Artificial Intelligence and Law **4:** 1–71, 1996.

BankXX: Supporting Legal Arguments through Heuristic Retrieval*

EDWINA L. RISSLAND, DAVID B. SKALAK and M. TIMUR FRIEDMAN
Department of Computer Science, University of Massachusetts, Amherst, MA 01003, U.S.A.
Email: rissland@cs.umass.edu

Abstract. The BankXX system models the process of perusing and gathering information for argument as a heuristic best-first search for relevant cases, theories, and other domain-specific information. As BankXX searches its heterogeneous and highly interconnected network of domain knowledge, information is incrementally analyzed and amalgamated into a dozen desirable ingredients for argument (called *argument pieces*), such as citations to cases, applications of legal theories, and references to prototypical factual scenarios. At the conclusion of the search, BankXX outputs the set of argument pieces filled with harvested material relevant to the input problem situation.

This research explores the appropriateness of the search paradigm as a framework for harvesting and mining information needed to make legal arguments. In this article, we describe how legal research fits the heuristic search framework and detail how this model is used in BankXX. We describe the BankXX program with emphasis on its representation of legal knowledge and legal argument. We describe the heuristic search mechanism and evaluation functions that drive the program. We give an extended example of the processing of BankXX on the facts of an actual legal case in BankXX's application domain – the good faith question of Chapter 13 personal bankruptcy law. We discuss closely related research on legal knowledge representation and retrieval and the use of search for case retrieval or tasks related to argument creation. Finally we review what we believe are the contributions of this research to the understanding of the diverse disciplines it addresses.

Key words: legal argument, heuristic search, best-first search, evaluation function, bankruptcy law, "good faith", information retrieval, information harvesting, case-domain graph, argument pieces, argument dimensions, argument factors, neighbor methods

Part I: The Approach

1. Introduction

In this article we present our research on the problem of perusing and gathering information for use in legal argument. In particular, we discuss our program BankXX** and its use of the heuristic search paradigm as a computational framework for this information harvesting task.

This research attempts to bring together a number of ideas about artificial intelligence and about law. Its ideas unite information retrieval, architecture and control

* This research was supported in part by grant No. 90-0359 from the Air Force Office of Sponsored Research and NSF grant No. EEC-9209623 State/University/Industry Cooperative Research on Intelligent Information Retrieval.

** We pronounce the name of this program as "Bank-ex-ex".

of AI programs, search, case-based reasoning, legal research, legal knowledge representation and indexing, and legal argument.

Some of the points we will touch upon in our description of the BankXX system are these:

- BankXX is rooted in the task of performing legal research and provides a framework for modeling research strategies.
- The process of gathering information for legal argument can be usefully framed as classic heuristic best-first search.
- The presence of multiple types of legal knowledge and multiple ways to view and index it can be used to advantage in our task.
- Retrieval of cases and other legal knowledge can fruitfully use a combination of knowledge-based indexing and heuristic search.
- Aspects of legal retrieval for argument generation can be modeled by a computer program that relies on a search-driven control strategy.
- *Argument pieces* can be used to represent argument and define an evaluation function.

These ideas are not all new. For example, the body of research on conceptual legal retrieval (e.g. [Hafner 1987a, 1987b; Bing 1987]) has proposed the organization of legal knowledge as a semantic network that implicitly permits multiple indexing. Work in case-based reasoning (CBR) has also made use of multiple indexing [Kolodner, 1983; Turner, 1988]. But our work brings together both these old and a number of new ideas into a single framework.

This work complements and extends our own work on legal argument. For instance, because of its more bottom-up nature, this work on BankXX complements our past work on top-down control of legal reasoning, for example, through argument strategies and tactics, as in CABARET [Rissland & Skalak, 1991; Skalak & Rissland, 1992] and context-sensitive skeletal task plans, as in FRANK [Rissland et al., 1993]. It broadens the scope of what type of knowledge has been explicitly represented in our systems, for example, legal theories and prototypical factual stories. By explicitly representing legal theories (in terms of domain factors), it extends our earlier work on HYPO [Rissland et al., 1984; Ashley, 1990]. It extends the use of "dimensions" to the "meta" realm of argument assessment by using *argument factors* to evaluate and compare arguments. It complements our purely precedent-based representation of argument (e.g., "3-ply arguments") by inclusion of a more diverse set of components – called *argument pieces* – in BankXX's representation of argument (e.g., leading cases, applicable legal theories).*

This research also has a strong evaluation component, and we have striven to understand how our program performs, compared to previous programs, compared to legal opinions, and relative to different parameter settings within the program itself. Consistent with the trend towards more evaluation of AI research, we have

* Of course, there is much more that could be included, such as jurisdictional or procedural aspects, both of which are important [Berman & Hafner, 1991].

performed an extensive series of experiments. These evaluative aspects of our project are reported on in a companion article [Rissland et al., 1995].

This article first introduces the BankXX system generally by discussing in turn each of the bulleted points we have made. It then describes the domain of the BankXX program, an aspect of U.S. federal bankruptcy law concerning personal bankruptcies under Chapter 13. The program itself is described, with emphasis on its representation of its particular area of legal knowledge and its representation of argument. Then we detail the mechanisms of heuristic search, such as the evaluation function that drives the program. Once all these pieces are in place, we give an extended example of the processing of BankXX on the facts of an actual legal case. We conclude this article with a discussion of closely related research on legal knowledge representation and retrieval and the use of search for case retrieval on tasks related to argument creation. Finally we review what we believe are the contributions of this research to the understanding of the diverse disciplines it addresses.

1.1. LEGAL RESEARCH AS SEARCH

The approach to information gathering in BankXX is similar to what a junior associate in a large law firm might do when charged with the task of finding information to support an argument that is being crafted by a senior attorney. Using indices and connections provided by legal materials, the junior lawyer must search through volumes of primary sources (e.g., opinions, statutes) and secondary legal commentary (e.g., treatises, law journal articles) for the legal cases, legal theories, and statutory and regulatory citations to underpin an argument on a designated issue.

Additionally, the associate's search must be completed within a certain time frame and is subject to further constraints, such as the legal materials that are actually available, the intended use of the research (e.g., internal memorandum, formal brief), the amount of time and space that the final legal "product" can use (e.g., a two-page memorandum, a 15-minute oral argument), and even the amount of money that can be spent. That is, there are real limits on the resources that be can be expended.

Obviously, exhaustive blind search is not viable because of the sheer volume of legal materials available. Further, because of the need to make the best use of available resources, the search must be intelligent. To manage his* research activities, the junior associate must use his knowledge of the law and methods of legal research. He must use heuristics that encapsulate useful approaches to his task. Even though heuristics are by nature approximate, they represent usually accurate, ideas of what's important to an argument, both in its details and in its overall quality. Heuristics are often the critical knowledge that makes an expert expert. Thus, the process of gathering information for argument can be seen to fit

* Masculine pronouns should be read to encompass both males and females.

the molds of heuristically-guided search and resource-constrained problem solving so often used in AI [Barr et al., 1981].

BankXX reifies the resource-constrained, heuristically guided best-first search approach in the following way. BankXX gathers information by performing a heuristic search through a network of available legal information. At any point in the search, BankXX has a list of pieces of information that it has discovered and could choose to examine in depth. Based on its heuristic evaluation that captures a sense of what sort of information is needed in a legal argument, BankXX chooses one new piece – the "best" one – to examine in depth and to mine for information. The process of in-depth examination and information extraction opens up new possibilities (e.g., citations to check), which are themselves added to the list of possible items to explore. After this expansion of the list of possibilities, BankXX re-evaluates all the items now on the list since the new information just gathered might have changed the evaluations of some old items. For instance, if BankXX now has an abundant supply of ordinary supporting cases, there is no need to gather more of them, so such cases will have their ratings reduced. Based on the revised list and rankings of possible information to explore, BankXX chooses the most highly valued item – a new "best" item – to examine next. And the process repeats itself. BankXX thus performs an iterative process: visit an item of information, extract information from it, discover new possibilities to explore, add these to the list of possibilities, re-evaluate the revised list, and choose the most important one to act on. The process continues until BankXX's resource limits (e.g., time allowed, limit on number of cases gathered) are exhausted.

In researching a legal issue, it is often the case that a lawyer has a very good general sense of what types of information are needed to mount a convincing argument, even if he does not actually know the specifics of the legal area, such as particular precedents. In addition, a researcher knows how to exploit this general knowledge in the context of his specific problem to find more knowledge. General knowledge used in search includes:

1. what general types of domain knowledge exist (e.g., cases, treatises, annotated statutes, legal theories) and how they are interconnected (e.g., cases cite other cases, cases can announce legal theories, cases invoke legal theories);
2. what basic pieces of information are needed to make an argument (e.g., good supporting cases, cases that trump the opponent's cases, a viable legal theory, an appealing story to tell) and how these support each other (e.g., supporting cases give rise to justifying analogies, a prototypical story can help frame an issue);
3. what makes an argument a good one (e.g., to the extent that one uses central cases an argument is better than one that uses rarely cited outliers; to the extent that supporting cases fit under one theory, an argument is better than one where a variety of theories must be cobbled together).

These general notions about legal knowledge and legal argument help drive the researcher's quest for specific useful information needed to flesh out an argument

about a particular legal problem. At each step of research, such general research knowledge plus emerging problem-specific knowledge provides a scaffold to help frame and mount new knowledge-harvesting forays.

In BankXX such general knowledge about the law and legal argument is used to define certain key computational mechanisms needed to implement the search procedure – the evaluation function, in particular – and represent the underlying knowledge of the domain and components of argument.

1.2. Argument Generation as Search

In our approach, we thus view the generation of argument as resource-constrained heuristic search. At each stage in developing an argument, choices need to be made. Should one seek a broad set of supporting cases, anticipate the best cases for the opposing side, or create a telling hypothetical? Each choice takes the emerging argument to a new state of development. Limited resources force the arguer to make choices about which avenues to pursue. They also cause the researcher to quit when resource limits are reached.

One way to implement argument as search would be to define the search space as the space of all arguments, the start state as the empty argument, and the search operators as ways to advance the argument. Another would be to model argument as the emerging by-product of the search that an expert might perform in a space of domain knowledge.

We have adopted the second view: BankXX performs search in the "domain space" rather than in an "argument space." We adopted this view in part because we are interested in modeling the legal research activities of attorneys and in part because the indexing fabric of the domain space is better understood. In our approach, BankXX carries out its search in a network of frames representing items of traditional legal materials: legal cases, legal theories, etc. As it searches, BankXX identifies nodes that contain domain knowledge that can support an argument. Information from these nodes is analyzed, extracted, and amalgamated in a data structure that represents an argument in terms of various key, desirable ingredients of argument – what we call *argument pieces* – such as citations to cases, applications of legal theories, references to prototypical factual scenarios, etc.

1.3. The Network of Legal Knowledge

Legal experts working in Anglo-American jurisdictions can access a great variety of information in the course of researching a legal issue. Each of these provides an entry point, or index, into a large library of highly interconnected legal materials. Some of these indices are well-tried tools of legal scholarship, such as citation links, legal rules, domain taxonomies and terms of art, and well-known secondary sources. Less traditional indices include legal factors, legal theories and recurring fact patterns. We divide indices into "traditional" and "non-traditional" based on

the degree to which reference materials are typically available to support that type of index to case retrieval.

Traditional Indices

(1) *Citation linkages between cases.* A case cites precedent cases for support of the various legal propositions it advances. Indexing services such as *Shepard's Federal Citations* track the citations among many published cases. Citation signals like *see, but see,* and *cf.* reference precedents in precise ways specified in legal style manuals such as *The Blue Book* [*Bluebook*, 1986]. See [Ashley & Rissland, 1987].

(2) *Rules.* There is no shortage of rules in the law [Twining & Miers, 1982]: statutes; agency regulations; "blackletter" rules (generalizations of case law found in restatements of the law) and the rules of a case (stating the holding of the case). Each type of rule provides a means to access cases: the cases leading to the rule, the cases elaborating the rule, the cases used to justify a predicate of a rule, those following the rule, those representing exceptions to the rule, etc.

(3) *Domain taxonomies and terms of art.* Commercial publishers have also developed indexing schemes, such as the key number system used in WestLaw [West, 1992], in which legal topics are assigned key numbers. Such schemes provide a useful taxonomy of the law and an index to legal opinions. Dictionary, digest, and encyclopedia entries (e.g., *Words and Phrases* [1994], *American Law Reports* [1992]) point legal practitioners to cases that define, interpret, elaborate and refine the meaning of legal terms, which are often the subject of litigation.

(4) *Other secondary sources.* Law review articles and notes, practice manuals, treatises, and other reference works also organize legal knowledge and provide links to related legal sources. The volume and variety of the secondary authorities makes for a research task in itself.

Non-Traditional Indices

(5) *Legal factors or "dimensions."* Many legal areas make use of factors to help frame approaches to resolving legal questions; some come directly from statutes, others arise in the common law. In domains where cases can be compared with respect to a stable set of discernible factors, the factors can be conceptualized and implemented as "dimensions" [Rissland et al., 1984; Ashley, 1990]. Dimensions may be used to index and retrieve cases from a case base and to order precedents by their relevance to a problem situation as in the HYPO and CABARET systems [Ashley, 1990; Rissland & Skalak, 1991].

(6) *Legal theories.* Courts and advocates usually strive to provide what is often called a *legal theory* as to why a case should be decided a certain way. In many domains, such as ours, a legal theory is frequently couched in terms of a collection of factors or features explicitly to be considered when adjudicating an issue. Thus,

although there is no universally accepted definition of a legal theory, there is ample motivation to model a legal theory as a collection of legal factors or dimensions. That is the approach we have taken in BankXX. Knowing what cases have been argued under a theory is a means to access other cases, such as cases in which a theory was clearly held to control a decision. In addition, relations between legal theories themselves, such as the refinement of one theory by another, permit "nearby" theories to be retrieved, along with the cases that apply them.

(7) *Recurring prototypical fact patterns or legal stories.* Generic cases or recurring fact patterns – what could be called *legal stories* or *scripts* – have been used in legal reasoning [Gardner, 1987] as well as in other domains [Schank, 1990]. If, for example, a problem involves a former student with large educational debts who files for bankruptcy within a few years of graduating, a bankruptcy expert may be reminded of other student loan cases. In particular, knowing that a legal theory has successfully been applied to cases fitting a particular fact pattern provides a basis for creating analogies to justify applying that theory. Sullivan et al.'s very thorough analysis of bankruptcy law [Sullivan et al., 1989] uses such *story prototypes* to organize the data and conclusions.* In fact, titles for six chapters in the book refer to story prototypes: entrepreneurs, homeowners, women, medical debtors, credit card junkies and repeat bankruptcy filers.

Each type of legal material has its own emphasis or imparts its own perspective on legal knowledge. Each displays its own strengths and weaknesses as an indexing medium. While our project deals with primary legal authorities, one should not overlook the role of the other types of knowledge in legal research. For instance, secondary authorities like the *American Law Reports* (ALR) or *Corpus Juris Secundum* are very useful places to find cases to start one's search. Kunz and colleagues describe an interesting experiment in which four of the authors of this book on legal research strategies performed a research task and maintained a protocol of the materials consulted [Kunz et al., 1992]. Each started with secondary authorities. It was noted that secondary authorities provided "insight into pertinent legal theories" [p. 468] as well as citations to primary authority.

One should also not overlook the importance of rules. In previous work, we examined the role of rules as indices to legal cases in [Rissland & Skalak, 1991; [Skalak & Rissland, 1992]. In our CABARET system, relations between cases and rules were used to define strategies and tactics that ultimately specified what type of case to index in a given argumentative situation. Branting in his work on GREBE also explored in depth the relation between rules, rule predicates, and cases [Branting, 1991].

The role of prototypical fact patterns in organizing case knowledge should not be overlooked either. Gardner made strong use of fact patterns in her groundbreaking research [Gardner, 1987]. Legal arguments are often based on similarity

* For instance, they found that cases involving "credit card junkies" accounted for less than 2% of the bankruptcies they studied. Medical calamity debtors accounted for 1%–2% [Sullivan, et al., 1989, p. 168, p. 188].

of the underlying fact patterns. Fact patterns provide clues as to how the cases can be analogized and distinguished. In particular, where the temporal sequence of facts of two cases is similar, a legal argument can be made that the cases are alike and should be treated similarly under the law. In many domains such as contract or property law, a sequence of facts arises time and again (e.g., an offer is made and accepted, a piece of property is leased and then leased by the lessee to a sublessee). These recurring fact patterns provide the opportunity to make legal arguments based on the similarity of a case to a general factual pattern. Note, legal stories can be very useful indices even if there is not complete agreement about what the prototypical stories are, just as a West keyword does not require that everyone agree on its meaning to be a useful index.

Because legal materials can be viewed from various perspectives and can be linked in various ways, legal knowledge is naturally represented as a highly interconnected network whose nodes represent various items of knowledge and whose links represent their interconnections. Said another way, the representation of legal knowledge is a graph of heterogeneous nodes rather than a tree since in a graph, there can be multiple routes to an individual node whereas in a tree, there can be only one.

1.4. The Complementary Nature of Indexing and Search

The network representation of legal knowledge provides flexible ways of indexing. This flexibility aids retrieval of information, like cases, in several ways. Multiple paths to cases, found through the sequential application of different types of indices, can be coupled with case representations at different levels of abstraction to yield a finer retrieval granularity. The use of multiple types of indices also increases the robustness of case retrieval in "real world" domains in which noisy cases can be indexed incorrectly. Mis-indexing a case by one index does not render it inaccessible since other indices still provide a path. Finally, from a cognitive vantage point, in a richly connected domain like the law, people use a variety of indices for reminding or for accessing information [Schank, 1982; Rissland, 1978].

However, the existence of this rich indexing fabric means there are many choices of how and what information to index. In search terms, there are many choices of how to wend one's way through this highly branched network. In the face of limited resources, this means that there is a premium placed on effective exploration of the network. In other words, some intelligence is needed to search and harvest information from this rich domain network; an exhaustive search is not practical.*

In BankXX, knowledge about argument and legal materials – captured, for instance, in the evaluation function – is used to constrain and guide the search.

Indexing and search present two extremes for retrieval. At one extreme, a set of indices may function as database retrieval keys, and no search of the case memory

* This is especially so if one considers a realistic law library situation, where materials constitute a network with gigabytes upon gigabytes of information.

need be done, only whatever minimal search is required to match the database key; cases are pre-indexed to permit immediate retrievals. At the other extreme, search is relied on entirely. Through an evaluation function, spreading activation, planning, blind rummaging, or some other technique, the case space is searched for the desired cases. Search may be needed even in a supposedly well-indexed case base if static indices cannot function as database keys; perhaps the domain is rapidly changing or cases need to be retrieved in ways not anticipated or enabled by the original indexing. We see both indexing and search as useful, and the question is how to combine them.

