hp.com/techreports/2003/HPL-2003-153.html

[6] A. H. Karp, R. Wu, K-Y. Chen, and A. Zhang,"A Game Tree Strategy for Automated Negotiation", HP Labs Tech Report HPL-2003-154, 2003, http://www.hpl.hp.com/techreports/2003/HPL-2003-154.html

[7] H. Li, "A New Category of Business Negotiation Primitives for Bilateral Negotiation Agents and Associated Algorithm to Find Pareto Optimal Solutions", Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, Newport Beach, CA, June 2002

[8] C. Mudgal and J. Vassileva, "Bilateral Negotiation with Incomplete and Uncertain Information: A Decision-Theoretic Approach using a Model of the Opponent", in Klusch and Kerschberg (Eds.) Cooperative Information Agents IV, LNAI **1**860, pp. 107-118, Springer-Verlag, 2000

[9] B. Schneier, *Applied Cryptography, Second Edition*, John Wiley & Sons, 1996

[10] Soundex, http://www.nara.gov/genealogy/soundex/soundex.html

A Ordered and Unordered Attributes

Some attributes, such as price, are clearly competitive; what one party gains, the other loses. The situation for other attributes is not as clear, color, for example. The distinction between ordered and unordered attributes corresponds loosely to cooperative attributes and competitive ones. Ordered attributes correspond to those that are cooperative, in the sense that for a particular attribute value either both parties gain or both lose. Unordered attributes are those for which there is a conflict for all values of the attribute.

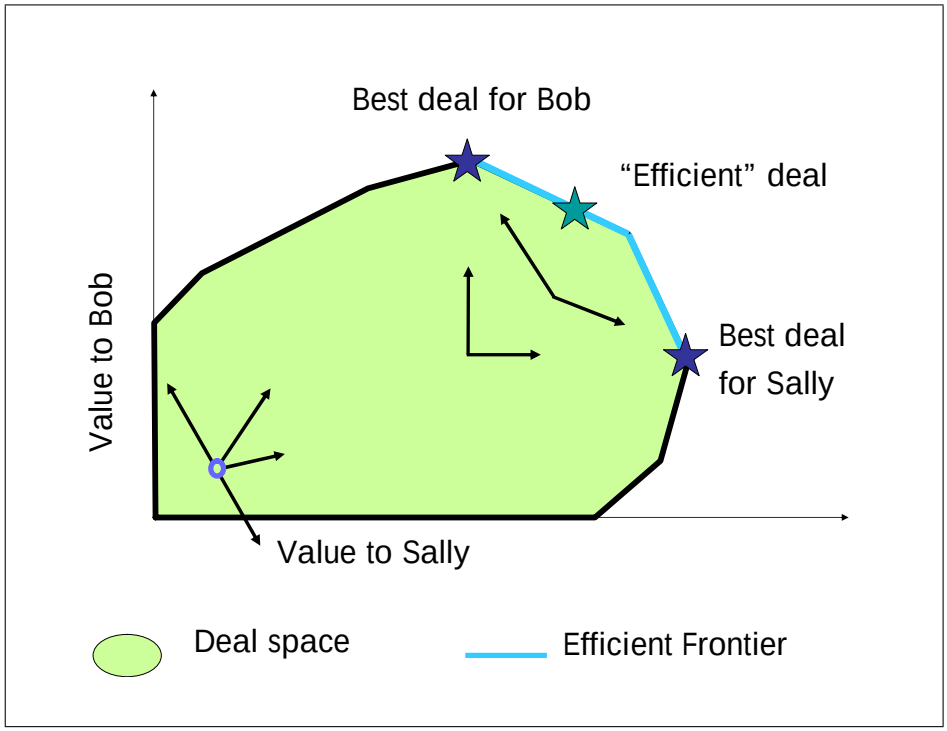

Figure 2: Effect of moving through the deal space.

We can understand this issue by looking at Figure 2, which shows the space of possible deals between Sally Seller and Bob Buyer. Sally's value increases to the right; Bob's, upward. Clearly, the best possible deal for Sally the deal farthest to the right, and the best for Bob is the highest point. Points on the part of the boundary between these two deals, called the *efficient frontier*, are Pareto optimal, which means that neither party can do better without the other doing worse. Both parties can improve from any other deal.

Consider now the deal represented by the point near the origin. There is a lot of room for improvement. If we change the value of an unordered attribute, such as price, we move up and to the left or down and to the right. Either change benefits one party at the expense of the other. On the other hand, if we change an ordered

attribute, such as color, we may move up and to the right, benefiting both Sally and Bob. Even if Sally prefers red, and Bob prefers blue, a deal for purple might be better for both than the current deal.

Of course, unless both parties want the same thing, no attribute is purely cooperative. Consider the deal near the efficient frontier. Most changes that don't penalize both parties benefit one party at the expense of the other. Perhaps purple is no longer an option, and the only choices left are red and blue.

Because of the change in cooperative tension of such attributes, we call them *ordered*. Although we might be revealing information useful to the other party, we can use the ordering of these attributes to reduce the negotiation to consider only the Pareto optimal subset of values[7]. The result might be better if reaching a deal faster results in a net gain. An example of this benefit is shown in Section 5. At any rate, the risk is small because there is no indication of the strength of the preference.

There are also cases that are not as clear-cut. For example, the store may not have the color shoes I want, yet they may provide them, say by buying them from a competitor. In this case, color becomes a competitive attribute because the store may lose money on this deal. Clearly, knowledge that such a case exists can affect the negotiation strategy. It might also be the case that the vocabulary designer doesn't know if a particular attribute is cooperative or competitive. Delivery time may be cooperative, if both parties benefit from a shorter delivery time, or competitive if one benefits and the other is penalized. Fortunately, correctly specifying whether attributes are ordered or unordered should only have the effect of shortening the negotiation, reducing the impact of an incorrect designation. It is also possible to extend the protocol to allow negotiators to signal whether the attribute values are ordered or unordered.