Chapter 1

ADAPTIVE RESPONSE GENERATION FOR DECISION SUPPORT

The previous chapters focused on the problem of generating effective messages in real-time decision support. I presented my solution, TraumaGEN, a text generation system that takes into account an arbitrary and often inter-related set of communicative goals and produces a message that realizes the entire set in a concise and coherent manner. TramuaGEN takes into account the purpose of the messages, the situation in which the messages will be received, and the social role of the system.

However, a decision support system must be adaptable to a wide variety of users in different situations. The kinds of information that will be most useful for decision-making will vary depending on the individual. For example, in the financial domain, some users might be most concerned about the riskiness of an investment while others are more interested in an investment's prospects for financial gain. In addition, constraints or priorities on resource usage will differ. For example, some users would be willing to pay substantially for an expert opinion, while others would prefer to spend less money even though their information will come from a generic news source. Different individuals will have different priorities, and those priorities will change over time.

To understand why user priorities are essential to decision support, consider a system that does not take them into account. For example, a system that does not vary its responses to reflect user constraints on length may provide the user with a message that the user does not have time to read, memory to store, or room to display for viewing. A system that ignores the monetary cost of a response might arrive at a message that the user cannot afford (or, conversely, a message so cheap that the user will not value it). And a system that does not consider time may provide a message whose usefulness expired before its delivery (e.g. a message about a stock purchase delivered after the close of the market).

The situation is complicated when the decision support environment is not static. In addition to user priorities changing, a user's status may change, or the environment could change in a way that affects a user's relative status. For example, a user's ownership of an asset such as land may be static, but a flood or a change in zoning laws may change the user's status by affecting the value of the land.

Thus the problem is to

1. develop a response generation methodology that takes into account information about the user, the user's resource constraints, and the user's priorities (with regard to information content and resource usage);
2. make the methodology responsive to internal (user) and external (environmental) dynamic factors; and
3. implement and evaluate the methodology.

The remainder of this chapter outlines my solution: a decision-theoretic agent-based approach to response generation for decision support that ranks different possible full responses according to their value to the user, and takes into account both resource and content attributes in doing so. The system assumes a dynamic environment and poor predictive models of ultimate information utility, and thus requires dynamic organizational management in response to run-time results, necessitating a distributed, adaptive system. This methodology has been implemented and tested in the system MADSUM (Multi-Attribute Decision Support via User Modeling).

1.1 An Overview of MADSUM

MADSUM is the result of my investigations into the design of a decision support system that can adapt to a user's resource constraints, resource priorities, and content priorities in a dynamic environment. The design combines approaches from both user-modeling and agent architecture. The implemented system consists of a hierarchy of cooperative agents that use a negotiation process to solicit and organize other agents to produce information, and a presentation assembly process to coherently assemble the information into text for decision support. At each stage the decisions of the agent consider the preferences and constraints represented in the user model.

1.1.1 Multi-Attribute Utility Function

The user model includes a multi-attribute utility function. Attributes are chosen for a particular domain, though many domains share attributes such as dollar cost. Constraints are hard limits on the possible values of attributes set by the user to prevent the system from generating unacceptable results. Constraints are employed to determine if a particular result is acceptable, but provide no guidance to the system about which acceptable decisions are preferred over others.

In contrast, the utility function allows the agent to weigh the benefit of different decisions about resource usage and information selection that do not violate constraints. In other words, utility provides a means to evaluate decisions, while constraints limit the decision space. The user of the system can tailor the utility function to affect:

- the information content of the result (which kinds of information are included, and how much);

- attributes of the resulting message itself (such as text length and cost); and
- attributes of the planning process (e.g. time).

This approach provides a structure in which the priorities of the user can be explicitly represented and considered in light of the environment (information currently available, the cost of getting the information, etc.). Furthermore, the use of an appropriately designed agent architecture allows the system to dynamically respond to changes in the environment and/or user priorities.

1.1.2 Agent Architecture

An agent architecture is appropriate for this problem for several reasons. First, the nature of the information gathering problem is distributed, and a distributed architecture can solve this class of problems utilizing parallelization, distributed expertise, and fail-soft performance[Jen95, SV00].

Second, MADSUM is intended to be easily adapted to new domains. The distributed approach is an ideal way to decompose a decision support task so that previously designed MADSUM source and task agents can be re-used without modification in domains that require expertise which overlaps with previously implemented domains. Also, new agents can be easily added to existing domain implementations as new sources or new areas of content expertise are added (this can facilitate the incorporation of legacy code).

Third, in agents as in business, distributed decision-making is ideal for rapid reaction to a dynamic environment[**?**, **?**]. In the case of decision support, the dynamic aspects of the agent environment can be thought of as consisting of both the external world of information and information sources, and also the internally-modeled user preferences regarding constraints and priorities.

Finally, the use of a hierarchy of independent agents avoids the information bottleneck associated with having all information processed by a single agent. Reducing the size of the problem makes individual decision makers simpler and more reliable[Jen95] and can facilitate parallelization.

1.2 The Financial Investment Domain

I have applied the MADSUM architecture to decision-support in a financial investment domain. MADSUM provides investment information to a user making a buy/don't-buy decision on a single investment, i.e. a specific amount of a certain instrument at a certain price. The MADSUM decision making algorithms and the agent hierarchy, communications, and interaction are domain-independent (see Chapter **??**). Extending MADSUM to a new domain requires two modifications. First, the set of attributes in the user utility function must be altered to reflect the user preferences of the new domain. For example, decision-support in a restaurant domain might require utility function attributes regarding decor and food quality[MFLW04].

The second modification is the implementation of a set of domain-dependent information agents. For example, tailored decision-support in an investment domain requires domain-dependent agents that can estimate how significant a particular piece of information will be to the current user, given her current personal and financial status. In MADSUM these are "wrapper" agents that act as interfaces between the actual information source (possibly a website, a database, or an external agent) and the MADSUM system.

MADSUM is implemented in a financial investment domain. I chose this domain for the following reasons:

- While full-scale investment advisement involves developing a financial plan, the plan eventually decomposes into binary buy/don't buy decisions that reduce the complexity of the decision support problem.

- Financial domain decision-making typically includes a large numeric component, which is composed of the computable answers to multiple, independent questions about a particular investment or investment strategy. This facilitates the design of source and expert agents, since there exists an accepted body of standard methods of numeric analysis. (Contrast this with decision support for buying art, or adopting a child.)

- It is a domain that has a huge web presence (typing "investment advice" into Google recently retrieved 21 paid listings and over 17 million results), thus providing many examples of actual information sources available and their products.

- Many people are interested in making investment decisions to some degree due to the prevalence of 401K accounts and similar vehicles that allow market investments.

- It is a domain with which I am familiar, due to my experiences in the banking industry and contacts with people in the field.