Another problem in retrieval is that an indexing fabric may not be adequate if the constraints stemming from the task cannot be readily translated into it. Typically, indices are encoded at the domain level whereas tasks are described in a different vocabulary altogether. This is certainly true in most CBR systems where typical tasks are planning, design, argumentation, or teaching, and cases and indices are encoded at the domain level [Ashley & Aleven, 1991; Hammond, 1989; Sycara & Navinchandra, 1991].

Such a mismatch between the encoding of task and case indices exists in generating arguments. The vocabulary of the constraints on an emerging argument is different from the indexing vocabulary available for case retrieval. To take an extreme (but real) example, suppose that the cases are full-text legal opinions and Boolean combinations of keywords are the only indices available; further suppose the requirement of the argument is to supply a case that uses the opponent's best theory, so that one can distinguish the case from the current problem and thereby discredit application of the opponent's theory. The constraints of this task cannot be readily expressed in terms of the available indices: there is a mismatch between the indexing and the task vocabularies.* That may be true even in more sophisticated indexing regimes, such as those based on "dimensions" used in our own work, although the problem is not as extreme, since dimensions are designed to capture factors that are important to the task of arguing. In situations with a indexing gap, information needed for the argumentation task cannot be found by indexing alone. Barring revision of the indexing vocabulary or a re-conceptualization of the domain, some search of the information resources is probably needed.

BankXX bridges the gap between what's available from the indexing schemes and what's needed for the task of argument by using best-first search guided by evaluation functions defined at various levels of abstraction. At the lowest level – the *domain* level – the evaluation function uses only information readily available from indexing at the domain level. At the highest level – the *argument task* level – the evaluation uses information addressing the overall substance and quality of the argument. At the intermediate level – the *argument piece* level – the evaluation function uses information computable from the domain level but geared to the needs of the argument task level.

* An analogous vocabulary gap between instances and their generalizations has been noted by [Porter, Bareiss & Holte, 1990].

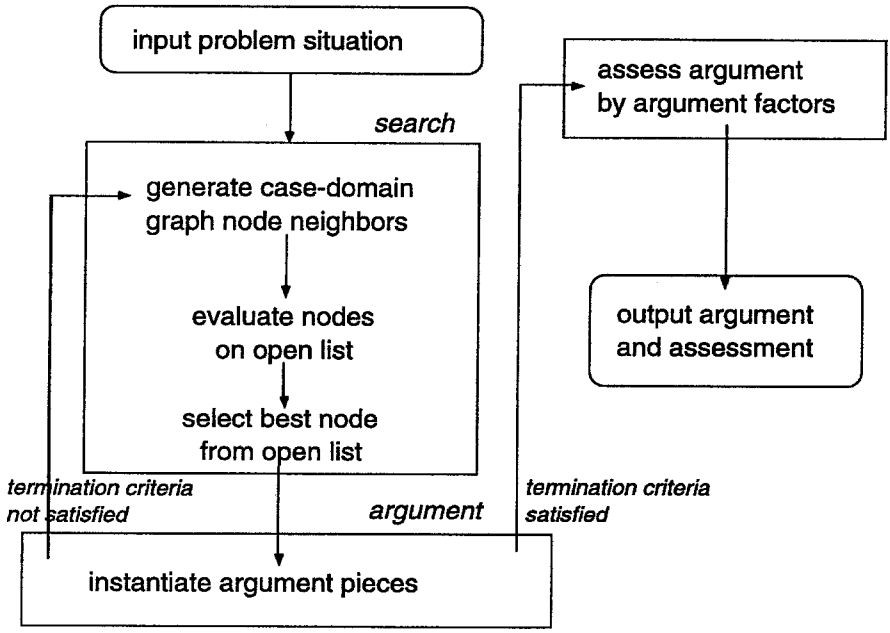

Figure 1. Control flow of the BankXX system.

In summary, BankXX incorporates a hybrid search-indexing approach that couples indexing with search of cases and other domain knowledge through best-first search in order both to address shortcomings in available indexing structures and to increase the leverage obtainable from the existing indices.

1.5. Overview of BankXX's Architecture: Search-Driven Control

BankXX uses a search-driven control architecture to search a network of domain knowledge in order to harvest information that can be used to support a legal argument. The search is heuristic and constrained by resource limits. BankXX's overall control flow is shown in Figure 1. BankXX runs on the Macintosh family of computers, and is written in Macintosh Common Lisp v.2.0 using Common Lisp Object System (CLOS).

BankXX searches through its network of domain knowledge using a variation on the classic method of heuristic best-first search [Barr et al., 1981]:

> BankXX "expands" the "current node" (the material currently being examined) to generate "successor" nodes (new materials to look at); these are placed on a list of "open" nodes (a list of possible materials to examine). A heuristic evaluation function is applied to them* and the "best" of all of these becomes

* *N.B.* In the BankXX implementation, each node on the OPEN list is (re-)evaluated in each cycle. This is a departure from the traditional algorithm. Details are discussed below in Section 4.1.

the new current node, which is then mined for the information it can yield for the evolving argument. The process iterates until specified resource limits are reached.

BankXX finds successor nodes by calculating and chasing interconnections through the network used to represent legal knowledge. For instance, BankXX can chase citation linkages between cases. The network, called the *case-domain graph*, is described in detail in Section 3.1. The methods, called *neighbor methods*, used to generate successor nodes – that is, the "expansion" methods in search parlance – are described in detail in Section 4.2.

BankXX uses a heuristic evaluation function to assess a node's potential for contributing useful information to the emerging argument. We have experimented with three different evaluation functions. They capture knowledge at one of three levels of abstraction: (1) knowledge about general types of information in the domain, (2) knowledge about types of information needed for an argument, and (3) knowledge about what makes an argument good. Evaluation functions are described in Section 4.3.

BankXX places two kinds of limits on resources to constrain its processing. The first are overall limits, such as processing time used (measured in so-called *billable seconds**) and number of nodes opened. The second are the so-called *fill limits* used when BankXX is run with the argument-piece evaluation function.

When BankXX is run with the argument-piece evaluation function, a fill limit is a limit placed on the number of items that can be harvested for a particular component of the argument data structure. The components are called *argument pieces*; each argument piece has a fill limit. Fill limits model the notion that when enough of a particular kind of information has been harvested, there is little value in harvesting more of it. When a fill limit on an argument piece is reached, BankXX considers the argument piece to be full and will not harvest any more items for it. Additionally, the argument-piece evaluation function thereafter gives less value to such items. Fill limits are described in Section 4.3.2.

BankXX can be viewed as using a bottom-up approach since its overall processing is data-driven. Its information gathering activities are driven by the research materials that are encountered. However, we should note that the task (i.e., filling in argument pieces) and important details of the search model (i.e., evaluation functions) import top-down concerns into this bottom-up approach. The perspective we take in BankXX complements previous work in which we sought to recognize and apply structures of legal argument imposed from the top down [Skalak & Rissland, 1992]. In a complete picture, we believe that argument generation includes a flexible control strategy, combining top-down, bottom-up and island-driving strategies. Legal research seems clearly an opportunistic process [Kunz, et al. 1992].

* *Billable seconds* is the unit of time used in BankXX. Processing time starts from the moment BankXX is invoked until it returns its output and includes any time spent on garbage collection and output to the screen during intermediate processing.

2. The Bankruptcy "Good Faith" Domain

BankXX is instantiated in the area of bankruptcy law for individuals that is covered by Chapter 13 of United States bankruptcy law (11 U.S.C. §§ 1301–1330). Chapter 13 provides a means for individual debtors to obtain relief from debts while keeping much of their property. Under Chapter 13 a debtor pays his creditors according to a court-approved plan that allocates 100% of his disposable income for a period of three to five years. Successful completion of the plan discharges the entire debt, regardless of the portion that is actually repaid. By contrast, Chapter 7 (11 U.S.C. §§ 701–766) is based on liquidation of a debtor's assets to satisfy debts.

There is potential for abuse of the debt-absolving power of Chapter 13. For example, a consumer could take out a large loan and spend the money with no intention of repaying it; a student could take out an educational loan and default on it without even trying to repay the loan. By declaring bankruptcy such a debtor would hope to get away with just repaying a small fraction of what is owed. One way the law is designed to prevent this and other abuse is by requiring that a repayment plan be "proposed in good faith" as required in § 1325(a)(3):

> § **1325. Confirmation of plan**
>
> (a) Except as provided in subsection (b), the court shall confirm a plan if –
>
> (1) the plan complies with the provisions of this chapter and with the other applicable provisions of this title;
>
> (2) any fee, charge, or amount required under chapter 123 of title 28, or by the plan, to be paid before confirmation, has been paid;
>
> *(3) the plan has been proposed in good faith and not by any means forbidden by law;*
>
> (4) the value, as of the effective date of the plan, of property to be distributed under the plan on account of each allowed unsecured claim is not less than the amount that would be paid on such claims if the estate of the debtor were liquidated under chapter 7 of this title on such date;
>
> (5) with respect to each allowed secured claim provided for by the plan
>
> . . .
>
> (6) the debtor will be able to make all payments under the plan and to comply with the plan.
>
> 11 U.S.C. § 1325(a), emphasis added

Since Chapter 13 took effect as part of the Bankruptcy Reform Act in October, 1979 many cases have been litigated around the good faith issue. Evolving case law has elaborated what constitutes "good faith," a term left undefined in the original text of the law. Courts of Appeal for most of the federal circuits have articulated legal theories on the issue; to date the Supreme Court has not. The general approach taken by most courts has been to list a number of "factors" that a bankruptcy court should consider in making its decision. For example, one influential standard was articulated by the Eighth Circuit Court of Appeals in 1982 in *In re Estus*:

"[I]n addition to the percentage of repayment to unsecured creditors, some of the factors that a court may find meaningful in making its determination of good faith are:

(1) the amount of the proposed payments and the amount of the debtor's surplus;

(2) the debtor's employment history, ability to earn and likelihood of future increases in income;

(3) the probable or expected duration of the plan;

(4) the accuracy of the plan's statements of the debts, expenses and percentage repayment of unsecured debt and whether any inaccuracies are an attempt to mislead the court;

(5) the extent of preferential treatment between classes of creditors;

(6) the extent to which secured claims are modified;

(7) the type of debt sought to be discharged and whether any such debt is nondischargeable in Chapter 7;

(8) the existence of special circumstances such as inordinate medical expenses;

(9) the frequency with which the debtor has sought relief under the Bankruptcy Reform Act;

(10) the motivation and sincerity of the debtor in seeking Chapter 13 relief; and

(11) the burden which the plan's administration would place upon the trustee."

In re Estus, 695 F.2d 311, 317 (8th Cir. 1982) at 317

The *Estus* court leaves open the questions of whether these are all the factors that a bankruptcy court should consider and how are they to be applied. In fact, the court prefaces its listing of the factors by saying, "We make no attempt to enumerate all relevant considerations since the factors and the weight they are to be given will vary with the facts and circumstances of the case." (*In re Estus*, 695 F.2d 311, at 317). However, *Estus* does give special emphasis to one factor: the percentage of debt repaid to unsecured creditors, stating that "[a] low percentage proposal should cause the courts to look askance at the plan" (*ibid.*).

There are many such factors in the corpus of cases addressing the "good faith" issue. More examples are a *per se* minimum payment requirement, which requires that a threshold percentage of debts be repaid under a plan (see, *In re Burrell*, 2 B.R. 650 (1980), *reversed*, 25 B.R. 717 (1982)), and a blanket test of good faith, which requires that "all the facts and circumstances" be considered (see, *Barnes v. Whelan*, 689 F.2d 1983 (D.C. Cir 1982)). We will call a collection of such factors a *legal theory*.

Often legal theories can be viewed as related or derived from each other. For instance, the *Easley* case expands the set of *Estus* theory factors by listing additional factors borrowed from other cases (*In re Easley*, 72 B.R. 948 (Bkrptcy M.D. Tenn. 1987)). Thus the 16-factor *Easley* theory is related to the *Estus* theory because it

includes the *Estus* factors as a subset. Sometimes one theory modifies individual factors from another theory, or alters another theory's set of factors by grouping them or suggesting different relative weightings. For instance, *Iacovoni* uses a factor-by-factor approach, citing a number of previous applied factors, but adds the superstructure that a particular group of factors are to be used in determining whether a good faith effort to make a "meaningful repayment" has been made (*In re Iacovoni*, 2 B.R. 256 (1980)).

Links between legal theories show the derivation and change in theories due to their treatment by the courts. Legal theories can also change because of changes in the bankruptcy code itself. For instance, The Bankruptcy Amendments and Federal Judgeship Act of 1984 added to section 1325(b) a requirement that 100% of a debtor's disposable income be used in the plan, eliminating the relevance of *Estus* factor (1) in subsequent cases.*

In summary, in BankXX, we explicitly model some aspects of legal theories. In particular, we model legal theories as collections of factors and we use a network to represent relationships between theories.

Part II: The Implementation

3. Knowledge Representation in BankXX

In this section we begin our in-depth description of BankXX. Here we describe the static aspects of the system, particularly the representation of knowledge about argument, cases, and other aspects of the bankruptcy domain. In Section 4, we describe the dynamic workings of BankXX's search process, including the heuristic evaluation functions that drive the search of the static data structures.

3.1. Representation of Domain Knowledge

3.1.1. *The Case-Domain Graph*

The knowledge base in BankXX consists of a semantic network whose nodes represent cases and legal theories, and whose labeled links represent their interconnections. We call this network the *case-domain graph.* It consists of *case-domain-graph nodes* together with labeled link edges. There are six types of case-domain-graph nodes. One type represents legal theories and five represent legal cases from various perspectives proven useful to human legal reasoners. The five ways legal cases are represented are:

* The current statute requires in 11 U.S.C. 1325(b)(1) "If the trustee or the holder of an allowed unsecured claim objects to the confirmation of the plan, then the court may not approve the plan unless, as of the effective date of the plan – (A) the value of the property to be distributed under the plan on account of such claim is not less than the amount of such claim; or (B) the plan provides that all of the debtor's projected disposable income to be received in the three-year period beginning on the date that the first payment is due under the plan will be applied to make payments under the plan." Subsection (b) was added by the Bankruptcy Amendments and Federal Judgeship Act of 1984.

(1) as factual situations,
(2) in terms of domain-specific factors,
(3) as bundles of citations,
(4) as stereotypical stories or scripts, and
(5) by the measure of their prototypicality.

Although the case-domain graph is one network, cases of like type in it can be grouped into *spaces*: *Case Citation Space*, *Legal Theory Space*, etc. Each space captures a particular type of knowledge and its natural interconnections. They impart a structure similar to the partition of a blackboard system's working memory into spaces. Each space represents legal information according to a particular perspective.

The case-domain graph is highly interconnected: *in-space* links connect objects within a space and *cross-space* links connect objects in different spaces. During search of the case-domain graph, links are traversed by BankXX's neighbor methods, operators that expand nodes by following in-space and cross-space links. Traversing a link is tantamount to using the link label as an index.

The current implementation contains 54 cases from the bankruptcy good faith domain. These directly spawn 108 case-domain-graph nodes since each case gives rise to two nodes: one represents the case as a factual situation and one as a set of domain factors. BankXX uses 26 domain factors. The case-domain graph also contains 18 nodes representing legal theories and 70 representing intercase citations. BankXX uses 9 prototypical story scripts. Altogether there are 251 nodes.

3.1.2. *The Spaces within the Case-Domain Graph*

We now describe each space within the case-domain graph, including its in-space links and some of its cross-space links.

Fact Situation Space. Fact situation space case-domain-graph nodes – or simply *case nodes* – encode legal cases as sets of facts. A case node, which is the representation in which a case is input, is the surface level of factual description and in many ways is the "generic" representation of a case. Each case node is represented as a tree of frames implemented as Common Lisp Object System instances. Examples of frames at this level describe the proposed plan and payments, the debt, the debtor's income, and other generic information about the case. Conceptually, cases in this space are linked to each other through case citations.* Cases plus their intercase connections make up *Fact Situation Space.* An example of a top-level frame for a case (the *Estus* case) as a fact situation node is given in Figure 2.

Domain Factor Space. Legal cases can be represented in terms of their values on domain-dependent factors or "dimensions" [Rissland et al., 1984; Ashley, 1990].

* Case citation linkages are actually implemented indirectly through case-citation nodes. Regardless, the net effect of citations is to link cases.

```
(make-instance 'STUDENT-LOAN-CASE
:name 'ESTUS
:case-link 'ESTUS
:citation "695 F.2d 311 (8th Cir. 1982)"
:year 1982
:level :court-of-appeals
:judge 'henley
:summary "Holder of VA student loan claim appeals confirmation of 15-month plan that
gives zero payment to unsecured creditors. Henley, J. held that good-faith requirement
does not require <<substantial payment>> to unsecured creditors and lists meaningful
factors. Reversed and remanded."
:procedural-status 'appeal-of-plan-confirmation
:decision-for 'creditor    ;appellant, reversed and remanded
:citations nil
:factual-prototype 'student-loan    ;$2900 of $11000 unsecured debt is student loan
:alternative-factual-prototype nil
:legal-prototype nil
:legal-theory '(ESTUS-THEORY)
:chapter 13
:plan-confirmed :no
:estus-factors 'ESTUS-ESTUS-FACTORS
:debt 'ESTUS-DEBT
:plan-payments 'ESTUS-PLAN-PAYMENT
:past-filings nil
:ch-7-filing-date nil
:plan-filing-date "07–09–80"
:unfair-manipulation nil
:attempts-to-pay nil
:profession 'federal-employee
:dropout nil
:change-in-field nil
:loan-due-date nil
:makarchuk-factors 'ESTUS-MAKARCHUK-FACTORS)
```

Figure 2. Example of the top-level case frame for the Estus case.

Factors are derived features recognized by domain experts as strongly influencing a case's outcome. A factor allows cases to be compared and assessed as stronger or weaker with respect to the factor's perspective. In *Domain Factor Space*, a case is represented by a vector composed of the magnitudes of the case on each dimension that applies to it; non-applicable factors are encoded as NIL. This vector of domain factor values represents a case as a point in an n-dimensional space. While there are no explicit links between nodes in the Domain Factor Space, the vector space structure of this space does allow us to define relations between them, for instance, according to how "close" they are. The specific sense of closeness depends on the

The *Estus* factors are:	The *Makarchuk* factors are:
percent-surplus-of-income-factor	relative-timing-factor
employment-history-factor	relative-total-payment-amount-factor
earnings-potential-factor	relative-monthly-payment-amount-factor
plan-duration-factor	use-of-skills-gained-factor
plan-accuracy-factor	relative-educational-loan-debt-factor
preferential-creditor-treatment-factor	de-minimis-payments-factor
secured-claims-modified-factor	
debt-type-factor	Other factors are:
nondischarge-7-factor	attempts-to-pay-factor
special-circumstances-factor	repayment-unsecured-debt-factor
frequency-relief-sought-factor	necessary-expenses-minus-plan-payments-factor
motivation-sincerity-factor	unfair-manipulation-factor
trustee-burden-factor	inaccuracies-to-mislead-factor
	other-relevant-considerations-factor
	likelihood-income-increase-factor

Figure 3. The factors employed by BankXX.

choice of metric or distance function imposed on the space (e.g., standard Euclidean distance, city block distance). With respect to any given BankXX dimension, it is easy to derive relations between cases on that dimension, as was done in HYPO.