The financial investment domain information can be viewed from many perspectives, resulting in a wide variety of classifications. For the purpose of implementing MADSUM, I chose three broad content areas for which information would be generated: investment risk (hereinafter RISK), potential for increased value (VALUE), and the impact of the investment on the user's financial goals (GOAL)(as stated or hypothesized from the user model). Within those three areas, I selected standard financial analysis measures (all available from free financial websites) to be the information provided by MADSUM source agents.

1.3 The User Interface

A user employs MADSUM via a graphical user interface. To minimize the complexity that the user sees at any given point in the interaction, the interface has three distinct parts, each a separate screen. Most user interactions will involve only the second part.

The first part consists of the user entering data for the user model (Figure **??**). This consists of financial portfolio information, financial goals, and limited personal data (e.g. years to retirement; see section 2.3.1).

The second part of the interface is the heart of the user-MADSUM interaction. Figure **??** shows the graphical sliders employed by the user to indicate relative priorities for the attributes in the utility function. The user can also enter numeric high and low constraints for the attributes, though these come with preset defaults.

This screen also presents the user with windows for proposing a particular investment for which they desire decision support. The user enters a stock symbol (selects from a menu?), a number of shares, and a price per share. Clicking the "GO" key forms a message[1] which is sent to the MADSUM Presentation Agent to begin the process. At the conclusion of the process a new screen shows MADSUM's resulting message.

More advanced features of the MADSUM interface are accessed via the third screen (Figure **??**). Here the user can modify the separate functions that control how utility is derived from each attribute term (see section 2.3.4). This allows utility for different attributes to have soft constraints or various behaviors over the range of the attribute.

[1] see Chapter **??** for information about the messages used by DECAF agents.

1.4 Summary

Individuals differ not only in the resources they have available to expend on information, but also in the priorities they place on different kinds of information. A decision support system must represent these differing priorities and related constraints in a user model, and then use that model to allocate resources for an unseen task across multiple agents in a dynamic environment. MADSUM is a distributed adaptive system that uses a negotiation process to solicit and organize agents to produce information, and a presentation assembly process to coherently assemble the information into text for decision support. Chapter 2 explains how a user model, including preferences and constraints on both content and the resulting message, informs both processes. An evaluation demonstrates that the influence of the user model on content selection and presentation improves system output. Chapter **??** then describes the aspects of the agent architecture that allow MADSUM to dynamically adapt to the runtime environment, and includes experimental data showing that the organization responds appropriately and predictably in the presence of inevitable information failures.

Chapter 2

A DECISION-THEORETIC APPROACH TO RESPONSE GENERATION

2.1 Introduction

As discussed in the previous chapter, an effective decision-support system must produce responses that are tailored to the individual user. In producing such responses, the system must not only take into account a user's preference for different kinds of information but also a user's priorities and constraints on the usage of different resources. This chapter presents my solution to adaptive response generation, which takes the form of a decision-theoretic approach that utilizes a formal utility function to rank different possible full responses according to their value to the user and that takes into account both resource and content attributes.

I first discuss work in areas related to the generation of responses that are adapted to a user's preferences. I then explain how MADSUM models user preferences regarding types of information content included in a response and attributes and constraints of the response itself. Finally I present evaluations of two aspects of MADSUM's output, showing that subjects presented with a user's preferences agreed with MADSUM's decisions regarding content selection and ordering.

2.2 Related Work

2.2.1 Adaptive Response Generation

MADSUM follows work in other systems that adapt their responses to an individual user. The ADVISOR system [MWM85] operated in a domain of student advisement, giving information about what courses a student should or could take to satisfy a particular goal. The system inferred a student's goal from a discourse segment, ranking the goals as either *definite*, *likely*, or *plausible* based on whether they were explicitly noted or could be repeatedly inferred. One of several information hierarchies is selected based on the detected topic of the last question and the current *relevant* user goal, and the different contents of the hierarchies trigger different rules, which determine the output.

Paris [Par88] modeled a user's level of expertise, and varied a response not only by modifying the content, but also by modifying presentation style. MADSUM does not make any high level decisions about style of presentation, but leaves that determination to agents that create a text plan about a specific piece of information at a low level. In principle, at least, style choices could be made at that level in a manner similar to Paris'.

McCoy used a detailed, pre-existing user model to design a response when the system detects that the user has a misconception. This work was the first to address the "overlay" assumption issue, i.e. to consider the possibility that the user's knowledge/beliefs are not a subset of the system's knowledge, but may simply intersect (or, as in RTPI, may be a superset).

All of these early systems modified the system response to a user based on a user model, either pre-existing or acquired. Section 2.2.2 considers work that specifically models the preferences of a user (as opposed to e.g. a user's goals, knowledge, beliefs, or attention state).

2.2.2 Modeling User Preferences

A very early system using a model of user preferences, GRUNDY [Ric79] employed stereotypes to instantiate individual user models to make book recomendations. This strategy reduced the number of questions that the system had to ask the user before it had a complete enough user model to recommend a book. Individual facts, such as gender, would expand via stereotype to imply preferences for or against books that were romantic, violent, intellectual, etc. These preferences would be assumed correct until the system had contrary information (of a stronger nature: the system also kept track of why it believed what it did) about the individual's preferences, at which point the model would be updated.

Elzer et al [ECCC94] focused on identifying user preferences dynamically during a dialogue. Her system could identify preferences stated by a user, or infer them by examining a selection of candidate options rejected by a user. Once the system was confident that it had a sufficient model of the user's preferences, it was capable of determining an action plan that better fit those preferences than the user's original plan.

A more recent effort at acquiring user preferences re-examined the problem of asking the user questions to determine preferences. Chin and Porage [CP01] use multi-attribute utility theory to make travel choices for a user. At each step in the decision process, their system chooses the next question based on maximum information gain (i.e. the best reduction in utility uncertainty). Work of this kind may eventually facilitate the use of systems like MADSUM that require user preferences to operate.

2.2.3 Tailored Response Generation

Adaptive systems have used concepts of utility theory, either informally or formally, to make decisions that take into account the user's preferences. Chu-Carroll [CCC94] has dialogue agents collaborating in a recursive *Propose–Evaluate–Modify* cycle to form a plan. Chu-Carroll did not use the term "utility", but in fact her method for evaluating plan alternatives is a summation of attribute preferences multiplied by a value (a distance from a target), or what is now called a utility function. Lesh[LHL97] uses a similar model in ranking candidate flights in a travel domain.

More recently, researchers have begun to design their responses to incorporate user preferences by using formal multi-attribute utility functions.