BankXX uses 26 factors in its domain. They are divided into three groups. The first comes directly from the *Estus* case. The second comes from the *Makarchuk* case. The third comes from a variety of sources. See Figure 3.

Using a HYPO-style analysis [Ashley, 1990], BankXX creates cross-space links between factors and the cases to which they apply (Figure 4). Factor analysis of a problem case is one of the first steps in processing a new case. From the usual HYPO-style factor analysis, BankXX creates a regular claim lattice, from which it computes most on-point and best cases.

Legal Citation Space. Citation nodes encode the citations found in cases. They link cases together via their cites. Each citation case-domain-graph node represents one citing/cited pair: *Case-x citation-signal Case-y*. The *citation signal* specifies the sense in which a case is cited [*Shepard's*, 1994] [*Bluebook*, 1986]. Citation nodes are not linked to each other. Conceptually they provide links between the citing case and the cited case in Fact Situation Space (Figure 5).

Citation space currently uses eight links. All but two (*agrees-with* and *discusses*) have definitions that correspond to their definitions as citation signals in *The Blue Book* [1986]. Our *cites* corresponds to *The Blue Book's* "[no signal]."

(1) *cites* – "Cited authority (i) states the proposition, (ii) identifies the source of a quotation, or (iii) identifies an authority referred to in text."

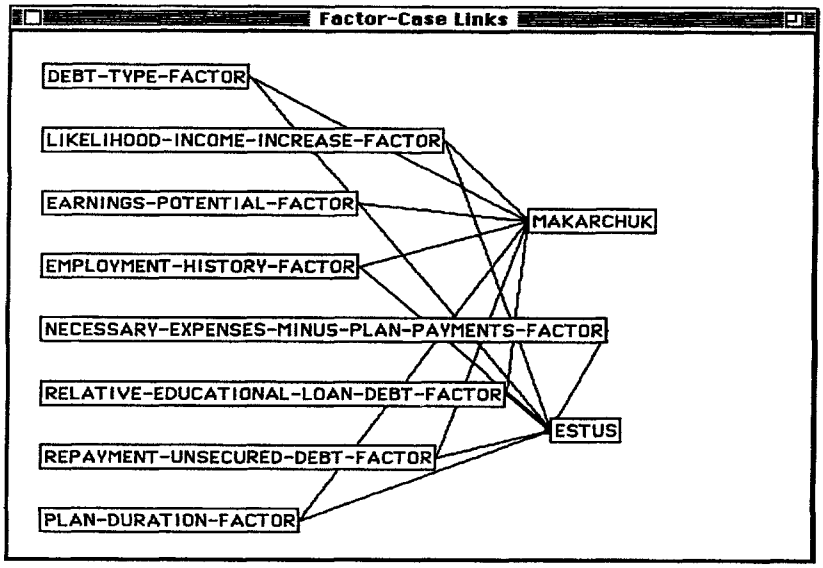

Figure 4. A small subset of the indexing links between domain factors and cases.

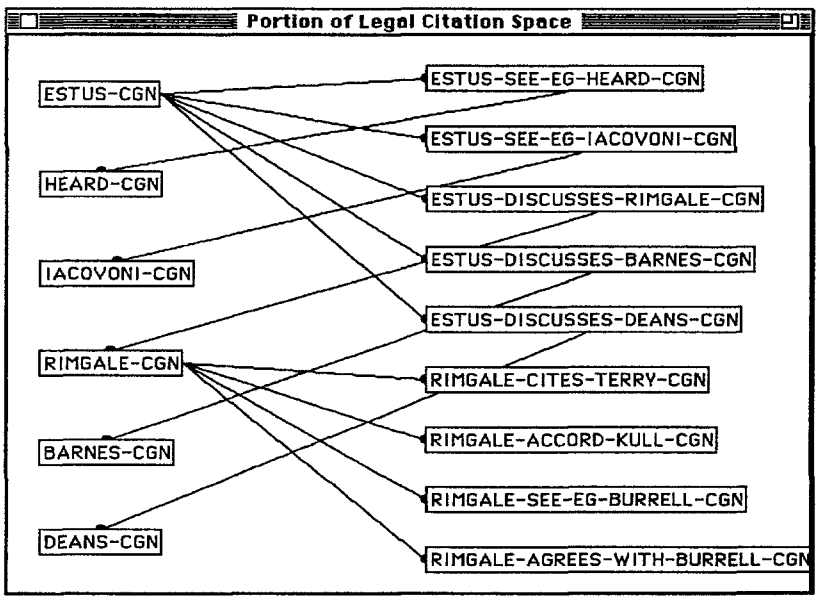

Figure 5. A small subgraph of the case-domain graph showing cases in Fact Situation Space (left side of figure) linked to citation nodes in Legal Citation Space (right side). "CGN" stands for case-domain-graph node.

(2) *eg* – "Cited authority states the proposition; other authorities also state the proposition, but citation to them would not be helpful."
(3) *accord* – Cited authority "directly supports the proposition" but in a slightly different way than the authority(ies) first cited.
(4) *see* – "Cited authority directly supports the proposition. *See* is used instead of "[no signal]" when the proposition is not stated by the cited authority but follows it."
(5) *see-also* – "Cited authority constitutes additional source material that supports the proposition."
(6) *see-eg* – Cited authority provides illustration of the proposition.
(7) *agrees-with* – The court opinion explicitly used the phrase *agrees with* (as in "We agree with the analysis in the *Estus* case . . . ")
(8) *discusses* – same as the intuitive notion of discusses: the court analyzed aspects of the precedent, usually devoting several paragraphs to a case's facts and/or previous analysis.

Legal Story Space. In bankruptcy, as well as in other domains, cases often follow certain standard scripts or prototypical story lines. We call such stories *factual prototype stories.* In BankXX, we use many of those given in Sullivan's book on bankruptcy [Sullivan et al., 1989] as well as several that we identified. (See Appendix D for a list of story lines identified in our bankruptcy domain.) Of the 9 stories actually used in BankXX, the following are the most frequently used:
(1) *the student loan story* – student incurs educational debts and soon after graduating files for bankruptcy protection from his educational loan creditors.
(2) *the dishonest debtor* – debtor commits fraud or some other offense, a judgment is entered against debtor, debtor files for bankruptcy.
(3) *the automobile debtor* – debtor borrows money to purchase automobile beyond his means, and declares bankruptcy to avoid repaying the portion of the loan above the value of the car.
(4) *the consumer debtor* – debtor purchases goods and services on credit, usually with credit cards, and files for bankruptcy to discharge the liabilities.

BankXX does not link story-prototype nodes to each other. Exploiting such links would require an understanding of how stories can be related and, ideally, an automated means to recognize them (e.g., plot units [Lehnert, 1981]).

Legal Theory Space. Legal theories are defined by a list of factors (see the discussion of Domain Factor Space, above) that are prerequisite conditions for a theory to apply to a case. A theory node lists the case that promulgated it as well as cases that applied it.* Figure 6 shows the case-domain-graph node representing

* Our criteria for listing a case as applying a theory were quite strict. For instance, we checked the opinion to see if the court considered the factors defining it.

```
(make-instance 'legal-theory
  :name 'ESTUS-THEORY
  :other-names '(FLYGARE-THEORY)
  :description "<omitted>"
  :factors '(percent-surplus-of-income-factor
    employment-history-factor
    earnings-potential-factor
    plan-duration-factor
    plan-accuracy-factor
    preferential-creditor-treatment-factor
    secured-claims-modified-factor
    debt-type-factor
    nondischarge-7-factor
    special-circumstances-factor
    frequency-relief-sought-factor
    motivation-sincerity-factor
    trustee-burden-factor)
  :factor-evaluation nil
  :domain-theories 'debt
  :view :majority
  :cases-promulgating '(estus)
  :cases-applying '(estus flygare ...)
  :cases-rejecting nil
  :courts-adopting :8th-circuit    )
```

Figure 6. Representation of the Estus theory.

the *Estus* theory; the *Estus* case promulgated the ESTUS-THEORY and *Estus* and *Flygare*, among others, applied it.

Legal theory nodes are linked by pointers that describe the relationships between them, such as "overlaps with," "rejects," and "agrees with." Figure 7 shows some of the legal theories in the constellation of theories connected to the *Estus* case. Legal theories were culled from opinions by hand. In order for us to record that a theory applies to a legal case, the opinion must explicitly state that it applies to the case.

Currently, BankXX uses nine different linkages between theories:

(1) *overlaps-with*,
(2) *conflicts-with*,
(3) *rejects*,
(4) *derives*,
(5) *is-derived-from*,
(6) *agrees-with*,
(7) *refines*,
(8) *is-refined-by*,
(9) *is-equivalent-to*.

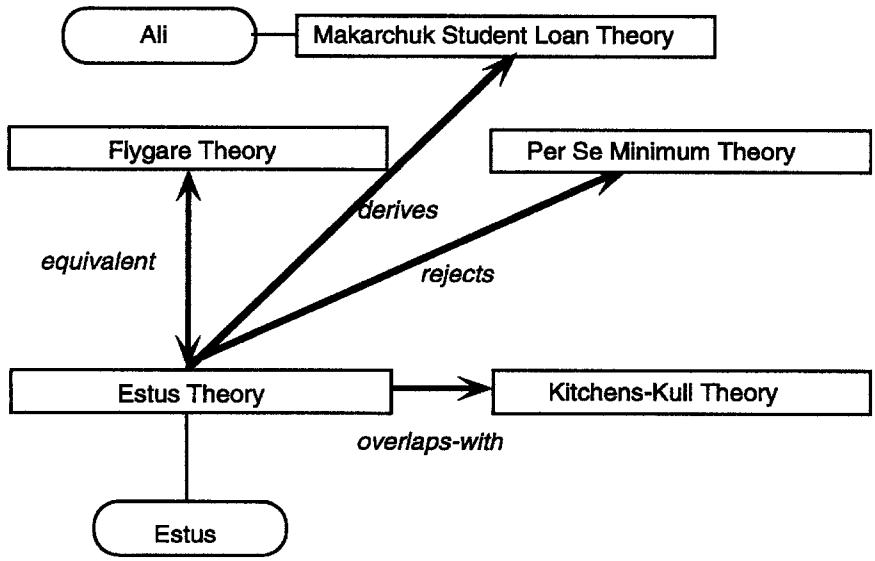

Figure 7. A small subgraph of the case-domain graph, showing inter-theory links and links from theories to cases.

	case	citation	theory	factual-prototype	domain-factor
case	x	x	x	x	x
citation	x				
theory	x		x		x
factual-prototype	x				
domain-factor	x		x		

Figure 8. Matrix of links between case-domain-graph node classes that are implemented in BankXX. Case-to-case links are implemented via citation links.

In some preliminary research that is not incorporated into BankXX, we have begun to build a taxonomy of methods that manipulate legal theories. Theories may be manipulated by adding factors, limiting the factors considered, changing the way the factors are combined (as by a weighting scheme), or simply shifting the burden of persuasion of proving the theory. There is much research that can be directed at theory formation in the law.

In addition to the spaces with their in-space links just described, a variety of bi-directional, cross-space links exist. For instance, links exist between factors and legal theories that use those factors, and between story prototypes and cases instantiating them. See Figure 8.

3.2. Argument Knowledge Representation

3.2.1. *Argument Pieces: Some Building Blocks for Argument*

We have chosen a simple representation of an "argument" for purposes of this project. An argument is simply a collection of *argument pieces*, that represent fragments of arguments or pieces of legal knowledge that an advocate would ideally like to have to support his position. Argument pieces represent building blocks of argument. BankXX's task is to gather information necessary to fill in these building blocks. We recognize that this idealization of argument does not reflect the logical and rhetorical connections between the various pieces of an argument, or the complexity of argument in general.

The 12 argument pieces currently used in BankXX are:

1. **Supporting Cases** – cases decided for the same side as the viewpoint assumed in the current problem situation. Synonymously called *pro* or *same-side* cases.
2. **Best Supporting Cases** – the best cases decided for the current viewpoint.
3. **Contrary Cases** – cases decided for the opposing side.
4. **Best Contrary Cases** – defined similarly to 2.
5. **Leading Cases** – the five most frequently cited cases in the BankXX corpus.
6. **Supporting Citations** – citations found in same-side cases that lead to other same-side cases.
7. **Factor Analysis** – the set of domain factors ("dimensions") that are applicable to the current problem situation.
8. **Overlapping Cases** – cases sharing a large proportion (75%) of domain factors.
9. **Applicable Legal Theories** – if each factor defining a theory is applicable to the problem situation, the theory is considered applicable.
10. **Nearly Applicable Legal Theories** – a theory is nearly applicable if a threshold percentage (50%) of defining factors apply.
11. **Factual Prototype Story category** – the story lines that the debtor's factual situation falls under.
12. **Family Resemblance Prototype** – same-side cases having the highest family resemblance rating, with respect to a given family, to the instant case according to the Rosch measure of family resemblance [Rosch & Mervis, 1975].

Each of these argument pieces is defined computationally in BankXX. The definitions of "most on-point," "best," and domain "factor" are based directly on those used in previous systems, such as HYPO and CABARET, occasionally with some modification. For each argument piece, there is a "functional predicate" that determines if a node can supply that useful piece of an argument and a data structure containing an object slot to store entities that satisfy its predicate. For example, the *overlapping cases* argument piece has a predicate to determine if a case shares more than 75% of the factors found in the current problem. BankXX builds up their content incrementally (as its search proceeds) and the collection of all argument

pieces is output to the user at the conclusion of BankXX's processing. There is no argument text generation facility within BankXX, however.

We describe each argument piece in a bit more detail.

1. *Supporting-Cases.* The cases that were decided for the same side – debtor or creditor – as the viewpoint assumed in the current problem situation. Cases in which the plan was confirmed are considered to be decided for the debtor; if the plan was not confirmed, the decision is considered to be for the creditor.

2. *Best-Supporting-Cases.* The "best" cases decided for the viewpoint assumed in the current problem. BankXX uses a variation on the definition of best case used in HYPO [Ashley, 1990]. One requirement of a best case in HYPO was that it be a most on-point case. In some situations, this means that a side may have no "best" cases. In order to assure that each side has at least one "best" case, BankXX includes as "best cases" those supporting cases that are most similar to the problem situation, whether or not they are also most on-point.* Also, best cases in HYPO depended on a sense of whether each dimension favored one side or the other, which is absent from the BankXX implementation.

3. *Contrary-Cases.* A case is contrary if it was decided for the viewpoint opposite of that taken the problem situation.

4. *Best-Contrary-Cases.* Cases decided for the opposing viewpoint that are best cases in the sense described above.

5. *Leading-Cited-Cases.* Leading cited cases are the five cases cited most frequently in the full text opinions of the cases in the BankXX case base. These were identified by analyzing the full text of each of the opinions for the cases in the BankXX case base, extracting the citations, and counting the number of cites to each case.** The leading cited cases in order are:

1. *Rimgale,*
2. *Estus,*
3. *Goeb,*
4. *Deans,*
5. *Iacovoni.*

We note that there is significant overlap between the cases that one might intuitively identify as "leading" in this area and the most frequently cited cases according to this analysis.

---

* Compositionally these definitions of best case can be expressed as follows. In HYPO, best = same-side(mopc (claim-lattice)). In BankXX, best = mopc (same-side (claim-lattice)). The operations of selecting the same-side cases and the most on-point (maximal) cases do not necessarily commute. The only way that BankXX can fail to have a best case for a viewpoint is if there are no same-side cases for the viewpoint.

** Multiple citations within an opinion were not taken into account.

6. *Supporting-Citations.* In BankXX these are defined in the following technical sense: they (i) are found in cases with the desired viewpoint, (ii) point to other cases with the same viewpoint, and (iii) use a "citation signal" indicating that the citing case agrees with the cited case (i.e., *accord, agrees-with, cites, see-eg*) (See Section 3.1.2). For example, the citation "*Estus* agrees with *Flygare*" would be considered supporting if three conditions hold: (1) the citing case (*Estus*) is decided for the current viewpoint; (2) the cited case (*Flygare*) is also decided for the current viewpoint; and (3) the citation signal ("agrees with") indicates that the citing case and cited case are compatible on this point. The usual direct sense of supporting citation as a "cite to a supporting case" is already covered by other argument pieces (i.e., *supporting cases, best supporting cases*), and thus the implementation uses this indirect definition in order to capture citations that were found even though the cited case has not been analyzed (i.e., "closed") by BankXX.

7. *Factor-Analysis.* The set of domain factors that are applicable to the current problem situation. This argument piece is instantiated before commencing the search of the case-domain graph. It is used to create the claim lattice for the problem case. It is also used to determine if a legal theory is applicable to the problem case (when a theory node is chosen from the open list for in-depth analysis).

8. *Overlapping-Cases.* Cases that share a large percentage of factors with the problem case. These cases may not be most on-point cases, but deserve to be captured by research, since they are potentially useful if the most on-point or best cases fail to pan out. The current implementation requires that a case share at least 75% of all the factors applicable to the problem situation.

9. *Applicable-Legal-Theories.* Each theory is defined in terms of a set of domain-specific factors. If all of these prerequisite factors are applicable to the problem situation, then the theory is considered applicable. This is analogous to HYPO's definition of an applicable dimension with respect to a current fact situation. See Figure 6 for an example of a legal theory.

10. *Nearly-Applicable-Legal-Theories.* A theory is nearly applicable to a case if a user-defined threshold percentage of factors defining the theory are applicable to the problem situation. The default threshold in BankXX is 50%.

11. *Factual-Prototype-Story-Category.* These are the story categories that the debtor's factual situation falls under. Each factual story prototype encountered during the search of the case-domain graph is added to this argument piece. The current implementation does not filter these nor does it attempt to determine which of them are in fact appropriate in the current problem case. Story categories were assigned by hand to the cases in the case base. This category is a first attempt

to capture episodic knowledge about a case, that is, a general factual pattern that it entails.

12. *Family-Resemblance-Prototype.* The favorable cases that have the highest family resemblance rating to the given case, according to the Rosch measure of family resemblance [Rosch & Mervis, 1975], with respect to a given family. This argument piece is not fully utilized by the current implementation.

3.2.2. *Argument Factors*

Just as cases may be indexed and compared on the basis of domain factors [Rissland et al., 1984; Ashley, 1990], so may arguments be compared on the basis of what we call *argument factors. Argument factors* are dimensions along which the quality of arguments may be compared and contrasted. They can aid the system in identifying the best arguments (e.g., by sorting arguments according to a partial order based on the factors that apply to an argument). The third type of evaluation function we have experimented with in BankXX is based on these factors (see Section 4.3.3). This function is also used to evaluate the final argument output by BankXX.

We have developed these argument factors on the basis of our own observations and experiences with the types of arguments that are made in legal education and practice. Advances in traditional jurisprudence would be required to provide a firmer basis for argument factors. There have been few programs that attempt to judge the strength of arguments computationally. In fact, there has been very little work on the evaluation of arguments in general. The characterization of an argument's promise in terms of argument factors is a new step for computational legal argument.

We have identified several argument factors. Nine that are implemented in BankXX are:

(1) *centrality-of-theory*,
(2) *win-record-of-theory*,
(3) *win-record-of-theory-for-factual-prototypes*,
(4) *strength-of-best-case-analogies*,
(5) *centrality-of-best-cases*,
(6) *equally-on-point-cases*,
(7) *strength-of-citations*,
(8) *number-of-domain-factors*,
(9) *factual-prototypicality-strength.*

We briefly describe each of these argument factors.

1. *Centrality-of-theory* determines how often the theory has been used, invoked, or compared to a theory used in a case. The idea is that an argument should rely on a well-known theory. Its value is the number of cases in the case base that have applied any of the theories in the *applicable-legal-theories* argument piece.

2. *Win-record-of-theory* determines how often the applicable theories found supporting cases that have been associated with winning arguments for the current viewpoint. For each theory a ratio is computed: the ratio of (a) the number of cases in the case base that applied the theory and are also supporting cases for the current problem to (b) the number of cases in the case base that applied the theory. The value of the factor is the maximum of these ratios over all the theories applicable to the current problem case. This factor rewards theories that have been applied in cases decided for the current viewpoint. It rewards arguments citing applicable theories that have been associated with past winning arguments for the current viewpoint.

3. *Win-record-of-theory-for-factual-prototypes* determines the proportion of cases in which the theory has been successfully used in a case whose facts follow a recognized, stereotypical pattern, such as a student-loan case. For each theory a ratio is computed: the ratio of (a) the number of cases in the case base that (i) exhibit a factual prototype specified in the problem situation, (ii) apply a theory, and (iii) are decided for the current viewpoint to (b) the number of cases in the case that apply that factual prototype and that theory. The value of the factor is the maximum of these ratios computed for each applicable theory. This factor, which is a version of *win-record-of-theory* specifically aimed at factual prototypes, determines the extent to which a theory has been successfully relied on in cases that follow a particular stereotypical pattern.

4. *Strength-of-best-case-analogies* is implemented in terms of the number of domain-specific legal factors that are in common to both the best cases cited in the argument and the current problem. Since it is possible for the cases in the *best-supporting-cases* argument piece to share only a few factors in common with the problem situation, a measure of the number of factors in common is desirable. Best cases that have more factors in common with the current case, all other things assumed equal, are often better cases to use in argument (for instance, because they allow a wider band of analogies to be made). This factor computes the maximum across all the best cases of the ratio of (a) the number of factors in common between a best case and the problem to (b) the total number of factors applicable to the current case.

5. *Centrality-of-best-cases* assesses centrality of the best cases retrieved for the argument. It determines how often those cases have been cited in other cases in the case base. BankXX computes a ratio: (a) the number of best supporting cases that are also leading cited cases, to (b) the number of best supporting cases. This factor is designed to capture the idea that better known best cases provide a stronger argument than obscure ones.

6. *Equally-on-point-cases* measures the proportion of best cases for which there

are no equally on-point cases for the opposing side that share the same subset of dimensions. It provides one indication of how many best cases may not be countered by the opposing cite with a contrary best case that is equally similar to the problem situation.

7. *Strength-of-citations* gives a measure of how often the cases mentioned in the *supporting-citations* argument piece point to *leading cases* or *best cases*. The more citations from the *supporting-citations* argument piece that cite leading cases or best cases, the better. Recall that a citation consists of a citing case, a citation signal, and a cited case. The value of this factor is the ratio of (a) the sum of (i) the number of cited or citing supporting cases that are leading cases and (ii) the number of cited or citing cases that are best cases to (b) the number of cases in the union of the best and leading cited cases.

8. *Number-of-Domain-Factors* computes the number of domain factors that are applicable in a domain-factor node.

9. *Factual-prototypicality-strength* computes a normalized family resemblance rating for the current problem, to determine how prototypical it is for cases of a particular story prototype [Rosch & Mervis, 1975].

We were able to identify many other dimensions along which arguments might be assessed. These include *provenance of support* (cite good courts – and respected jurists – in the proper jurisdictions), *strength of argument type* (straightforward argument types may be considered more desirable than convoluted ones, such as the "turkey, chicken and fish" argument [Skalak & Rissland, 1992]), *abstractness* (appeal to principles or policies versus appeal to rules or cases), *consistency of support* (extent to which cited cases do not undermine propositions in an argument for which they have not been cited), *base of support* (reliance on many weak cases or a few good ones), and so forth.

4. Search in BankXX

In this section we provide details of BankXX's search algorithm for discovering information in the case-domain graph. Section 4.1 describes the neighbor methods used to expand nodes during the search. Section 4.2 describes three heuristic evaluation functions, which we have used in BankXX.

4.1. Comparison with Classic Heuristic Search

In general, classic state-space search is defined by a triple: *initial state, set of operators on states, set of goal states.* In classic best-first heuristic search, there is also a heuristic *evaluation function* that is used to guide the exploration of the

Search Graph: The opened nodes in the case-domain graph (Section 3.1) plus the edges calculated by neighbor methods (Section 4.2).

Initial State: (1) Problem situation or (2) user-specified node in the case-domain graph (this Section).

Operators on States: Set of functions called *neighbor methods* that trace a single link or a sequence of links in the case-domain graph (Section 4.2).

Goal States: None (this Section).

Termination Criteria: (1) Empty open list, or (2) user-specified time bounds exceeded, or (3) user-specified space bounds exceeded (this Section).

Heuristic Evaluation: One of three linear evaluation functions at different levels of abstraction (Section 4.3).

Figure 9. Summary of the search model used by BankXX, with section references in this paper.

state-space [Barr et al., 1981]. BankXX is modeled on the classic algorithm for best-first heuristic search, although with certain key differences. We discuss the similarities and differences here. Figure 9 provides a summary of how BankXX search can be viewed as classic state-space heuristic search.

The Search Space. In BankXX the search space includes the case-domain graph, the semantic network whose nodes represent cases and legal theories from the application domain and whose labeled links of various types represent interconnections. The case-domain graph was described in detail in Section 3.1.

Note that technically the case-domain graph does not constitute the search space for BankXX because although there are no more additional nodes in the search space than are present in the case-domain graph, *there are additional edges* between nodes. These edges are created by the neighbor methods that determine which nodes the search may permissibly move to from its current node in the search space. Thus the true search space is the case-domain graph plus all the edges calculable by the neighbor methods.*

The Start Node. In BankXX the default for the initial state is the user-supplied problem situation, which is represented as a case node in the case-domain graph using the usual representation of a case as a collection of facts. Alternatively, the user can input the problem case but specify another node as the start node, for instance, a favorite or well-known case, like the *Estus* case, if one is known, in order to initiate the search in a particular region of the case-domain graph. In a

* In search parlance, one distinguishes between the *search space* and the *search graph.* [Barr et al., 1981]. The *search space* is the space in which the search implicitly occurs. The *search graph* is that part of the search space that has been actually searched. Thus, the case-domain graph nodes actually opened plus the links actually created by the neighbor methods constitute the search graph. In BankXX, the search graph is never a mere subgraph of the case-domain graph.

companion article on evaluation, we address empirically the impact of start node selection on program performance [Rissland et al., 1995].

The Operators. The set of operators used in BankXX are called *neighbor methods*. These use links in the case-domain graph to generate the "successor" nodes that are opened in search. BankXX has 16 neighbor methods. In general, they are more complex than the simple following of outward arcs from a given node, although some follow in-space or cross-space pointers in a straightforward way. For instance, *case-theory-neighbors* generates all the cases that have applied a particular theory. Others, similar to macro-operators, follow a more complex sequence of links. For instance *theory-case-theory-neighbors* finds all the theories applied by any of the cases the use the theory used in the current node. Neighbor methods are described in detail in the next section, Section 4.2.

Goal Nodes. There are none in BankXX. (Although one could use them in the BankXX architecture, if one so desired.)

Evaluation Function. BankXX can be run with any of three evaluation functions, each of which captures knowledge about legal information or legal argument. These are described in detail in Section 4.3.

The search performed by BankXX differs from the usual heuristic search algorithm in three ways:

1. the richness of node expansions carried out with its neighbor methods,
2. the absence of well-defined goal states,
3. re-evaluation of the nodes on the OPEN list in each search cycle.

All of these differences were motivated by requirements of our problem domain.

In most search applications, node expansion – that is, the generation of successor nodes of the current node – is straightforward (e.g., the generation of valid game moves). In BankXX's node expansion, each neighbor method that is applicable to the current node is invoked; this makes each BankXX neighbor method analogous to a legal move generator for an individual piece in game search. Unlike a game-move generator however, a BankXX neighbor method often involves examination of more than just the current node. For instance, *theory-case-theory-neighbors* involves examination of the theory nodes linked to all the case nodes linked to the theory node linked to the current node. Thus, BankXX node expansions involve more than expansions in typical search applications.

We do not include goal states in our model because of the difficulties inherent in defining an "argument goal" in a way that is consistent with our informal understanding of how humans develop and evaluate legal arguments. It is hard to say in general that an argument does or does not meet some plausible persuasive or rhetorical goal, or even that one has completed the supporting research. In real life situations, one often keeps working until the available time or other resources have been exhausted.

The third difference – re-evaluation of opened nodes – is the real algorithmic departure from the classic heuristic, best-first algorithm for graphs, for instance, as given in *The Handbook of AI*, Vol 1, Chapter 2, Section C3a, p. 61 [Barr et al.,

1981]. In the classic algorithm, one recomputes an evaluation of an opened node only if the node is encountered again, and one updates a value only when the new evaluation is an improvement over the old (i.e., the heuristic estimate of value has gone up).

By contrast, in BankXX all nodes on the OPEN list are re-evaluated and have their values updated in each search cycle. We chose to do this because we wanted the heuristic evaluations of nodes on the OPEN list to reflect the current state of BankXX's problem solving. This is important in the information-gathering task that BankXX is designed to carry out since it is possible for a type of information once considered highly desirable to become less valued, and a once less desirable type of information to become more valued. For instance, if BankXX has harvested more than enough contrary cases, it is more desirable to search for cases of other types than to keep searching for more contrary cases. That is, the values of opened-but-not-yet-closed contrary case nodes should decrease.

In BankXX, the ability to apply an up-to-date view of the state of the problem-solving in assessing open nodes is achieved with the use of heuristic evaluation functions that can dynamically change during the course of the search: that is, a given node can receive different evaluations at different times because of the changed state of the developing argument. The evaluations produced by two of the three evaluation functions that we have used – the argument-piece and argument-factor evaluation functions – typically change during the course of problem-solving. For example, when adequate amounts of information of a particular kind have been gathered by BankXX – supporting cases, contrary cases, etc. – the estimated value of this kind of information decreases.* The third evaluation function – the node-type evaluation function – does not change, and thus, BankXX's search algorithm is essentially the classic one in this case.

Regardless of these differences however, the basic paradigm of BankXX is indeed heuristic search. BankXX builds up the content of the argument pieces by performing heuristic search in its network of domain knowledge. BankXX always begins its processing by analyzing the problem situation for applicable domain factors and computing a claim lattice, which partially orders the cases that have some of the same applicable factors as the current problem. The best and most on-point cases are identified. These provide potential new nodes to be explored and are always the first nodes to be placed on the open list.

BankXX continues by performing a variation on the standard cycle of iterative, best-first search. Neighbors of the current node are generated using BankXX's neighbor methods. Their worth, as well as those already on the OPEN list, are calculated according to the evaluation function. The "best" node on the open list – one with the maximum value under the evaluation function – is identified and then examined by each of the argument pieces in turn in order to determine if it can contribute to that component of the argument. Information that can be is

* This is implemented in the argument-piece evaluation function by zeroing-out a term if its given maximum number of items (i.e., the fill limit) has been gathered. See Section 4.3.2 below.

harvested by the argument pieces and is appended to their data structures. This cycle continues until the search exceeds a user-specified space or time bound (e.g., 30 nodes closed, 1000 billable seconds), or until the open list is empty.

At the conclusion of the search, the argument is assessed in terms of the argument dimensions* and BankXX outputs the argument in a template structured by the argument pieces. In this way the information needed to build up the various argument pieces and ultimately the overall argument is acquired incrementally during the search (see Figure 1).

4.2. Traversing the Case-Domain Graph – Neighbor Methods

The set of nodes to which a search algorithm may permissibly move in legal research is less constrained than in a classical search applications like game-playing since there are no universally agreed upon "rules" on how to generate "moves." In legal research, the number of "legal moves" is extremely large due to the immense variety and volume of legal knowledge and the ways in which it can be manipulated. In the law, there are numbers of sources of compiled knowledge to aid the attorney or legal assistant in the task of uncovering "moves" to be explored in researching an issue. For instance, *Shepard's Citations* gives inter-case and statute-case links for use in tracking down legal materials [*Shepard's*, 1992]. The West Publishing Company has developed a system of keys that index specific areas of legal practice. In addition, there are other, implicit links used in practice that are not reflected in standard materials: links that capture the fact that a case presents an instance of a typical, recurring fact situation, that is, a story; links between a case and the legal theory that is used to decide it, etc.

We have implemented 16 *neighbor methods* to exploit the case-domain graph's high degree of interconnectedness. Each uses links from the case-domain graph to generate (successor) nodes as candidates for examination (nodes on the OPEN list) in BankXX's search of the graph. Neighbor methods are of three types. They:

(1) return nodes that are one intervening case-domain graph link away, typically by following all outbound links of a specific type from the node;
(2) return nodes that are more than one case-domain graph link away, typically by moving to another space and then back again, or
(3) return nodes that are connected by dynamically-created links.

The neighbor methods contain the knowledge of how to move about in the case-domain graph.

Although there are 16 neighbor methods implemented in BankXX, we disabled five of these for the extended example given in Section 5. We did this to make the example easier to follow. The disabled neighbor methods are marked here by a dagger †.

* In the work reported here we do not utilize this assessment. Our intent was to use it as feedback for applying methods of machine learning to improve the system.

Neighbor methods are "methods" in the object-oriented programming sense of a function that applies only to objects of a specific class. Depending on the class or type of the current node (e.g., case, domain-factor analysis, legal theory), which we enumerated in Section 3.1.1, only those neighbor methods that apply to instances of that class will be applicable.

Single-link methods take a case-domain-graph node as input and output all the nodes linked to the input item via one in-space or cross-space link (of a specific type). For example, *legal-theory-link-neighbors* returns all the theory nodes that are linked to a given theory node by any of the nine in-space legal theory space links (e.g., *conflicts-with, rejects, refines*), which we described in Section 3.1.2. The neighbor method *legal-theory-case-neighbors* returns all the cases that apply a theory.

There are ten single-link neighbor methods:

(1) *citation-neighbors* – returns all the citations (nodes in Case Citation Space encoding the citation signal, citing case, cited case, etc.) for a given case (implemented with cross-space case-to-citation links).
(2) *cited-case-neighbors* – returns all the cases (nodes in Fact Situation Space) cited in a given case (via case-to-citation links).
(3) *specific-citation-neighbors* – returns the specific cited case (node in Fact Situation Space) in a given citation node (via citation-to-case links).
(4) *legal-theory-link-neighbors* – returns all the theories (nodes in Legal Theory Space) linked to a given theory (via in-space links in the Legal Theory Space).
(5) *legal-theory-case-neighbors* – returns all the cases (nodes in Fact Situation Space) that apply a given legal theory (via theory-to-case links).
(6) *legal-theory-neighbors* – returns all the theories (nodes in Legal Theory Space) applied in a given case (via case-to-theory links).
(7) *factual-prototype-case-neighbors* – returns all the cases that share a given factual prototype story (via factual-prototype-to-case links).
(8) *factual-prototype-neighbors* – returns the factual prototype stories for a given case (via case-to-prototype links).
(9) *domain-factor-analysis-neighbors* – returns the domain-factor-analysis node for a given case (via case-to-domain-factor links).
(10) *domain-factor-neighbors* – returns the domain-factor-analysis nodes whose applicable factors include the factors of the given domain-factor node.*

Boomerang neighbor methods move from one space to another and then back again. They are similar to macro-operators [Fikes, Hart & Nilsson, 1972] in that they collapse a series of link traversals into a single operator in order to perform a retrieval that has been recognized as useful by legal researchers. They permit indirect linking between nodes of a given type.

* If the universe of factors were restricted to those found in a particular problem case (current fact situation), this method in effect would return the clusters of factors found in more on-point cases. This method simply examines set-subset relations between domain factor clusters.

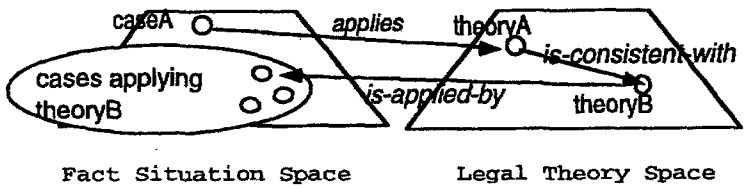

Figure 10. Tracing a series of links using the boomerang neighbor method *case-theory-theory-case.*

For example, the method *case-theory-theory-case-neighbors* starts with a case, follows the link – a case-to-theory cross-space link – to the first theory listed as applied in that case, then follows links – within the Legal Theory Space – from that theory to each theory that has been explicitly referred to in a favorable way (e.g., by an *agrees-with* link) by that theory, and then follows links – theory-to-case cross-space links – back to the cases that have applied the favorably viewed legal theories. The *case-theory-theory-case-neighbors* method returns cases that have applied similar theories, and provides a useful collection of cases to examine in the next stage of the search (Figure 10).

The particular traversal path can be read directly from the method's name. For instance, *case-theory-theory-case-neighbors* moves from a case to a theory, to other theories, and then, back to other cases.