MATCH [JBV$^+$, SWWM02]consists of a sophisticated user interface on specialized hardware for restaurant choice based on user preferences expressed in utility function. The architecture is intended for use in other domains as well; however, several design choices limit the kinds of domains for which MATCH would be suitable. In particular:

1. Mapping of attribute values is uni-directional, e.g. increased cost is always bad; while this makes some sense in the restaurant cost domain, it is not suitable for attributes in other domains. For example, MADSUM allows a user to express that increased length is good up to a point, then bad beyond that point.

2. MATCH uses ordinal assignment of weights, versus the arbitrary weights in MADSUM. MATCH's SMARTER [EB94] procedure for eliciting multi-attribute decision models cannot express e.g. that risk and value information are both of equal high value. While this is not a critical issue in a domain like restaurant choice, it could be unsatisfactory to a user making critical decisions about money or health options.

3. The MATCH utility function computes the utility of each *restaurant* to the user, not the utility of the system's *message* to the user. MATCH is principally an argumentative system, making a choice and attempting to convince the user that the choice is correct. MATCH may choose to leave important counter-arguments out of its presentation if those arguments are not considered part of the most persuasive message. This is appropriate when the user's knowledge

and expertise are a subset of the system's, but not when the user has final responsibility for making a critical choice.

4. MATCH "agents" are not independent decision makers, and thus not autonomous agents in the sense of [WJK99]. Rather, they appear to be separate program functions that can run on different parts of a network if necessary, passing information via XML structures. In particular, crucial parts of MATCH appear to be "distributed", e.g. the text planning module

FLIGHTS ([MFLW04]) uses a multi-attribute utility function in a user-model to influence the generation spoken language of airline flight recommendations. Like MADSUM, FLIGHTS applies the influence of the user-model at multiple stages of the process. But in FLIGHTS, as in MATCH, the user-specific weights are strictly for domain information, such as whether a business class seat is available.

While each of these systems makes use of a utility function, they each use utility function to compute the utility of the expected *result* of a choice by the user - the utility of a certain train ride, or a particular airline flight. In contrast, MADSUM's utility function explicitly includes user influence on attributes of the system's output, in terms of cost, length, and time, as well as domain-specific preferences such as topic. Thus MADSUM is concerned with the total utility of the *message* to the user, including the utility of the both information provided as well as the the utility of the attributes of the presentation.

2.3 Modeling User Preferences and Priorities

A user affects MADSUM's message output by specifying preferences and priorities with respect to a set of predetermined message attributes. This specification is performed in the user interface (see 1.3). MADSUM is designed to utilize an arbitrary number of attribute preferences, since the number of attributes will vary for different domains. For my implementation I selected the attributes of text

LENGTHand dollar COST, and three domain specific attributes, one for each of three topics in the financial investment domain (see 1.2). These topics are RISK(the riskiness of an investment), VALUE(the prospects for the investment gaining in value), and GOAL(how the investment relates to the individual's portfolio allocation goals). Agents categorize all information as belonging to at least one of these topics, and lower level agents are grouped by topic area (see 2.6).

All information about user preferences from the user interface becomes part of the user model. The next section explains the user model and its components, and the ways in which these components affect MADSUM's message ouput.

2.3.1 The User Model

Adaptive response generation requires a model of the individual user. User models can have both long-term and short-term components [KF88]. Long-term components are assumed constant during a single decision support task, and include data such as user age and portfolio allocation goals. Short-term components may vary during the user-MADSUM interaction, and include preferences like length and cost.

MADSUM was designed to exploit a user model in the context of multi-agent decision support. The user model has three components: User Attributes (long-term), Constraints (short-term), and a Utility Function (short-term). Although User Attributes might be captured in a long-term user model that is constructed and revised over time (as in **??**), the Constraints and the Utility Function will vary with different user interactions and perhaps even change during an interaction.

2.3.2 User Attributes

The User Attributes component of the user model captures characteristics of the user, including appropriate domain-specific information. For the financial investment domain, this component of the user model includes the user's:

- age
- salary
- expected number of years to retirement
- approximate annual expenditures
- existing investment portfolio
- portfolio allocation goals (Portfolio allocation goals refer to an individual's desired distribution of investment categories, such as stocks, bonds, or cash equivalents.)

If not expressly given, portfolio allocation goals can be hypothesized based on a simple stereotype ([Ric79]) derived from an individual's personal characteristics such as age, salary, and years to retirement. For example, investment advisors typically recommend that people close to retirement age maintain a progressively smaller percentage of their portfolios in stocks.

The User Attributes are used by agents to affect the significance of certain pieces of information (see 2.3.4). For example, if a proposed investment would cause one's investment portfolio to deviate from one's portfolio allocation goals, information about the deviation becomes more significant as the deviation grows.

2.3.3 Constraints

The Constraints component of the user model offers the user the option of setting hard constraints for a given attribute. Hard constraints are values that an attribute *must not* exceed in a response, and are used to pare the search space before utility is calculated. For example, if the user sets 75 words as the *hard constraint* for length of the response (perhaps because he is using a handheld device with a miniature viewing facility), then longer responses will not be considered by the system.

The user also has the option to set soft constraints for certain attributes. Soft constraints are attribute values that the user would prefer not be exceeded in constructing a response. If the user sets 75 words as a *soft constraint* rather than as a hard constraint, then the estimated utility of the response will depend in part on how much the length of the proposed response exceeds the soft constraint. Soft constraints must be implemented as part of the utility function to operate in this manner, and are further discussed in 2.4.0.1.

2.3.4 Preferences

User preferences (that are not hard constraints) influence MADSUM output via the utility function. A user expresses preferences about one or more attributes of the output. MADSUM is designed to accept an arbitrary number of attribute preferences, but for my implementation I selected the attributes of text length (Length) and dollar cost (Cost), and three domain specific attributes, one for each of three topics in the financial investment domain (see 1.2). These topics are Risk, Value, and Goal. With these the user can indicate his or her preference for information on a certain topic within the broader category of information about the investment under consideration. For example, a user can set the Risk slider high to indicate that, other things being equal, they would prefer the final message to contain some information about Risk (see 1.3).

The slider for each preference covers a one to ten continuum. The output of the slider, as a real number, is included in a message to the Presentation agent (see 2.6) that starts the decision support process. Each message attribute is represented by a term in the user's utility function, and MADSUM uses the user preference number from the slider as a coefficient for that attribute term (see 2.4). The rest of the term consists of the actual attribute value and a function that maps it into a 0..1 valued space, discussed in 2.4.0.1.

The attribute value is either a hard number, as for LENGTHand COST, or a number representing information significance for a topic preference (see 2.4). Attributes can capture characteristics of propositions that might be presented to the user, and for information significance the value captures the significance of a set of propositions to the decision task at hand — i.e. its significance in the environment of the user's personal characteristics and the application domain. I have termed this approximation *Decision Specificity* or *DS*. Determining DS is a domain-specific task, and thus in the MADSUM architecture, the functions that compute DS are provided by the application designer as part of the domain-specific information agents that propose propositions for inclusion in the response to the user.