There are four boomerang neighbor methods:

(1) *case-theory-theory-case-neighbors* – goes from a case to a theory applied in the case to other theories favorably viewed by the applied theory and then returns all the other cases that also applied to favorably viewed theories.

†(2) *case-citation-citation-case-neighbors* – goes from a case to all its citations, then to other citation nodes, and back to case nodes. This method returns all cases cited by the cases that have themselves been cited by the input case. This method returns the frontier of case-domain-graph nodes that are two citation links away from the input case node.*

†(3) *case-factual-prototype-case-neighbors* – goes from a case to its factual prototype and returns all the cases that also share that prototype.

†(4) *factual-prototype-citation-neighbors* – goes from a factual prototype to all the cases that share the prototype, and then returns all the cases cited in any of those cases.

Dynamic neighbor methods create links dynamically. They take the problem case as a functional parameter, so that depending on the current problem different paths can be traced through the case-graph from a given node. There are two

* Another way to say this is: if Case-x cites Case-y, and Case-y cites Case-z, this function takes as input Case-x and returns all cases like Case-z.

dynamic neighbor methods. Both use the "family resemblance measure" of case prototypicality [Rosch & Mervis, 1975]:

†(1) *family-resemblance-for-prototype-neighbors* – returns the case with the highest family resemblance score of those cases sharing its factual prototype.

†(2) *family-resemblance-neighbors* – returns the five cases having the highest family resemblance score to the current problem situation.

4.3. Heuristic Evaluation Functions

BankXX uses three different types of evaluation functions. They differ in the level of abstraction they use to evaluate case-domain-graph nodes. All are simple linear functions. They form a progression of increasingly more informed evaluation methods, whose considerations range from (1) only the type of information encoded in a node, to (2) the contribution of a node to the standard argument pieces, and (3) the incremental impact of a node on the overall state of the evolving argument, as assessed with the argument factors.

The three evaluation functions and their levels of abstraction are:

1. The *node-type evaluation function* – domain (node) level;

3. The *argument-piece evaluation function* – argument-piece level;

2. The *argument-factor evaluation function* – argument level.

Each function can be described in a canonical way as a weighted sum $\sum w_i f_i$. What differs among them are the weights and terms used. In particular, the information used to compute the value of a term varies from the simplest tag, as to the type of a node being evaluated, to a value, based on an argument factor, of a node's contribution to the emerging argument.

All three functions give greater weight to terms associated with legal theories. In our own experience as well as in reference works on research methods [Kunz et al. 1992], there is evidence for the importance of legal theories in legal research. The five authors of this legal research methods text performed an experiment in which one of them posed a hypothetical research situation for the others to research. In the recorded protocols, one of the authors, a reference librarian, researched the case thoroughly. On the other hand, the three other authors "chose to stop when he or she determined that there were some legal theories to pursue on behalf of the client." [Kunz et al. 1992, p. 454]. Thus biasing the BankXX evaluation functions toward legal theories appears to make good sense.

4.3.1. *The Node-Type Evaluation Function*

The form of the *node-type evaluation function* is $\sum w_i f_i(c)$. The f_i are *type-predicates* that check the *type* (e.g., legal case, legal theory) and some other basic information about the node c. The w_i are non-negative, scalar weights. This function has the general form:

$$w_1\ \textit{type-pred}_1(c) + w_2\ \textit{type-pred}_2(c) + \ldots + w_n\ \textit{type-pred}_n(c).$$

1. *Theory nodes*	8
2. *Cases-as-facts nodes*	6
3. *Citation-bundle nodes*	5
4. *Domain-factor nodes*	4
5. *Story Prototype nodes*	3

Figure 11. Types of terms and their weights used in the node-type evaluation function.

For a given node, the node-type evaluation function can have at most one non-zero term since only one of the type-predicates can be non-zero. Thus the value returned is simply the weight given to the non-zero term. It causes node-types to be examined in the order defined by the weights.

The node-type evaluation function is deliberately coarse-grained and simple. It provides a baseline. Essentially the question asked by this evaluation function is: "How well will this node contribute information of a type known to be useful to argument?"

Legal theories are given some preference but there is not much difference among the values given to the other types. In this implementation, we included only five terms; we did not use the prototypicality (family resemblance) view of cases. The types of nodes and their weights are given in Figure 11.

All five type-predicates test the type of the node. The three type-predicates for nodes of the citation-bundle, domain-factor and story-prototype types, perform additional tests of the node. There are no additional tests for theory nodes and cases-as-facts nodes; the evaluation function tests only the type of those nodes. For a type of node having an additional test, if the test is not satisfied, the value of the type-predicate is 0, and the value returned by the node-type evaluation function is 0. If the additional test is satisfied, the type-predicate is 1 and the function returns the weight listed in Figure 11.

The additional test performed on citation-bundle nodes checks whether the citing case was decided for the point of view assumed in the problem case. The rationale for this additional test is that citations made in cases decided for the problem case's point of view are more likely to provide useful argument support than those cited in cases decided for the opposing side.

The additional test performed on domain-factor analysis nodes checks whether more than a threshold number of domain factors are applicable in the node being evaluated. The rationale for this test is the heuristic that a "fat" domain-factor case analysis where a substantial number of factors apply is more likely to contribute to the argument than a "lean" one where only a few factors apply. The threshold is set at 10 applicable factors. (Recall BankXX uses 26 domain factors. See Appendix C.)

The additional test for story-prototype nodes checks whether the story of the story-prototype node being evaluated is a story for the problem case. The rationale is that story-prototype nodes representing other factual scenarios may not provide useful leads in researching a problem that represents a different legal story.

The additional tests yield somewhat finer distinctions in the evaluation of case-domain-graph nodes of the three types. For instance, if the case-domain-graph node being evaluated is the *dishonest debtor* story-prototype node, but the problem case is not about a dishonest debtor – say it's a *student loan* case – then the story-prototype term would be assigned weight 0 because the story prototypes are different. By contrast, if the story-prototype node being evaluated were *student loan*, the weight for that term in the evaluation function would be 3.

4.3.2. *The Argument-Piece Evaluation Function*

The form of the *argument-piece evaluation function* is $\sum w_i f_i(c, a)$, where c is the current node and a is the current state of the argument. Each f_i is an *argument-piece-predicate* that tests whether a particular *argument piece* is fillable by the current node *and* whether that argument piece has *not* already been completely filled (i.e., not reached its *fill limit*). If so, f_i returns 1; else, 0. It is of the form:

$$w_1 \textit{ arg-piece-pred}_1(c, a) + w_2 \textit{ arg-piece-pred}_2(c, a) + \ldots + w_n \textit{ arg-piece-pred}_n(c, a)$$

Essentially the question asked by this evaluation function is: "Can the particular domain knowledge contained in this node be used to fill one of the desired components of an argument that has not already been completely filled?" This intermediate-level evaluation function prevents BankXX from wasting computing resources by unnecessary bolstering of parts of the argument that are already well-established. It captures our intuition that there are limits on how much information of a particular kind is desirable to gather, even for information of a type having high importance. In other words, it is more desirable to have a balance of information of various useful kinds than a surfeit of information of one kind and a possible deficit of other kinds.

Note that all the argument-piece-predicates are computed with respect to the problem case, not with respect to the current node. So, for example, the *applicable-theories argument-piece-predicate* is only true (and returns 1) if the current node is a legal theory that applies *to the problem situation* posed to the system and BankXX has not found its fill of such theories.

In our experiments this evaluation function had only ten terms since we did not use the *family resemblance prototype* and *domain factors* argument pieces.* The function has five terms that apply to cases, one that applies to each of citation, domain factor and factual prototype nodes, and two that apply to theories. (The *overlapping cases* argument-piece-predicate applies to domain-factor analysis nodes.) The actual terms, weights and fill limits (given in brackets) are given in Figure 12.

* Since the domain factor analysis is constant throughout a given problem case (it is computed but once, initially, at the beginning of the search), it simply contributes a constant value to the evaluation function; thus, we left it off.

1. *supporting cases*: weight=2 [limit=3]
2. *best supporting cases*: 7 [5]
3. *contrary cases:* 1 [3]
4. *best contrary cases:* 5 [3]
5. *leading cases*: 6 [5]
6. *supporting citations*: 1 [5]
7. *overlapping cases*: 1 [5]
8. *applicable legal theories*: 8 [6]
9. *nearly applicable legal theories*: 6 [3]
10. *factual prototype stories*: 6 [1]

Figure 12. Terms, weights, and fill limits, given in brackets, for the argument-piece evaluation function.

The argument-piece evaluation function is biased somewhat towards legal theories, best supporting cases, leading cases, and stories. Note that a case that fits into more than one category can have quite a high score due to the additive nature of the function. For example, a *leading-case* like *Rimgale* that in a given problem situation is also a *supporting-case*, would have an evaluation score of 8 (= 2 + 6). If it were also a best case, its score would be 15 (=2 + 7 + 6).

Note, that the value an item receives depends on the state of BankXX's problem-solving. For instance, suppose *Rimgale* is not a best case and that at some point, the *supporting-cases* argument piece has reached its fill limit of 3 cases, and the *leading-cases* argument piece has not, then the evaluation score for *Rimgale* would be only 6.

It is important to note that when BankXX is run with the argument-piece evaluation function, the fill limits on the various argument piece terms *also determine the argument pieces for which an item is actually harvested.* It is possible for a case to be ignored for certain argument pieces that it ostensibly fits because these categories are *already filled* with harvested information. For instance, in the *Rimgale* example of the previous paragraph, *Rimgale* would only be harvested – and listed in the output – under *leading-cases* even though it is also a *supporting-case.* Note that this filling up of a category is reflected in the evaluation score awarded a node: the argument-piece-predicate for a filled-up piece is always 0. That is, a filled piece has a zeroed-out term in the evaluation function. Thus, one can say that information is harvested only for those argument pieces for which there is a compelling rationale to do so, that is, a non-zero contribution to the evaluation score. In the *Rimgale* example here, only the *leading-cases* term will contribute a non-zero value because the *supporting-cases* term will be zeroed out.

In summary, when BankXX is run with the argument-piece evaluation function, the fill limits on argument pieces affect BankXX in two ways:

(1) by causing terms in the evaluation function for argument pieces filled to their limits to be zeroed out, and

1. centrality-of-theory	8
2. win-record-of-theory	8
3. win-record-of-theory-for-factual-prototype	8
4. strength-of-best-case-analogies	5
5. centrality-of-best-cases	5
6. equally-on-point-cases	4
7. strength-of-citations	4
8. number-of-domain-factors	4
9. strength-of-factual-prototype	3

Figure 13. Terms and weights for the argument-factor level evaluation function.

(2) by causing closed nodes not to be harvested by completely filled argument pieces.

4.3.3. *The Argument-Factor Evaluation Function*

The form of the *argument-factor evaluation function* is $\sum w_i f_i(c, a, a^*)$, where a^* is the provisional argument that would result from incorporating node c into the current argument a. The f_i are *argument-factor functions* that compare the value of each *argument factor* applied to the current argument a with its value on the provisional argument a^*. The value returned is a weighted sum of terms that represent the improvement in an argument that would be realized by adding node c to the argument pieces whose requirements it satisfies.

The argument-factor evaluation function is of the form:

$$w_1\ \textit{arg-factor-fcn}_1(c, a, a^*)\ +\ w_2\ \textit{arg-factor-fcn}_2(c, a, a^*) + \ldots + w_n\ \textit{arg-factor-fcn}_n(c, a, a^*)$$

Essentially the question asked by this evaluation function is: "Can the domain knowledge contained in this node improve the quality of the argument?" Nine argument factors are used in this evaluation function. The actual terms and weights are given in Figure 13.

The argument-factor evaluation function is computed in the following way. Each argument-factor function *arg-factor-fcn*$_i$ corresponds to an argument factor (see Section 3.2.2). For each argument factor, BankXX calculates its value (i) when applied to the current argument (without node c) and (ii) when applied to the provisional argument. The provisional argument is created from the current argument by temporarily adding node c to the argument pieces whose requirements it satisfies.

Each *arg-factor-fcn*$_i$ is a binary function that returns 1 or 0, according to whether adding node c to the current argument would improve or fail to improve the argument with respect to its corresponding argument factor. If the value of the argument factor is higher in the provisional argument, the *arg-factor*-returns 1; else, it returns 0. The argument-factor evaluation function returns the weighted

sum of the binary values returned by each *arg-factor-fcn*$_i$. Note that for a given node an argument-factor function can return different values at different times since its computation takes into account the *current* state of the argument.

The argument-factor evaluation function is computationally expensive since it requires creating a provisional argument for each node on the open list and computing the value of each argument factor for it. To accelerate these computations, two argument-factor functions do not create a provisional argument in the current implementation: *number-of-domain-factors* and *strength-of-factual-prototype*. While in previous implementations these two argument-factor functions were computed in the same manner as the others, in the current version of the system these functions only consider the current node and the problem case.

The *number-of-domain-factors* function returns 1 if more than a threshold number of domain factors applies to the current domain-factor node, and 0 otherwise. The threshold is set at 10. The motivation behind this is that the more domain factors that are applicable, the more likely there will be a non-trivial intersection of domain factors with the problem case.

The *strength-of-factual-prototype* function simply returns 1 if the current factual prototype node is the same as a factual prototype story of the problem case, and 0 otherwise. In previous implementations, an expensive calculation was performed to determine the family resemblance between the current case and the other cases that share the same factual prototype. Family resemblance is a measure of prototypicality developed by Rosch and Mervis [1975].

Part III: An Extended Example

5. Example

To illustrate the workings of BankXX, we step through a run of BankXX on an actual legal case *In re Estus*, 695 F.2d 311 (8th Cir. 1982) where BankXX uses the domain representation described in Section 3 and the search algorithm described in Section 4. The cases in BankXX's case-knowledge base are listed in Appendix A. As we described in Section 2, *Estus* is a leading case in this area.

We have included this extended example in order to demonstrate how the program uses heuristic search to fill in an argument template. Readers not particularly interested in the details of the program's operation or this aspect of Chapter 13 bankruptcy law may wish to skim this section, and observe generally the variety of ways that legal sources are accessed via heuristic search and how the argument template is filled incrementally.

5.1. The Estus Case

The facts of the case are these. The debtors, the Estuses, filed a Chapter 13 petition in 1980, listing almost $11,000 in debts to some 30 unsecured creditors, including almost $3000 in student loans from the appellant, the U.S. government, which was

a holder of an unsecured claim arising from a Veterans Administration educational loan. The Estuses also owed secured debts to a furniture and a piano store. They were also five months in arrears on the mortgage on income-producing rental property they owned. The Estuses had a monthly income of $745, and monthly expenses of $492, leaving a surplus of $253. They proposed to pay $250 each month, all of which was to be applied toward the two secured debts and the mortgage payments. No payments were to be made under the plan to any of the unsecured creditors. The term of the proposed plan was only 15 months.

Procedurally, this case was appealed by the U.S. government from a decision of the U.S. District Court for the Eastern District of Arkansas, upholding the Bankruptcy court's confirmation of the debtors Ronald and Doris Estus's Chapter 13 plan.

5.2. RUNNING BANKXX ON ESTUS

We treat *Estus* as a new fact situation, which we call the *Estus-problem* case. In order to treat *Estus* in a *de novo* manner, the real *Estus* case was excised from the case-domain graph. We did retain the node for the so-called *Estus* theory. However, there is no connection in the case-domain graph between the Estus-problem case and the *Estus* theory; it is as though the *Estus theory* is a legal theory that arose from some other case.

We configured BankXX with the argument-piece evaluation function and the following system parameters: close at most 30 nodes; use no more than 1,000 billable seconds*; start at a randomly-chosen most on-point case for the Estus-problem case; take the point of view of the creditor, the U.S. government. We used only 11 of the 16 implemented neighbor methods (see Section 4.2).** In effect, for this example we ran BankXX without any of the mechanisms that concern family-resemblance.

We note that fill limits (e.g., 3 cases each for ordinary *supporting-cases* and *contrary-cases*) will play a role in the extended example. (See Section 4.3.2.)

5.3. EXTRACTS FROM THE PROCESSING OF ESTUS

Before search begins, the problem case is analyzed to determine which domain factors apply to it. A standard claim lattice [Ashley, 1990] is built based on this analysis. The factor analysis is stored in the *factor-analysis* argument piece, whose sole function is to hold the results of this analysis.

---

* Recall that *billable seconds* is the time that BankXX takes from the moment it is invoked until it returns its output, and includes any time spent on garbage collection and output to the screen during intermediate processing.

** Neighbor methods with daggers in Section 4.2 were disabled. They are: the two family resemblance methods, *family-resemblance-for-prototype-neighbors* and *family-resemblance-neighbors*, and three of the four boomerang methods, *case-citation-citation-case-neighbors*, *case-factual-prototype-case-neighbors*, and *factual-prototype-citation-neighbors*. All of these can return a large number of neighbors which would have complicated the presentation of the example.

cases:
[AKIN]

Figure 14. The open list at the start of the first cycle for the Estus-problem case.

Of the 26 factors used in BankXX, the following are applicable in the Estus-problem case:

percent-surplus-of-income-factor
employment-history-factor
earnings-potential-factor
plan-duration-factor
preferential-creditor-treatment-factor
debt-type-factor
nondischarge-7-factor
relative-educational-loan-debt-factor
repayment-unsecured-debt-factor
necessary-expenses-minus-plan-payments-factor
likelihood-income-increase-factor

Creating the claim lattice allows the system to extract the most on-point cases from the claim lattice for the Estus-problem case: *Gunn*, *Gibson*, and *Akin*. See Figure 15. One case is chosen at random from among the most on-point cases: *Akin*. *Akin* is added to the open list. See Figure 14. This completes the initialization of BankXX on the *Estus* problem.

With one case to start with on the open list, BankXX enters the basic iterative portion of its execution cycle:

(1) evaluate each node on the open list and select one with the maximum value. Remove this maximal node – the new current node – from the open list and place it on the closed list;
(2) apply the predicate of each of the argument pieces to the current node, and append the current node to each argument piece whose predicate it satisfies;
(3) generate the neighbors of the current node using all the neighbor methods, and place them on the open list of potential nodes to visit.

This three-stroke cycle is then repeated, selecting a new current node each time.

Cycle 1

Since [AKIN] is the only node on the open list to start with, the first step is trivial in this first cycle. [AKIN] is the maximal node. It is selected as the new current node and moved to the closed list.

In the second step, each one of the 12 argument pieces examines [AKIN] to determine if it can harvest information from [AKIN] that can be used for the argument piece. *Supporting-cases*, which is a repository for cases that were decided for the same vantage point as the side taken by BankXX in the current case (i.e.,

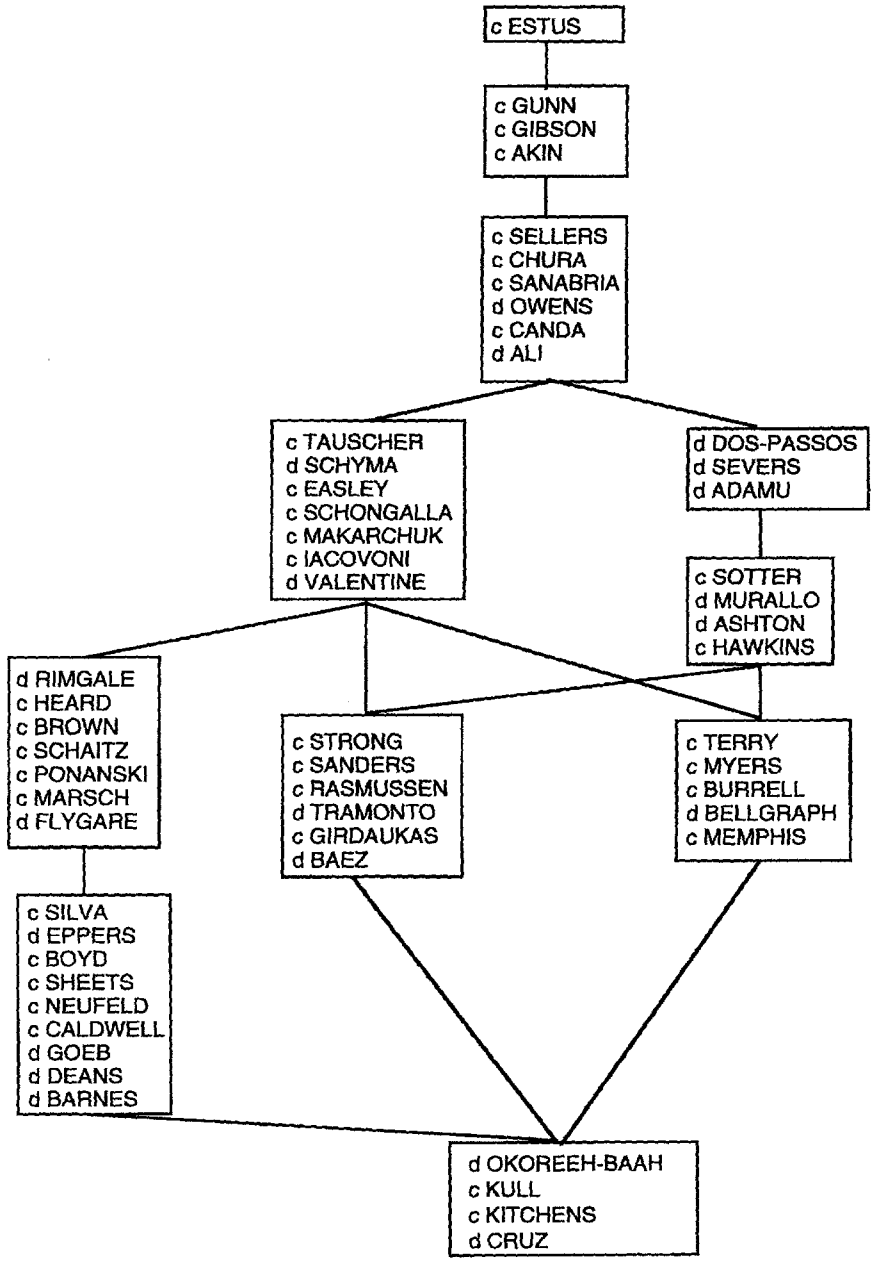

Figure 15. The claim lattice for the Estus-problem case. A **c** indicates that the case was won by the creditor. A **d** indicates that the case was won by the debtor.

SUPPORTING-CASES: ([AKIN])
BEST-SUPPORTING-CASES: ([AKIN])
CONTRARY-CASES: NIL
BEST-CONTRARY-CASES: NIL
LEADING-CASES: NIL
SUPPORTING-CITATIONS: NIL
FACTOR-ANALYSIS: ([ESTUS-PROBLEM-CASE-FACTOR-ANALYSIS])
OVERLAPPING-CASES: NIL
APPLICABLE-LEGAL-THEORIES: NIL
NEARLY-APPLICABLE-LEGAL-THEORIES: NIL
FACTUAL-PROTOTYPE-STORY-CATEGORY: NIL
FAMILY-RESEMBLANCE-PROTOTYPE: NIL

Figure 16. The argument template after one search iteration on the Estus-problem case.

the creditor), can use [AKIN] simply because it was decided for the creditor. *Best-supporting-cases* can also use [AKIN] because it is a most on-point case that has been decided for the creditor, and therefore is a best supporting case. [AKIN] is thus harvested by both *supporting-cases* and *best-supporting-cases*. Only these two argument pieces can glean information from [AKIN]; the other argument pieces cannot. For instance, because *Akin* is not one of the top five leading cases, the *leading-cited-case* argument piece passes over [AKIN]. Because [AKIN] does not share 75% or more of the Estus-problem case's domain factors, the *overlapping-cases* argument piece does not harvest [AKIN]. As a result of the first search cycle, the argument appears as in Figure 16.

In the third step BankXX uses the current node to locate new nodes to open. To do this, BankXX applies all of its 11 non-disabled neighbor methods (given in Section 4.2) to the current node. Six neighbor methods apply to [AKIN]; four of these yield new nodes for the open list. The remaining five neighbor methods do not apply to [AKIN]. The details are as follows:

> The *citation-neighbors* method applies and yields two citations: [AKIN-CITES-ESTUS] and [AKIN-SEE-ESTUS]. However, since both citations cite *Estus*, they cannot be used.*

> The *cited-case-neighbors* method follows the case-to-citations links from the [AKIN] case node to the citation nodes [AKIN-CITES-ESTUS] and [AKIN-SEE-ESTUS] and returns the cases from the citations. Both the citations yield the *Estus* case. But since *Estus* is the problem case and therefore is not an admissible node in the search, [ESTUS] is not added to the open list.

* While most linkages can be removed before processing a case in a *de novo* manner, some information is removed only when it is encountered. The removal is automatic.

The *specific-citation-neighbors* method does not apply to [AKIN] because it is not a citation node.

The *legal-theory-link-neighbors* method does not apply because [AKIN] is not a theory node.

The *legal-theory-case-neighbors* method does not apply to [AKIN] either.

The *legal-theory-neighbors* method returns the [ESTUS-LEGAL-THEORY] because it is represented as having been applied in [AKIN].

The *factual-prototype-case-neighbors* method does not apply to [AKIN] because [AKIN] is not a factual prototype node.

The *factual-prototype-neighbors* method yields the [STUDENT-LOAN-FACTUAL-PROTOTYPE] because [AKIN] was tagged as a student loan case.

The *domain-factor-analysis-neighbors* method returns the factor analysis of the *Akin* case, [AKIN-FACTOR-ANALYSIS], which was computed previously and stored.

The *domain-factor-neighbors* method does not apply to [AKIN] because [AKIN] is not a domain factor node.

The *case-theory-theory-case-neighbors* boomerang method looks at the cases that apply any of the theories that are linked to the theories applied in the *Akin* case. Only the [ESTUS-LEGAL-THEORY] is applied in *Akin*. The cases applying the theories linked to the [ESTUS-LEGAL-THEORY] are [MAKARCHUK], [CRUZ], [BAEZ], [ASHTON], [BURRELL], [RASMUSSEN], [CHURA], and [ALI].

The neighbor methods have discovered 11 nodes. All of these are placed on the open list. This concludes the first cycle.

Cycle 2

At the start of the second cycle, there are now many nodes on the open list. Applying the argument-piece evaluation function to each of opened nodes yields the open list with accompanying evaluations shown in Figure 17.* For instance, [ALI] is scored a 6 because it is both a contrary case (worth 1 point) and a best

* Note that applying the argument-piece evaluation function is not as onerous as it might appear; most of the terms can easily be computed by a simple check of the node or the claim lattice for the problem case.

theories:
[ESTUS-LEGAL-THEORY] 6

factual prototypes:
[STUDENT-LOAN-FACTUAL-PROTOTYPE] 6

factor analyses:
[AKIN-FACTOR-ANALYSIS] 1

cases:
[ALI] 6, [BURRELL] 2, [CHURA] 2, [MAKARCHUK] 2, [RASMUSSEN] 2, [ASHTON] 1, [BAEZ] 1, [CRUZ] 1,

Figure 17. The open list at the start of the second cycle from the Estus-problem case. For convenience, we show the nodes sorted according to type and value.

contrary case (worth 5 points). Note BankXX's definition of "best" is at work here since [ALI] is not in a child node of the root of the claim lattice (see Figure 15). The [AKIN-FACTOR-ANALYSIS] receives a score of 1 because it qualifies as *overlapping* since it shares at least 75% of the factors of the problem situation (See Section 3.2.1).

BankXX picks the node with the maximum evaluation function score. The tie-break policy for choosing among nodes with maximal non-zero evaluations is to pick a random theory if a theory is present, and otherwise to pick the maximal item that was first added to the open list. If all nodes have 0 value, one is picked at random. The [ESTUS-LEGAL-THEORY] is chosen. BankXX "moves" to it – making it the current node – from the [AKIN] node, which was closed in the first cycle. It is then placed on the closed list.

Having chosen [ESTUS-LEGAL-THEORY] as the new current node, the second step in this second cycle is for all the argument pieces to examine it. In fact, only the *applicable-legal-theories* and *nearly-applicable-legal-theories* argument pieces have to consider [ESTUS-LEGAL-THEORY] in any detail since they are the only neighbor methods that pertain to theory nodes. Since the Estus-problem case does not present enough information to conclude that all the prerequisite factors of the *Estus* theory apply to it, [ESTUS-LEGAL-THEORY] cannot be used by the *applicable-legal-theories* argument piece. However since more than 50% of the prerequisite factors do apply to the problem situation, it can be used by the *nearly-applicable-legal-theories* argument piece. Figure 18 shows the emerging argument for the Estus-problem case after [ESTUS-LEGAL-THEORY] has been harvested in cycle 2.

Having completed the harvesting step – in which the [ESTUS-LEGAL-THEORY] has been added to the *nearly-applicable-legal-theories* argument piece – the third step in this second cycle is to use [ESTUS-LEGAL-THEORY] to generate

SUPPORTING-CASES: ([AKIN])
BEST-SUPPORTING-CASES: ([AKIN])
CONTRARY-CASES: NIL
BEST-CONTRARY-CASES: NIL
LEADING-CASES: NIL
SUPPORTING-CITATIONS: NIL
FACTOR-ANALYSIS: ([ESTUS-PROBLEM-CASE-FACTOR-ANALYSIS])
OVERLAPPING-CASES: NIL
APPLICABLE-LEGAL-THEORIES: NIL
NEARLY-APPLICABLE-LEGAL-THEORIES: (**[ESTUS-LEGAL-THEORY]**)
FACTUAL-PROTOTYPE-STORY-CATEGORY: NIL
FAMILY-RESEMBLANCE-PROTOTYPE: NIL

Figure 18. The argument template on the Estus-problem case after two search cycles. The newly harvested [ESTUS-LEGAL-THEORY] is shown in bold.

more nodes for the open list. The following two neighbor methods are applicable to [ESTUS-LEGAL-THEORY]:

(1) The *legal-theory-link-neighbors* method yields the theories linked to [ESTUS-LEGAL-THEORY]: [MAKARCHUK-PRINCIPAL-PURPOSE-STUDENT-LOAN-DISCHARGE-LEGAL-THEORY], [KITCHENS-KULL-THEORY], [PER-SE-MINIMUM-PAYMENT-REQUIREMENT-LEGAL-THEORY] and [FLYGARE-THEORY]. (See Figure 7.)

(2) The *legal-theory-case-neighbors* method returns a list of cases that have applied the [ESTUS-LEGAL-THEORY]: [CHURA], [FLYGARE], [SANDERS], [SCHYMA], [SELLERS], and [VALENTINE].

Thus the *legal-theory-link-neighbors* neighbor method discovers four new theories by following outward Theory-space links from the [ESTUS-LEGAL-THEORY]. The *legal-theory-case-neighbors* method adds five new cases. The open list after these additions is given in Figure 19.* Note that [CHURA] was already on the open list and that there have been no changes in the evaluations of any of the nodes previously on the list (see Figure 17). Also note that the *Makarchuk* theory receives a 0 score; this is due to the fact that it fails to meet even the threshold required for a *nearly-applicable-legal-theory*. There are now 19 nodes on the open list.

Cycles 3, 4, 5

In its third, fourth, and fifth cycles, the system closes in turn [KITCHENS-KULL-LEGAL-THEORY] and [PER-SE-MINIMUM-PAYMENT-REQUIREMENT-LEGAL-THEORY], both of which qualify as *applicable-legal-theories* (and have maximal scores of 8), and the [FLYGARE-LEGAL-THEORY], which qualifies as

* This is actually the open list after evaluation scores are computed in the first step of cycle 3.

theories:
[PER-SE-MINIMUM-PAYMENT-REQUIREMENT-LEGAL-THEORY] 8, [KITCHENS-KULL-LEGAL-THEORY] 8, [FLYGARE-LEGAL-THEORY] 6, [MAKARCHUK-PRINCIPAL-PURPOSE-STUDENT-LOAN-DISCHARGE-LEGAL-THEORY] 0

factual prototypes:
[STUDENT-LOAN-FACTUAL-PROTOTYPE] 6

factor analyses:
[AKIN-FACTOR-ANALYSIS] 1

cases:
[ALI] 6, [BURRELL] 2, [CHURA] 2, [MAKARCHUK] 2, [RASMUSSEN] 2, **[SANDERS] 2, [SELLERS] 2,** [ASHTON] 1, [BAEZ] 1, [CRUZ] 1, **[FLYGARE] 1, [SCHYMA] 1, [VALENTINE] 1.**

Figure 19. The open list at the start of the third cycle on the Estus-problem case. Nodes are shown sorted according to type and value. Nodes opened in the second cycle are shown in bold.

a *nearly-applicable-legal-theory* and is chosen before the [ALI] and [STUDENT-LOAN-FACTUAL-PROTOTYPE] nodes, which also have scores of 6, because it is a theory. The *Makarchuk* theory remains on the open list because of its low evaluation score. The bias of BankXX for legal theories is evident here. At the completion of cycle 5 (after five nodes have been closed), the argument is as shown in Figure 20. At this point, BankXX has discovered and harvested information from a region of the Theory-space that provides theories useful to an argument for the Estus-problem case.

Cycle 6

The next node that is closed is the [ALI] case, which was placed on the open list by the *case-theory-theory-case* neighbor method back in cycle 1. (It is chosen before the equally-rated [STUDENT-LOAN] node because it is older.) This node turns out to be important since *Ali* is a useful case in many respects. Without going through all the details of the processing, *Ali* is harvested by the *best-contrary-cases* argument piece since it is one of the maximally on-point cases of those decided for the debtor (see Figure 15) and thus is a *best* case according to BankXX's definition of *best* (see Section 3.2.1). It is also harvested by the ordinary *contrary-cases* argument piece. *Ali*, in addition to providing another citation for the [FLYGARE-LEGAL-THEORY], cites [BARNES], [KITCHENS], [GOEB], [RIMGALE] and [DEANS]. These cases are added to the open list by the *cited-case-neighbors* method. [GOEB], [RIMGALE] and [DEANS] are all both contrary cases and leading-cited cases and thus are highly rated (with scores of 7). These three cases

SUPPORTING-CASES: ([AKIN])
BEST-SUPPORTING-CASES: ([AKIN])
CONTRARY-CASES: NIL
BEST-CONTRARY-CASES: NIL
LEADING-CASES: NIL
SUPPORTING-CITATIONS: NIL
FACTOR-ANALYSIS:
([ESTUS-PROBLEM-CASE-FACTOR-ANALYSIS])
OVERLAPPING-CASES: NIL
APPLICABLE-LEGAL-THEORIES: **([KITCHENS-KULL-LEGAL-THEORY]**
[PER-SE-MINIMUM-PAYMENT-REQUIREMENT-LEGAL-THEORY])
NEARLY-APPLICABLE-LEGAL-THEORIES: ([ESTUS-LEGAL-THEORY]
[FLYGARE-LEGAL-THEORY])
FACTUAL-PROTOTYPE-STORY-CATEGORY: NIL
FAMILY-RESEMBLANCE-PROTOTYPE: NIL

Figure 20. Intermediate argument for the Estus-problem case after the fifth cycle (after five nodes have been closed). Nodes that have been added since the second cycle are shown in bold.

become the most highly ranked items on the open list. [BARNES] and [KITCHEN] are ordinary contrary and supporting cases, respectively, and are scored accordingly (only as 1 and 2).

Cycles 7 & 8

In cycle 7, [RIMGALE] is chosen randomly from one of the three items with maximal scores and is closed. It is harvested by both *leading-cases* and *contrary-cases*. [RIMGALE] applies a legal theory, which uses the definition of "good faith" that appears in the old bankruptcy statute: the [OLD-BANKRUPTCY-ACT-GOOD-FAITH-DEFINITION-LEGAL-THEORY]. This theory is unearthed by the *legal-theory-neighbors* method, which chases pointers from a case to each of the legal theories discussed in it. Since this theory is applicable to the Estus-problem, it is rated very highly (8) and becomes the new maximal node. In cycle 8, it is closed and harvested. At this stage the argument is beginning to be fleshed out quite well, with many of the argument pieces containing information. See Figure 21.

Cycle 9

The next item to be harvested is the [GOEB] case, which is both a leading-cited case and a contrary case. It is harvested by both the *contrary-cases* and *leading-cases* argument pieces. Note that this is the third contrary case that BankXX harvests and it thus fills up the *contrary-case* argument piece, which has a fill limit of 3. From now on, the term for the *contrary-case* argument piece will contribute 0 to the overall heuristic evaluation. Any case that is only a contrary case is now scored

SUPPORTING-CASES: ([AKIN])
BEST-SUPPORTING-CASES: ([AKIN])
CONTRARY-CASES: (**[ALI], [RIMGALE]**)
BEST-CONTRARY-CASES: (**[ALI]**)
LEADING-CASES: (**[RIMGALE]**)
SUPPORTING-CITATIONS: NIL
FACTOR-ANALYSIS: ([ESTUS-PROBLEM-CASE-FACTOR-ANALYSIS])
OVERLAPPING-CASES: NIL
APPLICABLE-LEGAL-THEORIES: ([KITCHENS-KULL-LEGAL-THEORY]
[PER-SE-MINIMUM-PAYMENT-REQUIREMENT-LEGAL-THEORY]
[OLD-BANKRUPTCY-ACT-GOOD-FAITH-DEFINITION-LEGAL-THEORY])
NEARLY-APPLICABLE-LEGAL-THEORIES: ([ESTUS-LEGAL-THEORY]
[FLYGARE-LEGAL-THEORY])
FACTUAL-PROTOTYPE-STORY-CATEGORY: NIL
FAMILY-RESEMBLANCE-PROTOTYPE: NIL

Figure 21. Intermediate argument for the Estus-problem case after eight nodes have been closed. Nodes that have been added since the end of the fifth cycle are shown in bold face.

as 0. This means, for instance, that the [DEANS] case, which is still on the open list – it was added in Cycle 6 – and rated 7 up until now, goes down in value from 7 to 6.