In the financial investment domain I have implemented such domain-specific information agents within three categories of information: RISK, VALUE, and GOAL. The associated decision specificity functions produce estimates of significance that are referred to as DS_r, DS_v, and DS_g respectively. It is important to note that DS corresponds not with the precise attribute value but instead with the significance of the information. In the financial investment domain, for example, the significance of a company's debt-to-equity ratio depends on both its absolute debt-to-equity ratio (heavy debt is bad) and the particular industry (high debt is more acceptable in utilities than in manufacturing). Thus a .8 debt-to-equity ratio would be noteworthy in some instances but not in others. For example, suppose a source agent, DebtEquity, obtains the debt-to-equity ratio for company XYZ, 0.6, as well as the industry average debt-to-equity ratio of 0.8, and sends it to the task agent (its wrapper) DebtRisk. DebtRisk uses a domain-specific and ratio-specific formula[1] to

[1] The particular equations used by each domain agent to determine DS are not important to my work. Financial analysis texts I examined only associated specific ratio values with verbal descriptions of their import, so I extrapolated

calculate how good or bad XYZ's ratio is in light of the industry ratio. In this case, a value of 0.6 means that for a company of its size[2] XYZ has substantially lower debt than is average in its industry. DebtRisk expresses the significance of the result of this comparison by assigning a DS of 4.83. This is a very significant level of DS, indicating that the information (about the *relative* value of XYZ's debt-to-equity ratio and the industry average ratio) is likely to be of significant value to the user in making a decision about an investment in XYZ (see 2.5 for information on how DS is used in assembling text plans).

Similarly, the significance of a proposition from the Portfolio Agent that addresses the relationship of a proposed investment to the user's portfolio allocation goals depends on the extent that the investment would cause the user's portfolio allocation to deviate from his goals, while a proposition that addresses the appropriateness of the investment from an age perspective may depend on how close the user is to retirement. In my system, I currently compute the DS of a set of propositions in a particular category (RISK, VALUE, or GOAL) as the sum of the DS values for the individual propositions; this has worked well in my application but further experimentation is needed to fully validate the decision.

This system is subject to misunderstandings if the user is not familiar with the concept of relative preferences, as found in utility functions. All preferences within the utility function are relative to one another, and so setting all sliders high, for instance, does not give the system any guidance. Also, setting improper constraints can prevent the system from responding as a naive user might expect. For example, setting a hard constraint for LENGTHof ten words and then setting high sliders for all three topics will not result in a ten word answer with three topics, since most single topic phrases are longer than ten words.

numeric formulas from textual descriptions in []

[2] Actually, for a company of its level of capitalization...

Figure 2.1: Rough Drawings of Base Utility Functions (see text for key)

2.4 Multi-Attribute Utility-Based Evaluation

The intent of every decision support system is to be effective, but effectiveness is in the eye of the beholder. Specifically, the needs of the user in the context of a particular environment determine whether certain information will be deemed supportive or of little worth. Utility is a hard number used during planning to approximate effectiveness given (necessarily) incomplete models of the user and the environment.

MADSUM's utility function contains n attribute terms, each consisting of a weight w_i giving the importance of that attribute to the user, a parameter a_{value_i} that is related to the value of the attribute, and a function f_i.

$Utility = \sum_{i=1}^{n} w_i f_i(a_{value_i})$

The weights w_i, giving the importance of each attribute to the user, are extracted from the positions of sliders that are manipulated by the user in a graphical user interface. For resource attributes such as length of response or processing time, a_{value_i} is the actual value of the attribute, but as noted previously, for topic DS attributes a_{value_i} captures a numeric representation of significance to the decision at hand.

Each of the functions f_i that appear in the utility function map their parameter a_{value_i} into a utility value between 0 and 1. The particular function f_i that is used determines whether an increasing parameter value increases or decreases utility (and at what rate). Figure 2.1 illustrates the currently implemented functions in MADSUM's predefined library of utility functions. Each function can also take an additional argument (which the user can set in the advanced features section of the user interface); the effect of the additional argument varies according to the function, and is noted in the function descriptions below:

1. $f_{StartPlateauNorm}$ captures instances in which utility remains high over a plateau and then decreases for increasing values of its parameter; the additional argument locates the rightmost endpoint of the plateau and specifies the spread of the remaining curve.

2. $f_{EndPlateauNorm}$ is same as above, except that the curve rises to a plateau.

3. $f_{EndPlateauLinear}$ captures instances in which utility increases linearly for increasing values of its parameter until a plateau is reached at the point specified by the additional argument.

4. $f_{StartPlateauLinear}$ is as above, except with the plateau at the beginning.

5. $f_{EndZeroPlateauLinear}$ captures instances in which utility decreases linearly for increasing values of its parameter until a zero-valued plateau is reached at the point specified by the additional argument.

6. f_{Normal} approximates utility as a normal distribution of the possible values of its parameter. The additional parameter specifies the center of the distribution and the spread.

My financial investment domain by default uses $f_{EndPlateauLinear}$ for information attributes, and $f_{StartPlateauNorm}$ for resource attributes.

2.4.0.1 Soft Constraints

The user specifies soft constraints by using the appropriate f_i and then using the additional argument to determine its shape. The user interface shows how the additional argument relates to the f_i in question. For example, the function $f_{StartPlateauNorm}$ is used by default for the resource attribute of processing time; the soft limit determines where the plateau ends and also the rate of fall in utility after the plateau (the falling portion resembles a normal distribution whose spread is 1/2 the soft limit). This captures the notions that 1) the soft limit on processing time set by the user is the point at which the utility of the response will begin to decrease and 2) the larger the soft limit on processing time, the less severe will be the loss of utility for each second of increased processing time.

MADSUM has an easily extendable library of base utility functions. For example, instead of having the utility of length remain the same until the soft limit is reached, the user might want to say that utility increases with length of the response until the soft limit is reached and then decreases; such a user might select the base function f_{Normal} to evaluate the contribution of length to the overall utility of the response.

Thus the magnitude of a term in the overall utility function is affected by the value of the attribute in the environment (either its actual value in the case of resource attributes or the DS value computed from propositions in the case of information attributes), the base utility function f_i which is typically selected by MADSUM but which can be selected by expert users (under the advanced features of the user interface), and two user-selected modifiers (the weight w_i that gives the importance of this attribute to the user, and the additional argument that adapts the function f_i). Allowing the user flexibility in designing each term's contribution to utility ensures that the resulting utility function reflects not only the attribute value, but also the user's opinion of how the attribute contributes to utility.

2.5 Assembling Text Plans

Chapter **??** discusses the agent-based architecture in which my system collects and integrates information from distributed sources. In this section, I discuss the rules that are used to organize individual pieces of text into a coherent framework, ignoring for the moment the agent architecture. The coherence rules are based upon work by other researchers in natural language generation and are not the focus of my work.

2.5.1 Related work

2.5.1.1 This is the Related Work section for RTPI.

I include it all here since otherwise it will seem that I am leaving out all the basics.

There is currently no other work on assembling or integrating multiple, independently generated text plans. However, there is related work in the areas of text structure, text planning for generation, and text plan revision.

Fundamental to my development of RTPI is a theory of the functional analysis of text called Rhetorical Structure Theory, developed by Mann and Thompson [MT83, MT87]. According to RST, a text contains *relational propositions* that are conveyed by the way the text is structured. The structure of a text (derived the arrangement of clauses and the presence of discourse markers) is necessary to determine meaning, and the meaning of a text cannot be derived from simple composition of the semantics of lexical items.

The relational propositions that exist in the text are represented in the RST formalism by *relations.* According to Mann and Thompson, relational propositions possess the following properties: they are not explicitly expressed in a clause; they can be signaled via discourse markers, but often are not signaled at all; one relational proposition corresponds to one RST relation in the text structure; and relations are capable of performing rhetorical acts, such as changing a reader's beliefs or desire to act. Finally, "The relational propositions are essential to the coherence of their texts. Perturbing text to prevent the (implicit or explicit) expression of one of its relational propositions causes the text to become incoherent."

Relations hold between adjacent text spans, and are defined by a list of constraints and effects. Most relations are defined in terms of a nucleus and a satellite, where the nucleus is a span that remains comprehensible in the absence of the satellite. Restrictions on the way that relations are applied ensure that coherent texts (with a few special exceptions) will have an RST structure that is a tree of text spans joined by relations.

The contribution of RST most useful to text generation is the idea that recognizing the relations in an RST structure is equivalent to finding the basis for the text's coherence. RST also works at both the inter-clause and inter-sentence levels, allowing a single planner to plan both sentences and paragraphs. Hovy [Hov88, Hov91] exploited these properties, constructing text plans that were coherent because they had been built using the principles of RST.

Hovy developed text planning operators based on RST relations, and used them to plan text structures from the top down. Each operator used constraints to determine what content could be used to fill the nucleus and satellite roles of a given RST relation. Operators also had growth points, optional ways to extend the structure that could be posted as further goals for the planner. The user's communicative goal is achieved by matching it against the RESULTS field of an operator. The operator posts sub-goals, and the process continues so that a tree of relations is built with small text spans as leaf nodes.

Hovy's planner used operator constraints to select content to address a top-level goal, and then used the growth points to add as much additional information as possible. Thus only the first step of the process was directly related to the top-level goal of providing a particular piece of information. Successive steps were influenced by the available data[3], since that determined which growth points worked, and by the choice of growth points associated with an operator. Because the planner selected the text plan that included the highest number of propositions, it always "said" as much of the data as possible; thus an explicit goal of expressing all knowledge wasn't necessary. On the other hand, it was not possible to ensure that all data on

[3] The "available data" was a set of clause-sized, structured information units constructed from a less structured data base using domain-specific rules

a subject would be expressed.

Moore and Paris [MP90, MP93] also used RST relations to do top-down text planning. Their operators matched their effects against a goal posted above, and posted a nucleus and satellite combination designed to satisfy that goal. The nucleus and satellite became the new goals (until they became atomic actions and were designated leaf nodes). Like Hovy's growth points, the "knowledge" of how the tree could expand below a given goal was determined by the operator's nucleus/satellite combination. Unlike Hovy's planner, the Moore and Paris planner only employed an operator when it was necessary to satisfy a goal. Thus their system generated only enough text to satisfy the top-level goal, and favored concise trees over larger ones.

Moore and Paris recognized that for the resulting plan trees to be useful in a dialogue system, the plans would have to include not only the relations between spans of text, but also the intended effect that the chosen relation was supposed to have on the hearer. This property facilitates the process of designing a system response when the intended effect does not occur.

ILEX [MOOK98] is a bottom-up, opportunistic RST-based planner. Objects in the knowledge base are linked via RST-style relations, and the system constructs a tree by searching links starting from the desired topic. The structures are limited to relations that are already known to exist between items in the knowledge base. In contrast, my text plan integrator can superimpose certain new relations, such as CONCESSION, between parts of existing trees; this can be done because the new relation doesn't represent domain concepts, but a relation between the intentions in the tree structures.

Marcu [Mar97a, Mar97b] also builds RST-style structures in a bottom up fashion, and in contrast to ILEX, can have a top-level communicative goal of communicating all of the propositions in a knowledge base. Like ILEX, however, the system depends on having all relations encoded into the knowledge base. If some subset of the knowledge base is not connected to the rest of the knowledge via pre-coded relations, then a goal to communicate all of the information will not succeed. Furthermore, while the plan may express every proposition in the knowledge base, it does not attempt to express every relation between propositions in the knowledge base; so information contained in the *relations* of the knowledge base, such as NV-CAUSE, can be lost when a proposition is related to more than one other proposition.

Existing RST-based generation systems, like those above, function in explanatory or descriptive environments, not decision support, and the difference in environment was reflected in the tasks the systems performed. For example, while Moore and Paris' system had plan operators to convince a user to perform an action, decision support can (and RTPI does) require operators to convince a user that they should *not* do a certain act.

Another difference between the decision support environment and that of the RST-based explanation systems is the latter assume that system and user beliefs follow the overlay assumption: the beliefs and knowledge of the system are a superset of those of the user. Decision support cannot use this simplifying assumption, since it must contend with conflicting beliefs and goals. In particular, RTPI has to integrate plans that are designed to convince a user to adopt a goal, or to abandon a previously held goal.

Domain-independent text planning rules like the ones used by my text plan integrator are not a new idea. Appelt [App85] used "interactions typical of linguistic actions" to design critics for action subsumption in KAMP. After a plan had been generated from a single top-level goal, the plan would be "reviewed" and the critics could opportunistically subsume small clauses. If there were two communicative goals `(KnowsWhatIs John Tool(B1))` and `(Active Tool(B1))`, both could be satisfied by a single mention of `Tool(B1)`. Thus a single referring act could satisfy multiple communicative goals, though the system was limited to spans of one or two sentences. My text plan integrator's rules, instead of working only within clauses, work between text plans that encompass sentences and paragraphs.

HealthDoc's data-driven text planner is also rule-based[WH96, HDHP97]. It starts with a "master document" and then selects portions and repairs the resulting plan. Its rules can remove redundancy by aggregating neighboring expressions, but it does not address the aggregation of communicative goals (often requiring reorganization), the revision and integration of text plans to remove conflict, or the exploiting of relations between communicative goals as done by my text plan integrator. In contrast, my integrator's rules examine the communicative goals in a set of text plans, and certain rules may aggregate or subsume goals, or insert new goals in the process of improving the set's coherence and conciseness.