Processing continues in this vein with BankXX further filling out the argument pieces. Note that the [ALI] case has lead to a number of useful nodes: [RIMGALE], harvested as both a contrary and leading case, [GOEB], also harvested as both a contrary and leading case, and [DEANS], another contrary and leading case, which has not yet been harvested. The notion that some cases (such as *Ali*) are very rich in citations that turn out to be useful is a familiar experience in doing legal research.

Cycles 10, 11, 12

One piece that is lacking is a characterization of the case as a factual prototype. In the next cycle, [STUDENT-LOAN-FACTUAL-PROTOYPE], placed on the open list in cycle 1, becomes a maximal node (along with [DEANS] now devalued at 6). Being older, it is closed next. It is harvested by the *factual-prototype-story* argument piece. The neighbor method *factual-prototype-case-neighbors* yields the [GUNN] and [GIBSON] cases. Both are *supporting* and *best-supporting* cases and are placed on the open list with the same high rating (9). They are harvested in cycles 11 and 12.

Cycles 13, 14, 15

After harvesting [GUNN] and [GIBSON], BankXX closes the still highly-scored (6) [DEANS] case. In contrast to [RIMGALE] and [GOEB] however, [DEANS]

can be harvested only by the *leading-case* argument piece even though it is also a contrary case. This occurs because [ALI], [RIMGALE] and [GOEB] were previously harvested and the *contrary-cases* argument piece is already full.

The analogous situation obtains with [IACOVONI], which is closed in the cycle 14. It cannot be harvested by the *supporting-cases* argument piece – because this piece's fill limit of 3 cases has already been reached with [AKIN], [GUNN] and [GIBSON] – but it can be harvested by the *leading-cases* argument piece, which is not full since it has a limit of 5 and contains thus far only 4 leading-cited cases ([RIMGALE], [GOEB], [DEANS], [IACOVONI]). See Figure 23.

BankXX closes and harvests one more case – [OWENS] a *best-contrary-case* rated 5 – in cycle 15.

Cycles 16–20

BankXX considers the factor analyses of some of the cases that had been previously opened. For the next 5 cycles, it closes domain-factor-analysis nodes. (See items 16 through 20 in Figure 22.) Factor analyses for cases that share a large percentage of the factors with the problem situation, but whose cases are not most on-point cases, are collected in the *overlapping-cases* argument piece. This completely fills this argument piece, which has a fill limit of 5.

Cycles 21–29

In cycle 21, the system starts closing a set of low-scored cases ([MAKARCHUK] through [SANDERS]) on the open list. See Figure 22. These cases are all randomly chosen from the open list, which at this point contains only items scored as 0's. All these cases have evaluation function scores of 0 due to the fact that none is anything but an ordinary supporting or contrary case and the *supporting-cases* and *contrary-cases* argument pieces are already full. Consequently, none is harvested for the emerging argument. Indeed, none of these cases were actually mentioned in the actual *Estus* opinion, so nothing is actually missed by BankXX by not harvesting them (see discussion below).

Cycle 30

The last node closed in this example is the theory node [ABUSE-OF-CHAPTER-13-LEGAL-THEORY], which is pointed to by [SANDERS]. This theory turns out to be nearly applicable to the current problem and is harvested by the *nearly-applicable-legal-theories* argument piece. Since this argument piece is not full, this item is scored quite high (6). This theory – discovered at the last possible phase of the search – is an excellent addition to BankXX's store of information for the Estus-problem case since it is one of the five theories explicitly applied in the actual *Estus* opinion. Note, that it was discovered by a neighbor method applied to a minimally valued item.

1. [AKIN], 9
2. [ESTUS-LEGAL-THEORY], 6
3. [KITCHENS-KULL-LEGAL-THEORY], 8
4. [PER-SE-MINIMUM-PAYMENT-REQUIREMENT-LEGAL-THEORY], 8
5. [FLYGARE-LEGAL-THEORY], 6
6. [ALI], 6
7. [RIMGALE], 7
8. [OLD-BANKRUPTCY-ACT-GOOD-FAITH-DEFINITION-LEGAL-THEORY], 8
9. [GOEB], 7
10. [STUDENT-LOAN-FACTUAL-PROTOTYPE], 6
11. [GUNN], 9
12. [GIBSON], 9
13. [DEANS], 6
14. [IACOVONI], 6
15. [OWENS], 5
16. [AKIN-FACTOR-ANALYSIS], 1
17. [ALI-FACTOR-ANALYSIS], 1
18. [GUNN-FACTOR-ANALYSIS], 1
19. [GIBSON-FACTOR-ANALYSIS], 1
20. [OWENS-FACTOR-ANALYSIS], 1
21. [MAKARCHUK], 0
22. [CRUZ], 0
23. [BAEZ], 0
24. [ASHTON], 0
25. [BURRELL], 0
26. [RASMUSSEN], 0
27. [CHURA], 0
28. [FLYGARE], 0
29. [SANDERS], 0
30. [ABUSE-OF-CHAPTER-13-LEGAL-THEORY], 6

Figure 22. Order of case-domain-graph nodes closed when BankXX is run on the *Estus* case as a problem case with the argument-piece evaluation function. The numbers given are the evaluation function values of nodes at the time they are closed.

5.4. Results of the Processing

The state of the argument at the end of the processing is given in Figure 23.

BankXX has closed 30 nodes (the pre-set limit on this run), but it has opened a much larger number: 121. Of the 30 nodes closed, only 21 have actually been harvested by the argument pieces due to fill limits. While 121 is almost half of the nodes in the case-domain graph, it is a testament to the heuristic evaluation function that BankXX managed to focus its processing on a few well-chosen nodes that provided a great deal of useful information. Because of the highly interconnected nature of the case-domain graph and the fact that the neighbor methods can generate

SUPPORTING-CASES: [AKIN], [GUNN], [GIBSON]
BEST-SUPPORTING-CASES: [AKIN], [GUNN], [GIBSON]
CONTRARY-CASES: [ALI], [RIMGALE], [GOEB]
BEST-CONTRARY-CASES: [ALI], [OWENS]
LEADING-CASES: [RIMGALE], [GOEB], [DEANS], [IACOVONI],
SUPPORTING-CITATIONS: NIL
FACTOR-ANALYSIS: [ESTUS-PROBLEM-CASE-FACTOR-ANALYSIS]
OVERLAPPING-CASES: [AKIN-FACTOR-ANALYSIS],
[ALI-FACTOR-ANALYSIS], [GUNN-FACTOR-ANALYSIS],
[GIBSON-FACTOR-ANALYSIS], [OWENS-FACTOR-ANALYSIS]
APPLICABLE-LEGAL-THEORIES: [KITCHENS-KULL-LEGAL-THEORY],
[PER-SE-MINIMUM-PAYMENT-REQUIREMENT-LEGAL-THEORY],
[OLD-BANKRUPTCY-ACT-GOOD-FAITH-DEFINITION-LEGAL-THEORY]
NEARLY-APPLICABLE-LEGAL-THEORIES: [ESTUS-LEGAL-THEORY],
[FLYGARE-LEGAL-THEORY],
[ABUSE-OF-CHAPTER-13-LEGAL-THEORY]
FACTUAL-PROTOTYPE-STORY-CATEGORY:
[STUDENT-LOAN-FACTUAL-PROTOTYPE]
FAMILY-RESEMBLANCE-PROTOTYPE: NIL

Figure 23. The argument template at the conclusion of processing on the Estus-problem case.

many neighbors, it is typical for the system to open a large number of nodes and close far fewer. It is also typical for BankXX with the argument-piece evaluation function to harvest even fewer items than the number of nodes closed due to fill limits on argument pieces.

5.5. Comparison with the *Estus* Case - Discussion

In order to determine how well BankXX has performed, we compare BankXX output with the cases and legal theories identified in the actual *Estus* opinion (*In re Estus,* 695 F.2d 311 (8th Cir. 1982). See Appendix E. Since certain distinctions are hard to make when encoding an actual court opinion, we aggregated:

1. APPLICABLE-LEGAL-THEORIES and NEARLY-APPLICABLE-LEGAL-THEORIES,
2. BEST-SUPPORTING-CASES and ordinary SUPPORTING-CASES, and
3. BEST-CONTRARY-CASES and ordinary CONTRARY-CASES.

Also for this comparison we omitted from the output any cases decided or theories promulgated after *Estus*, a 1982 case,* since these are irrelevant to the actual *Estus* case.** See Figures 24 and 25. Note that ordinarily dates are ignored

* We did not eliminate items decided in the same year as *Estus* even though a few, no doubt, were actually decided after the *Estus* case was announced. Many theories existed prior to 1980, the date of our earliest case, and are never deleted.

** This approach to dealing with post-dated cases for comparison purposes is not the best since it works against BankXX. This is particularly true for BankXX run with the argument-piece and argument-factor evaluation functions. It is not so much of a burden in the node-type evaluation

AGGREGATED-THEORIES:
ABUSE-OF-CHAPTER-13-LEGAL-THEORY (before 1980)
PER-SE-MINIMUM-PAYMENT-REQUIREMENT-LEGAL-THEORY (before 1980)
ESTUS-LEGAL-THEORY (1982)
KITCHENS-KULL-LEGAL-THEORY (1981)
OLD-BANKRUPTCY-ACT-GOOD-FAITH-DEFINITION-LEGAL-THEORY (before 1980)
FLYGARE-LEGAL-THEORY (1983)

LEADING-CITED-CASES:
IACOVONI (1980), DEANS (1982), GOEB (1982), RIMGALE (1982)

AGGREGATED-SUPPORTING-CASES:
GIBSON (1985), GUNN (1984), AKIN (1984)

AGGREGATED-CONTRARY-CASES:
OWENS (1988), ALI (1983), GOEB (1982), RIMGALE (1982)

FACTUAL-PROTOTYPE-STORY:
STUDENT-LOAN-FACTUAL-PROTOTYPE

Figure 24. Aggregated argument pieces at conclusion of processing on the Estus-problem case. Dates of cases and theories are shown.

in the current version of BankXX since our assumption is that a case put to BankXX will be a new problem case, and thus anything already known to BankXX (i.e., present in the case-domain graph) is potentially relevant and is fair game for consideration by BankXX.

A partial comparison of BankXX output with the hand-coded version of the *Estus* opinion is given in Figure 26. *Overlap* means an item was found by BankXX and the opinion. *Missed* means it was present in the opinion but not harvested by BankXX. *Additional* means it was not present in the opinion but was harvested by BankXX.

The cases and theories that BankXX has culled from the case-domain graph show considerable overlap with those present in the actual *Estus* opinion. (The full encoding of the *Estus* case is given in Appendix E.) BankXX finds all of the cases in the actual *Estus* opinion that are considered leading cited cases: *Deans*,

function. In particular with the argument-piece evaluation function, there are built-in limits on how many items can be harvested for a given argument piece: there is a limit of 3 *supporting cases,* 5 *best suppoorting cases,* 3 *contrary cases*, 3 *best supporting cases*, etc. Post-processing filtering for dates hurts BankXX in two ways: (1) it allows BankXX to use valuable resources to chase down "irrelevant" (i.e., post-dated) cases, and (2) it allows BankXX to fill up on such "irrelevant" cases. A better approach is to check for dates at the time a node is expanded or opened; this is the approach taken in the comprehensive BankXX experiments reported in a companion paper [Rissland et al., 1995].

AGGREGATED-THEORIES:
ABUSE-OF-CHAPTER-13-LEGAL-THEORY
PER-SE-MINIMUM-PAYMENT-REQUIREMENT-LEGAL-THEORY
ESTUS-LEGAL-THEORY
KITCHENS-KULL-LEGAL-THEORY
OLD-BANKRUPTCY-ACT-GOOD-FAITH-DEFINITION-LEGAL-THEORY

LEADING-CITED-CASES:
IACOVONI, DEANS, GOEB, RIMGALE

AGGREGATED-SUPPORTING-CASES:

AGGREGATED-CONTRARY-CASES:
GOEB, RIMGALE

FACTUAL-PROTOTYPE-STORY:
STUDENT-LOAN-FACTUAL-PROTOTYPE

Figure 25. Argument piece items for the Estus-problem case that remain after date-filtering. Cases decided or theories promulgated after 1982 – the date for *Estus* – were deleted after BankXX completed its run.

AGGREGATED-THEORIES:		
OVERLAP	*MISSED*	*ADDITIONAL*
ABUSE-OF-CHAPTER-13	ALL-THE-FACTS-AND-CIRCUMSTANCES	KITCHENS-KULL-THEORY
PER-SE-MINIMUM-PAYMENT-REQUIREMENT	SUBSTANTIAL-OR-MEANINGFUL-REPAYMENT	OLD-BANKRUPTCY-ACT-GOOD-FAITH-DEF'N
ESTUS-LEGAL-THEORY	BEST-INTERESTS-OF-CREDITORS-TEST	
LEADING-CITED-CASES:		
OVERLAP	*MISSED*	*ADDITIONAL*
RIMGALE		
GOEB		
DEANS		
IACOVONI		
AGGREGATED-SUPPORTING-CASES:		
OVERLAP	*MISSED*	*ADDITIONAL*
HEARD	IACOVONI*	
	KULL	
	TERRY	
AGGREGATED-CONTRARY-CASES:		
OVERLAP	*MISSED*	*ADDITIONAL*
RIMGALE	DEANS*	
GOEB	BARNES	
	BELLGRAPH	

Figure 26. Partial comparison of the BankXX-generated argument with the hand-coded version of the *Estus* opinion, after argument pieces have been aggregated and post-dated cases removed. *Note that *Iacovoni* and *Deans* are listed as overlap LEADING-CITED-CASES.

Goeb, *Rimgale*, and *Iacovoni*. *Deans*, in particular, was deemed "persuasive" by the court (opinion at 316). BankXX has identified the student loan story as the prototype story for Estus-problem case. It has output none of the four same-side cases but two of the five contrary cases mentioned in the opinion. Note however that BankXX actually listed the *Iacovoni*, a same-side case, and *Deans*, a contrary case, under leading cases.* The cases that BankXX missed are pretty far down in the claim lattice (Figure 15), even when only those cases decided before or in 1982 are considered.

The program has also done quite well in identifying the ABUSE-OF-CHAPTER-13, and the PER-SE-MINIMUM-PAYMENT-REQUIREMENT theories as nearly applicable or applicable to the problem fact situation. Both of these theories were mentioned by the court. It also identified the ESTUS-THEORY, the theory promulgated by this leading case, as nearly applicable to the fact situation. (We noted above that there is not enough information in the Estus-problem case for BankXX to determine whether all the factors defining the ESTUS-THEORY are applicable, and so it is harvested only as *nearly-applicable*.) BankXX has also identified some other applicable theories that are not specifically mentioned by the court: the definition of "good faith" under the old bankruptcy statute (The Bankruptcy Act of 1898, ch. 541, 30 Stat. 544, as amended), and the KITCHENS-KULL THEORY, promulgated in *Kull* (1981) and reiterated in *Kitchens* (1983). It is also interesting to note that BankXX identified the FLYGARE-THEORY, which doesn't show up in the comparison because of its 1983 date; this theory is really the same as Estus even though it was treated by the Flygare court as its own.

Similar comparisons between the hand-coded *Estus* opinion and aggregated, date-filtered output from BankXX configured with the node-type and argument-factor evaluation functions can also be made. (See Appendix F). There is considerable similarity with what was harvested under the argument-piece evaluation function. However, *Iacovoni* and *Deans* are duly listed respectively in AGGREGATED-SUPPORTING-CASES and AGGREGATED-CONTRARY-CASES and not just in LEADING-CASES; this occurs because BankXX with these other evaluation functions is not restricted by limits on the number of cases that can fill the various argument pieces. In addition, *Heard* and *Barnes* – both missed with the argument piece evaluation function – are harvested in these other runs. *Kull*, *Terry* and *Bellgraph* are still missed.

---

* *Iacovoni* is not listed under the AGGREGATED-SUPPORTING-CASES because the relevant same-side argument pieces (SUPPORTING-CASES) were already filled up when *Iacovoni* was considered (i.e., closed) by BankXX and *Iacovoni* is not a *best* case for the *Estus* problem situation so it was not listed there even though the limit of 5 had not been reached for BEST-SUPPORTING-CASES. This is an example of how not checking dates during processing can hurt BankXX with regard to comparion purposes: other (post-dated) cases – *Gibson* (1985), *Gunn* (1984), *Akin* (1984) – have already filled up the relevant argument piece and blocked BankXX from including *Iacovoni*. The same is true for *Deans*. Before *Deans* is considered, *Goeb*, *Rimgale* and *Ali* filled the CONTRARY-CASE argument piece, which has a limit of 3 cases. *Deans* is also not a *best* contrary case. See Figure 21.

Regardless of the evaluation function used, BankXX does not come close to discovering *Bellgraph* since it is never opened. *Bellgraph* is a case that is very meagerly linked with the rest of the case-domain graph and is never opened through expansion by the neighbor methods. On the other hand, *Kull* and *Terry*, the other two supporting cases that are missed, are opened in all three versions of BankXX. However *Kull* and *Terry* are not leading or best cases and thus are not scored very highly in by the argument-piece evaluation function and in fact, after the *supporting-cases* argument piece is filled (in cycle 12), their value goes to 0. With the node-type evaluation function, *Kull* and *Terry* lose out by the luck of the draw – they are scored 6 along with a host of other cases and simply miss out in tie-breaking. With the argument-factor evaluation function, neither gets a very high score since neither contributes much to the evolving argument as measured by the argument factors.

The same set of theories is retrieved by BankXX with the node-type evaluation function as with the argument-piece evaluation function. BankXX with argument-factor evaluation function retrieves about half as many overall. It misses the important ABUSE-OF-CHAPTER-13-LEGAL-THEORY, as well the same three theories missed by the others. It ends up with only two theories, the PER-SE-MINIMUM theory and the ESTUS-THEORY itself, in common with the *Estus* opinion.

BankXX run with the node-type evaluation function also retrieves the student-loan prototype. This is missed with the argument-factor evaluation function; this is not too much of a surprise since such information is not highly valued with this evaluation function. In general, what is retrieved by each of the evaluation functions reflects their biases, that is, the terms and weights that they use.