REVISOR [CL97] is based on RST. As the name suggests, it is not a full planner, but makes revisions (in the form of clause aggregation) to an existing text plan. An important similarity to my work is that REVISOR makes its modification decisions without detailed semantic or lexical knowledge of low level constituents. However, REVISOR uses a minimalist representation of a discourse plan that removes all but the most essential semantic features for processing. In contrast, RTPI uses the entire plan tree, allowing the introduction of new rules without the concern that information previously considered unimportant will be missing. And because my rules operate on full RST-style text plans that include communicative goals, the rules can be designed to integrate the text plans in ways that still satisfy those goals.

WISHFUL [ZM93, ZM95] includes an optimization phase during which it chooses the optimal way to achieve a set of related communicative goals. However, the system can choose to eliminate propositions and does not have to deal with potential conflict within the information to be conveyed. Similarly, STREAK [Rob94] is not required to include all information in a plan. Its rule-based text plan revision component is given a single text plan containing "obligatory" facts, along with a set of optional "supplementary" facts to be opportunistically included if realization constraints are met. In contrast, my text plan integrator and MADSUM agents both operate on multiple text plans *after* a domain expert has determined what information is vital to the communication.

END of RTPI related work.

Like RTPI described in Chapter **??** the theory that underlies MADSUM's text plan assembly is Rhetorical Structure Theory (RST)[MT83, MT87]. However, the text plan assembly rules in MADSUM do not modify and rearrange the internal structures of the subtrees to the same degree as those in RTPI (see 2.5.2).

The MADSUM system can be viewed as a bottom-up text planning process distributed across multiple agents, where small text plans created by wrapper agents near the bottom of the hierarchy are combined by the agents above them until a single plan is created/selected at the top. ILEX [MOOK98] is a bottom-up, opportunistic RST-based planner implemented in a single program. ILEX and MADSUM are similar in that both superimpose connecting relations on pairs of subtrees; however, because ILEX is a single program it can consider all the possible combinations of all possible subtrees at once. In fact, ILEX employs various heuristics and a genetic algorithm to manage the complexity involved. MADSUM, having the sources of the subtrees distributed, avoids the same level of complexity (but also cannot consider as many possible different combinations). Both ILEX and MADSUM return result trees that may not be optimal.

Like MADSUM, ILEX can plan to a certain text length. Unlike MADSUM, ILEX is not designed to take an arbitrary set of other attributes; nor does ILEX use a utility function to control attribute trade-offs in a decision-theoretic manner designed to satisfy user preferences.

Marcu's bottom-up approach (also a standalone system) was the first to be designed to express all of the content chosen for expression. Similarly, a MADSUM agent presented with a set of text plan trees must design a new tree incorporating all of them. However, Marcu's system is incorporating content propositions, not separate text plan trees.

2.5.2 Generating Multi-Sentence Text

MADSUM is currently implemented with four simple rules for combining pairs of text plan trees. The system is designed to operate with additional rules, if necessary, and different rules can be turned on and off with ease. MADSUM uses a set of domain-independent rules for combining all component trees into one. I will describe how the rules are applied, and then review the rules themselves.

Once a set of text plans is presented to an agent from agents below, the agent is bound to design a text plan tree incorporating all of the received component trees. The set is first ordered, not by the utility or aggregate DS of the components, but by the size of the highest DS portion of the component tree. This facilitates implementation of a rule that when assembled, the component containing the highest DS sub-component will be kept to the left (and therefore be realized as text earlier in the resulting message). Since this rule is applied to all combinations of trees, from the bottom up, the resulting tree and text will have the highest DS text fragment first.

This seems counter-intuitive at first; the component trees were chosen based on the total utility they provide, why not order them the same way? The answer lies in the diverse nature of the utility components. Utility for a very uninteresting component tree t_{long} may only be high because the associated text will be very long and the user placed a high priority (coefficient) on LENGTH. Another component tree, t_{short}, has a very high DS (and so will be significant to the user) but a lower overall utility than t_{long} because the user emphasized Length. I suspected that even when a user wanted a long response, there was nothing about the length of a component that should cause it to appear before another, and so the shorter, more significant component should appear first in the resulting message. This was confirmed to some degree by limited testing (see **??**) and should be further explored.

When should I explain that DS is a pair of triplets? In many ways I feel that this chapter could be clearer if it were presented after the chapter on the architecture.

The first three rules are applied in the order below to each pair of trees. These rules happen to be exclusive in their application, i.e. only one of them will apply to the any given pair of trees, but the system does not rely on this property. As new trees are created they are added to the set, so that the set consists of both small component trees and partially or fully assembled trees.

1. The Elaboration rule (Figure **??**) is designed to aggregate the supporting clauses of two trees that are trying to accomplish the same purpose (this is determined by comparing the structure of the trees over the clauses under consideration). It is only appropriate if the two trees are of the same topic and orientation, and if one supporting clause is substantially more significant (higher DS) than the other.

2. The Joint-Under rule works similarly, but is used when the two supporting clauses are of the same or similar DS. This is also a form of aggregation.

3. When two trees do not agree on an investment (i.e. the DS orientations are different) then the Contrast rule can be applied, even if the trees are not of the same topic (Figure **??**).

4. Finally, when two trees cannot be joined any other way, a Joint relation can be superimposed over them. This is avoided if possible since a Joint contains no useful semantic content to aid realization (or reader understanding).

2.5.3 Generating Natural Language Via Templates

MADSUM generates text in a three stage process:

1. Domain specific agents provide the template-based text for the leaf nodes of each tree fragment they create, and insert the text into a text plan tree template.

2. Text plan trees are propagated up the agent hierarchy, and as trees are assembled/integrated, rule applications can modify the tree structure to affect realization (e.g. the aggregation performed by the Elaboration rule).

3. When the Presentation agent finishes creating a single tree from the trees it receives from below, the agent provides domain independent connectives and punctuation based on the full tree structure.

Templates offer many advantages: rapid development, simplicity, and the ability to include some complex grammatical structures without deep analysis of the text plan trees and domain concepts. [Rei95]. There is also substantial development cost involved in using an existing syntactic realizer [Rei99]. Other adaptive systems employ full syntactic realizers [JBV$^+$, MFLW04], but working with templates allowed me to focus on the issues of content selection, limited aggregation, and ordering. MADSUMS's use of RST-style trees will facilitate a transition to a full syntactic realizer should that be possible in the future.

2.6 Implementation

I have implemented and tested the MADSUM architecture for adaptive response generation in a financial investment domain. MADSUM is implemented in

Java(tm), and has been tested on Sun workstations and Apple G4 desktops and laptops.

The agent architecture (see Chapter **??**) uses a formal agent communication language to transfer information between agents that may be operating on different machines. This involves a high degree of communication overhead. Also, the system has many features designed to make it robust, and these add further communication requirements. Thus the system is rather slow - it takes just under one minute for the two full rounds of communication and simple text tree processing if all thirteen agents are running on a single G4 laptop. Processing a complex problem can take close to two minutes. Those figures can be reduced to ??? when the parallel nature of the architecture is leveraged. For more details on system performance, see Section **??**.

MADSUM has agent "wrappers" whose task is to make text trees out of the data returned by information source agents. The source agents can derive their information from external sources (e.g. websites or other agents), internal stored data from previous external acquisitions, or analysis of data. Currently all source agents access local data files to acquire information, to reduce the vagaries associated with dynamic information gathering (this allows my testing to focus on the dynamics of the data gathered, agent interactions, and user preferences). Work devoted to information gathering includes agents designed for a specific data gathering task [] and also automatically generated agents for simple data gathering [KAH94, Kus00].

2.7 Examples of Adaptive Responses

I plan to include other examples here, but I will wait until I have the sliders implemented to generate them.

Consider a user who proposes the purchase of 100 shares of stock in IBM. The user model contains personal characteristics of the user, including her current investment portfolio and her portfolio allocation goals. In addition, before proposing the stock purchase, the user has set soft constraints on the length of the response, the cost in dollars of any purchased information, and processing time. She has also adjusted sliders on the graphical user interface to indicate the importance she assigns to usage of different resources (length of response, cost, and processing time) and her interest in information that addresses each of the different content categories (investment risk, value, and impact on portfolio allocation).

Figure 2.2 displays MADSUM's response under different soft constraint and priority settings. In Figure 2.2a, the soft constraint on length was 75 words and the user placed a higher priority on risk information than on value and portfolio information. For the responses in Figure 2.2b and Figure 2.2c, the soft constraint on length was lowered to 35 words, resulting in the exclusion of some available propositions. In addition, the relative priorities on risk, value, and portfolio information were kept the same in Figures 2.2a and 2.2b, but were altered in Figure 2.2c to place a much higher priority on value information than on risk or portfolio information. Due to the 35 word soft constraint on length that was set for the response in Figure 2.2b, propositions had to be excluded. Since risk was given highest priority, much (but not all) of the risk information was included. However, the high significance of the proposition about the impact of the proposed investment on the user's portfolio allocation goals (she had already exceeded her goals for equities such as IBM) caused that proposition to increase the estimated overall utility of a response containing this proposition, and thus it was included despite the length of the resulting response slightly exceeding the soft constraint on length. In Figure 2.2c, the user's much higher priority for value information resulted in selection of the value proposition, even though it was of lesser significance than other available propositions. In addition, the highly significant proposition about portfolio allocation goals was included in the response. These examples illustrate the system's ability to vary its responses depending on the user's resource constraints, the significance of information, and

2.2a: *Risk metrics indicate IBM has a low debt-equity ratio, suggesting the ability to weather an economic downturn; further, the company has a strong current ratio, indicating good short-term liquidity. In addition, IBM has historically maintained a moderate debt policy, and the stock has maintained a moderate risk profile. On the other hand, from a portfolio perspective you have already exceeded your allocation goal for equities. Value metrics indicate IBM has a price earnings ratio similar to the tech industry average.*

2.2b: *Risk metrics indicate IBM has a low debt-equity ratio, suggesting the ability to weather an economic downturn; further, the company has a strong current ratio, indicating good short-term liquidity. On the other hand, from a portfolio perspective you have already exceeded your allocation goal for equities.*

2.2c: *Value metrics indicate the stock has a price earnings ratio similar to the tech industry average; on the other hand, from a portfolio perspective you have already exceeded your allocation goal for equities.*

Figure 2.2: Three responses, derived from different soft constraints and priority settings.

the priority that the user assigns to different resources and kinds of information content.

2.8 Evaluation

I have implemented and tested the MADSUM architecture for adaptive response generation in a financial investment domain. The examples in Figure 2.2 are actual responses produced by the system under the conditions described in Section 2.7. Experiments have demonstrated the system's success at varying its responses to adapt to different resource constraints and different priorities for resource and content attributes.

Mellish and Dale, having investigated the subject of evaluating NLG systems, recommend 1) that the components of a system be evaluated separately to distinguish their individual performance, and 2) that evaluations by humans use ranking when possible, instead of open questions, to help unify the continuum of responses. They report that many systems that ask seemingly simple open questions end up with a lack of human agreement that is very hard to statistically overcome [MD98].

In keeping with those strategies, the response generation aspect of my research can be viewed as having two components, content selection and content expression. Of these two, content selection is fundamental to and inherent in the design of MADSUM, insofar as it is the product of the user's utility function and the agent negotiation process. It was critical, therefore, to evaluate MADSUM's content selection choices. I also evaluated MADSUM's ordering of chosen content. While this aspect of MADSUM's response generation is not intrinsic to the system's design, it is nevertheless an important implementation decision that will influence any future extensions of the system.

2.8.1 Selection

To test the system's ability to select appropriate content, I performed an experiment in which 21 subjects were each presented with 7 scenarios (see Appendix **??**. Each scenario consisted of 1) a graphic depiction of sliders representing user priorities for the three kinds of content (Risk, Value, and Goal) 2) two sets of propositions, one of which had been produced by the system. Sometimes the system's response was listed first and other times the alternative appeared first. The subjects were asked to determine, given the user's slider settings for content priority, which content set was most appropriate for that user. In some scenarios, the significance (DS value) of propositions and the priority that the user placed on that kind of information were congruent (each proposition was either both significant and high priority, or both of lesser significance and lesser priority), and in other scenarios the significance and priority of propositions conflicted. The alternatives to the system's responses were constructed to give subjects the opportunity to choose between responses that favored priority in content selection, responses that favored significance, and responses that balanced priority and significance as is done by MADSUM's utility function. I applied a one-tailed binomial test to the results which showed that the subjects had a statistically significant ($p<.01$) preference for MADSUM's strategy of balancing significance and priority in content selection.

2.8.2 Ordering

As discussed in Section 2.5.2, MADSUM places the highest utility propositions early in the response, subject to coherence constraints on the construction of text plan trees. To test the system's ordering of propositions in its presentation to the user, I performed an experiment in which 16 subjects were each presented with 9 scenarios. Each scenario again included a graphic depiction of sliders representing user priorities for the three kinds of content. But in this experiment, the subjects were presented with two different orders of presentation of the same propositions. In each case, most cue phrases and connectives were removed in an attempt to prevent subjects from being influenced by phrasing. Once again, a one-tailed binomial test showed statistically significant ($p<.01$) support for the system's decisions about order of presentation of propositions. This was true even when proposition significance (DS value) and user priority for that kind of information conflicted, although the level of statistical significance for this subset of the scenarios dropped to ($p<.05$).

2.8.3 Evaluation Summary

One must note that full evaluation of a decision support system requires that subjects interact with the system on a real decision that they want to make. Only then is the user in the "frame of mind" such that he or she can make a reliable judgement about the system's performance. Nonetheless, our evaluation experiments support the strategies used by MADSUM and suggest that they will contribute to quality response generation in decision support.

2.9 Summary

MADSUM produces responses that are tailored to a particular user. The user can explicitly influence the system behavior by setting preferences on attributes of the response, including topic priorities and response cost, length, and time to produce. The user can also constrain the system's use of available resources in a hard or soft manner. I have demonstrated that even under a consistent data environment, MADSUM responds differently under different user preferences. Furthermore, my evaluations show that subjects strongly concur with MADSUM's choice of content, and concur to a lesser (but still significant) degree with MADSUM's ordering of content.

Chapter **??** details the contribution of the architecture to the response, as well as the features that allow MADSUM to operate successfully in a dynamic environment.

BIBLIOGRAPHY

[App85] Douglas E. Appelt. Planning english referring expressions. *Artificial Intelligence*, 26(1):1–33, 1985.

[CCC94] Jennifer Chu-Carroll and Sandra Carberry. A plan-based model for response generation in collaborative task-oriented dialogues. In *Proceedings of the 12th Annual American Association for Artificial Intelligence*, pages 799–805, 1994.

[CL97] Charles B. Callaway and James C. Lester. Dynamically improving explanations: A revision-based approach to explanation generation. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, August 1997. IJCAI.

[CP01] David N. Chin and Asanga Porage. Acquiring user preferences for product customization. In *UM '01: Proceedings of the 8th International Conference on User Modeling 2001*, pages 95–104. Springer-Verlag, 2001.

[EB94] W. Edwards and F. Barron. Smarts and smarter: Improved simple methods for multiattribute utility measurement. In *Organizational Behavior and Human Decision Processes*, volume 60, pages 306–325, 1994.

[ECCC94] Stephanie Elzer, Jennifer Chu-Carroll, and Sandra Carberry. Recognizing and utilizing user preferences in collaborative consultation dialogues. In *Proceedings of UM94*, pages 19–24, 1994.

[HDHP97] Graeme Hirst, Chrysanne DiMarco, Eduard Hovy, and Kimberley Parsons. Authoring and generating health-education documents that are tailored to the needs of the individual patient. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 107–118. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[Hov88] Eduard H. Hovy. Planning Coherent Multisentential Text. *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 163–169, 1988.

[Hov91] Eduard Hovy. Approaches to the planning of coherent text. In *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 153–198. Kluwer, 1991.

[JBV$^+$] Michael Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn A. Walker, Steve Whittaker, and Preetam Maloor. Match: An architecture for multimodal dialogue systems, acl 2002 phila pa.

[Jen95] Nicholas R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2):195–240, 1995.

[KAH94] Craig Knoblock, Yigal Arens, and Chun-Nan Hsu. Cooperating agents for information retrieval. In *Proceedings of the Second International Conference on Cooperative Information Systems*, Toronto, Canada, 1994.

[KF88] Robert Kass and Tim Finin. Modeling the user in natural language systems. *Computational Linguistics*, 14(3):5–22, 1988.

[Kus00] Nicholas Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1-2):15–68, 2000.

[LHL97] Greg Linden, Steve Hanks, and Neal Lesh. Interactive assessment of user preference models: The Automated Travel Assistant. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 67–78. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[Mar97a] Daniel Marcu. From local to global coherence: a bottom up approach to text planning. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Providence, Rhode Island, July 1997.

[Mar97b] Daniel Marcu. *The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts.* Ph.D. dissertation, University of Toronto, Toronto, Canada, December 1997.

[MD98] Chris Mellish and Robert Dale. Evaluation in the context of natural language generation. In *Computer Speech and Language*, volume 12, pages 349–373, 1998.

[MFLW04] Johanna Moore, Mary Ellen Foster, Oliver Lemon, and Michael White. Generating tailored, comparative descriptions in spoken dialogue, 2004.

[MOOK98] Chris Mellish, Mick O'Donnell, Jon Oberlander, and Alistair Knott. An architecture for opportunistic text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagra-on-the-Lake, Ontario, Canada, 1998.

[MP90] Johanna Moore and Cecile Paris. Planning text for advisory dialogues. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 203–211, Vancouver, Canada, 1990.

[MP93] Johanna Moore and Cecile Paris. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–695, 1993.

[MT83] William C. Mann and Sandra A. Thompson. Relational Propositions in Discourse. Technical Report ISI/RR-83-115, ISI/USC, November 1983.

[MT87] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, ISI/USC, June 1987.

[MWM85] K. R. McKeown, M. Wish, and K. Matthews. Tailoring explanations for the user. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 794–8, Los Angeles CA, August 1985.

[Par88] C&#233;cile L. Paris. Tailoring object descriptions to a user's level of expertise. *Computational Linguistics*, 14(3):64–78, 1988.

[Rei95] Ehud Reiter. NLG vs. Templates. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, pages 95–105, Leiden, The Netherlands, May 1995.

[Rei99] Ehud Reiter. Shallow vs. deep techniques for handling linguistic constraints and optimisations. In *Proceedings of the KI-99 Workshop on May I Speak Freely: Between Templates and Free Choice in Natural Language Generation*, Bonn, Germany, September 1999.

[Ric79] Elaine Rich. User modeling via stereotypes. In *Cognitive Science*, volume 3, pages 329–354, 1979.

[Rob94] Jacques Robin. *Revision-Based Generation of Natural Language Summaries Providing Historical Background: Corpus-Based Analysis, Design, Implementation and Evaluation.* Ph.D. dissertation, Columbia University, December 1994.

[SV00] Peter Stone and Manuela M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[SWWM02] A. Stent, M. Walker, S. Whittaker, and P. Maloor. User tailored generation for spoken dialogue: An experiment, 2002.

[WH96] Leo Wanner and Eduard Hovy. The healthdoc sentence planner. In *Proceedings of the International Workshop on Natural Language Generation*, pages 1–10, 1996.

[WJK99] Michael Wooldridge, Nicholas R. Jennings, and David Kinny. A methodology for agent-oriented analysis and design. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 69–76, Seattle, WA, USA, 1999. ACM Press.

[ZM93] Ingrid Zukerman and Richard McConachy. Generating concise discourse that addresses a user's inferences. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, August 1993. IJCAI.

[ZM95] Ingrid Zukerman and Richard McConachy. Generating discourse across several user models: Maximizing belief while avoiding boredom and overload. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1251–1257, 1995.