In general, BankXX run with the node-type evaluation function harvests (i.e., adds to some argument piece) more cases than BankXX run with the argument-piece evaluation function* although they both close roughly the same number of cases.** This is because there are no limits placed on the number of cases that can be harvested for the individual argument pieces when the node-type or argument-factor evaluation functions are used. Limits on the argument pieces when the argument-piece evaluation function is used can be quite restrictive (see Figure 12.) and can have a very significant impact on what is output by BankXX.‡ Thus, BankXX run with the argument-piece evaluation function isn't just simply BankXX with a

* For the Estus-problem case, BankXX harvests: 3 AGGREGATED-SAME-SIDE-CASES with the argument-piece evaluation function, 8 with the node-type evaluation function, and 7 with the argument-factor evalaution function. It retrieves 4 AGGREGATED-CONTRARY CASES with the argument-piece evaluation function, 10 with the node-type evaluation function, and 8 with the argument-factor evaluation function. See Figure 24 and Appendix H.

** 18 cases are closed with the node-type, 17 with the argument-piece, and 15 with the argument-factor evaluation functions.

‡ These limits can greatly impact BankXX's performance under certain circumstances. Notably if the opinion mentions a large number of cases, BankXX will, by definition, miss some. Also, if BankXX harvests post-dated cases (i.e., cases occurring after the date of the problem case), these will not show up in the BankXX-hand-coded comparisons since they are deleted because they are simply not relevant to the problem case. However, they have used up a certain amount of BankXX's

different evaluation function but also with bounds placed on what can be harvested by the argument pieces.

Note that these differences in cases harvested are not immediately apparent when considering the output *after* it has been date filtered since so many of the harvested cases are post-1982 and thus are deleted. (See Appendix F.) Date-filtering makes BankXX run with the node-type evaluation function appear more conservative and precise than it actually is. BankXX run with the argument-factor evaluation function often harvests fewer cases than BankXX run with either of the other two evaluation functions since the use of this evaluation function is expensive computationally.*

It is important to note that even though BankXX's output under the different evaluation functions is similar, it does not behave similarly in its search. The order of exploration of the case-domain graph under the various evaluation functions is quite different,** as are the values assigned to individual nodes (i.e., the search graph is different). Thus, BankXX behaves quite differently with the different evaluation functions. In a larger case-domain graph, the differences would become more apparent. There appears to be a classic knowledge-performance trade-off occurring with BankXX run under the various evaluation functions. This is especially evident when non-date-filtered output is examined. BankXX with the node-type evaluation function harvests more cases, including more cases not listed in the opinion and fewer MISSED cases, than the BankXX under the other two – especially the argument-piece evaluation function – which have more MISSED but fewer ADDITIONAL cases.‡ This trade-off is persistent. It shows up throughout the extensive set of experiments we have performed on BankXX [Rissland et al., 1995].

As a rough summary, one can say that BankXX with the node-type evaluation function is somewhat "dumber" – not particularly selective nor sensitive to problem context – whereas BankXX with the argument-piece and argument-factor evaluation functions is "smarter" – more selective and more problem-sensitive. These generalizations are examined in detail in a companion paper [Rissland et al., 1995].

---

case limits and possibly prevented relevant, non-post-dated cases from being harvested. A fuller discussion of the problem of evaluation can be found in a companion article [Rissland et al., 1995].

* Although there are some problem cases, like *Estus*, where BankXX run with the argument-piece evaluation function is very similar to it run with the node-type evaluation function.

** Sometimes there are some "chunks" of the case-domain graph that are opened in the same order. This is due to neighbor methods which perform exactly the same under all three evaluation functions.

‡ This is also true if one calculates traditional precision and recall scores for just this one example case using the date-filtered output compared against the hand-encoded opinion. This sort of quantitative analysis is pursued in a companion article [Rissland et al., 1995].

Part IV: Related Work and Conclusions

6. Related Research

We have not discussed generally here either argument or legal argument, which are treated well and at length elsewhere (e.g., [Perelman & Olbrechts-Tyteca, 1969], [Toulmin, 1958], [Levi, 1949]), or argument modeled through other means than search ([McCarty & Sridharan, 1982], [Sycara, 1989], [Alvarado, 1990]). In addition, our present goal is not to provide a formal logical model of legal argument. We refer the reader to [Prakken, 1993], [Gordon, 1991] and [Loui *et al.*, 1993] for excellent discussions. Recent work by Prakken, Loui and others brings to light some of the connections between a formal analysis of argument and the arguments created by HYPO [Ashley, 1990] and CABARET [Rissland & Skalak, 1991].

Several researchers have addressed aspects of argument as search. [Bhatnagar, 1989; Bhatnagar & Kanal, 1991] treat an argument as a search for a causal model that supports a given proposition. Bhatnagar uses a variant of A* search to create models that satisfy argument goals, in which it assumed that probability values may be computed for the validity of supported propositions given a particular model. While we also view argument creation as theory construction [Rissland & Skalak, 1991], we believe that such a probabilistic approach may be difficult to apply in a domain as "weak" as law.

Branting's GREBE system [Branting, 1991] uses structured representations of the explanations for legal decisions supplied in the opinions of legal cases. The use of factors in BankXX's legal theories is similar to the use of precedent constituents in GREBE. Also, GREBE uses heuristic A* search for one aspect of argument creation: retrieval of a precedent that best explains a problem case. Best-first search is performed in a space consisting of all mappings from a problem case to these structured representations of precedent cases. Thus GREBE's use of A* search is not in the same search space as that of BankXX, but search is used to the same end – to retrieve relevant cases.

While we do not rely on research using artificial neural networks, or on related massively parallel techniques, for information retrieval, the flavor of some of this work is similar to ours. In particular, Rose's SCALIR [Rose 1994; Rose & Belew, 1991] is a hybrid symbolic and sub-symbolic system that uses a network of legal knowledge, including *Shepard's* links and West's key number taxonomy links, through which numerical activation is spread to perform retrieval.

BankXX's approach to legal retrieval can be contrasted with SCALIR's approach. First, while BankXX relies on best-first heuristic search directed by any of three evaluation functions at different levels of abstraction, SCALIR uses spreading activation to perform the retrieval. Thus the search control strategies of the two programs are distinct. Second, approximately 90% of the links in the SCALIR network are weighted connectionist links, with 75% of all the links between cases and terms. BankXX does not incorporate such standard information retrieval term-document indices, but relies solely on symbolic links between data. SCALIR

has no explicit representation for a legal theory or a factual prototype, which are important parts of BankXX's representation. Thus the two semantic networks are quite different. Third, the tasks performed by the programs are different. SCALIR is a generic legal information retrieval program, whereas BankXX's retrieval is directed and informed by the requirements of a particular task: creating an argument. Thus, while SCALIR provides a progressive model for legal retrieval, it is quite different from BankXX in its search control strategy, representation scheme, and task application.

This research also shares certain knowledge representation approaches with earlier work in legal information retrieval (e.g., [Hafner, 1981; Bing, 1987; Dick, 1987]). Such projects in conceptual legal retrieval relied on graphs of diverse legal entities and concepts, where labeled links captured influences and taxonomic information. A more recent project in legal information retrieval is Gelbart and Smith's FLEXICON [1991], which uses a vector space model for retrieval. FLEXICON can perform automatic thesaurus construction and relevance feedback, and can extract important paragraphs of an opinion to generate headnotes automatically. Our BankXX work also shares certain conclusions on the utility of providing multiple paths to information to aid retrieval demonstrated by earlier work in case-based reasoning (e.g., [Kolodner, 1983]).

Some ideas used in this paper – "in-space" and "cross-space" neighbor methods that make use of graph linkages, interconnected "spaces" of nodes, strongly linked cases and theories, use of indirect "dual space" methods, etc. – echo some of the work first presented on structured representations for (mathematical) knowledge [Rissland, 1977]. For instance, in Rissland's work, there were methods – akin to BankXX boomerang methods – for indexing and retrieving relevant, but not directly or closely linked, items in a given space by visiting nodes in another "dual" space (e.g., the method to "find all the examples that apply the theory used in this example"). As in BankXX, there were also methods that simply followed in-space pointers (e.g., "find all the examples that a particular example references or builds upon"). In fact, the use of examples in that body of work presaged many aspects of current case-based reasoning. However, the structure and methods used by Rissland were much less dynamic than those in BankXX. Most were simply pointer-chasing methods, as opposed to those in BankXX, like the dynamic neighbor methods, which generate new linkages. In addition, all the indexing in Rissland's work used a static indexing scheme; there was no sense of context-sensitive indexing through dimensions or dynamic neighbor methods. While there were heuristics, there was no real sense of heuristic search, replete with evaluation functions, start nodes, etc.

BankXX's notions of a leading case and of prototypical stories bring into play the idea of prototypes and their role in indexing and organizing knowledge. The notion of a prototype is important in a number of realms. Lakoff has gone so far as to claim "Prototypes do a great deal of the real work of the mind" [1987, p. 145]. In the law, McCarty and Sridharan proposed a representation for cases consisting of legal prototypes plus possible deformations of them [1982]. Rissland

suggested a similar "retrieval-plus-modifications" approach for generating counter-examples in mathematics [Rissland, 1980]. Much of the trail-blazing research in cognitive science on prototypes and their relation to category structure has been done by Eleanor Rosch and colleagues. Rosch conducted a series of experiments documenting "prototype effects," which are asymmetries in goodness-of-example ratings [Rosch & Mervis, 1975].

In machine learning, the role of prototypes in classification tasks is a long-standing subject of research. To make a class prediction, "instance-based" or "example-based" classification algorithms, such as the k-nearest neighbor algorithm, compute the similarity between an instance to be classified and the instances in the data set. The pattern recognition research community has developed a variety of algorithms to accelerate this computation by editing the data set to retain only a set of distinguished "prototypical" examples [Dasarathy, 1991]. More recently, researchers in machine learning have shown that on some data sets, identifying prototypes can increase the accuracy of instance-based classifiers as well as dramatically reduce their computational cost (e.g., [Skalak, 1994]).

In information retrieval, prototypical "document representatives" have been used to summarize the terms appearing in the documents in a document cluster. The use of document representatives facilitates the efficient computation of the similarity of a query to a cluster of documents. At the simplest, a mean document centroid is created [Salton, 1989]. More complex approaches have been developed by Croft [1979] and Voorhees [1985].

For central ideas about case-based reasoning, such as analyzing cases in terms of important domain factors or "dimensions," the construction of "claim lattices," which partially order retrieved cases by dimensions, and the selection of best and most on-point cases, we rely on ideas and methods developed in the CBR Laboratory at the University of Massachusetts over the last ten or so years, particularly in the HYPO system and its progeny [Rissland, et al., 1984; Ashley, 1990; Rissland & Skalak, 1991].

7. Conclusions

The goal of this project was to unite a number of areas that bear directly on the process of gathering information for use in legal arguments. The BankXX framework brings together research in information retrieval, control of intelligent computer programs, heuristic search, case-based reasoning, legal research, legal knowledge representation, and, of course, legal argument. In a companion article we discuss a series of experiments designed to measure how well BankXX performs [Rissland et al., 1995].

This article has described several new approaches and mechanisms including:

- Representation of *legal theories* in terms of domain factors.
- *Neighbor methods* for traversing the case-domain graph, a semantic network of case and other legal knowledge from a particular domain.

- Three evaluation functions – *node-type, argument-piece, and argument-factor evaluation functions* – to guide search of the case-domain graph, each capturing a different perspective on legal knowledge and argument and incorporating a set of terms to be used in the evaluation.
- *Argument pieces* for representing generic information needed for argument.
- *Argument factors* for evaluating the quality of an argument.

The incorporation of legal theories explicitly into the knowledge representation distinguishes BankXX from previous projects from our group, such as CABARET and HYPO. It also distinguishes BankXX from most other programs in the law. The use of argument pieces and argument factors is also unique to this project.

From the standpoint of case-based reasoning, BankXX has been a testbed for investigating the utility for legal retrieval of applying search in addition to knowledge-based indexing. While HYPO and CABARET retrieved cases indexed by factors and by factors and rules, respectively, BankXX performs retrieval using state-space search in an indexed network of legal knowledge. This application of search effects a data-driven control algorithm for argument creation, with some top-down constraints provided by the need to fill in the argument pieces. The data-driven approach can be contrasted with a top-down control scheme driven by stereotypical argument forms [Skalak & Rissland, 1992].

In addition to describing the computational mechanisms and knowledge representation used in BankXX, we ran through an extended example of a BankXX run on the *Estus* case, a landmark case addressing the "good faith" requirement that is central for approval of Chapter 13 plans in personal bankruptcy. In addition to illustrating BankXX's overall control flow based on heuristic search, the example presented certain of the computational details, such as the use of neighbor methods, evaluation of opened nodes, and the incremental building up of an argument through the argument pieces as BankXX's problem-solving proceeds.

Even on this single example, certain general themes about BankXX were evident:

- BankXX embodies **strong preferences** for certain kinds of information such as legal theories and leading cases (reified in its evaluation functions).
- BankXX **selectively harvests** useful information from the great wealth of information available for consideration.
- BankXX neighbor methods **greatly expand** the possible leads to examine in the course of problem-solving.
- BankXX's heuristic evaluation (and the fill limits in place when BankXX is configured with the argument-piece evaluation) **greatly limits** the information actually harvested.

The theme of *For many are called, but few are chosen* (Matt 22:14) is particularly evident in the way information is harvested by BankXX configured with the argument-piece evaluation function.

These themes reflect our own personal experience, as well as those of others [Kunz, et al. 1992], in performing legal research in a vast library of traditional

legal materials where one is constantly making choices about which information to examine in depth, which leads to follow, which cases and theories to incorporate into one's evolving informational harvest, etc., and all in the context of the need to make an argument, write a brief or memo, possibly on short notice and for a demanding audience. The possibility – nearly reality – of performing such research tasks electronically, possibly with the aid of "infobots" and intelligent network gophers, gives further practical significance to our own work with BankXX.

Our *Estus* example also illustrates certain qualitative trade-offs between the various configurations of BankXX. Such qualitative and quantitative observations are discussed in the detailed analyses described in a companion paper [Rissland et al., 1995].

BankXX configured with the argument-piece evaluation function – and its accompanying fill limits on argument pieces – is much more selective and problem sensitive than BankXX configured with the node-type evaluation function. In fact, without date-filtering BankXX with the node-type evaluation function is not particularly problem-sensitive, at least in terms of what information is ultimately harvested. Of course, the internal behaviors of the various configurations – particularly the order of nodes explored and harvested – do vary greatly. Such differences would no doubt be more apparent in a larger case-domain graph.

As with any exploratory computer program, there were design and implementation decisions that presumably have affected the performance of the program. For example, when the user specifies starting search with a most on-point case, one of the most on-point cases is selected at random, and search begins there. The other most on-point cases are discarded in the current implementation, but it would have been simple and probably preferable to have placed them on the open list as well at the beginning of the search. There is great potential also for identifying prototypical cases, using a family resemblance calculation or some other measure, that was not explored in the current implementation. In retrospect, we would also have increased the limits on the number of items that could fill an argument piece under the argument-piece evaluation. In particular, we would have raised the limits on ordinary *supporting-cases* and *contrary-cases*.

We also learned quite a bit about the impact of dates and date-filtering on performance evaluation. In fact, the issue of dates, while not crucial for the intended use of BankXX as a problem-solving program for new cases, is critical in evaluation using already existing cases run as de novo problem cases. These observations led us to modify BankXX to filter for dates in the course of problem-solving before carrying out the massive set of experiments reported in a companion paper [Rissland et al., 1995]. This change has no impact on our original intentions for BankXX but it does make evaluation more fair.

Analogous considerations regarding court jurisdiction and pedigree probably also exist. However, in our application domain, where there is a paucity of appeals cases and where courts tend to look as far afield as they need to for useful precedents,

overlooking jurisdiction and pedigree was not a major stumbling block. In other domains, it might.

One area for future work is to learn the evaluation function to do the search (e.g., [Samuel, 1959, 1967; Minton, 1988]). In fact, our inclusion of an evaluation step (using the argument factors) at the conclusion of a problem-solving run was aimed at this goal. There is a variety of algorithms and architectures that could be applied to learn evaluation functions, such as the fixed-increment error correction rule [Nilsson, 1990], learning from preference predicates [Utgoff & Clouse, 1991], and various neural network algorithms. However, most rely on some form of scalar-valued error function to assess the quality of the current evaluation function weights. To apply a technique that relies on linearly ordered supervisory information to evaluate the quality of an argument requires that that quality be expressed as a scalar value. In BankXX, however, the quality of an argument depends on its placement along a variety of argument factors. Thus, at best, the supervisory information available from BankXX is partially ordered and not linearly ordered, unless one combines the argument factors into a scalar value, or finds a learning algorithm that relies on partially ordered fitness values.

Although many legal issues involve the interaction of cases and legal statutes, BankXX does not incorporate statutes or regulations into its domain knowledge. We examined the interaction of arguing with a rule and with cases in detail in the CABARET project [Rissland & Skalak, 1991]. As we noted early in this article, codified legal rules also provide indices into other types of legal knowledge. The addition of statutory rules to the case-domain graph would also enhance the opportunities for multiple indexing inherent in this domain.

References

Alvarado, S. J. (1990). *Understanding Editorial Text: A Computer Model of Argument Comprehension.* Boston, MA: Kluwer Academic Publishers.

Ashley, K. D. (1990). *Modeling Legal Argument: Reasoning with Cases and Hypotheticals.* Cambridge, MA: M.I.T. Press.

Ashley, K. D. & Aleven, V. (1991). A Computational Approach to Explaining Case-Based Concepts of Relevance in a Tutorial Context. *Proceedings Case-Based Reasoning Workshop 1991,* 257–268. Washington, D.C. Morgan Kaufmann, San Mateo, CA.

Ashley, K. D. & Rissland, E. L. (1987). But, See, Accord: Generating Blue Book Citations in HYPO. *Proceedings First International Conference on AI and Law (ICAIL-87),* 67–74. Boston, MA. Association for Computing Machinery.

Barr, A., Feigenbaum, E. A. & Cohen, P. (1981). *The Handbook of Artificial Intelligence.* Reading, MA: Addison-Wesley.

Berman, D. H. & Hafner, C. D. (1991). Incorporating Procedural Context into a Model of Case-Based Legal Reasoning. *Proceedings Third International Conference on Artificial Intelligence and Law (ICAIL-91),* 12–20. Oxford, England. Association for Computing Machinery.

Bhatnagar, R. K. (1989). *Construction of Preferred Causal Hypotheses for Reasoning with Uncertain Knowledge.* Ph.D. Dissertation, University of Maryland, College Park, MD.

Bhatnagar, R. & Kanal, L. N. (1991). A Formalism for Automated Generation of Preferred Arguments. *Working Notes, AAAI Spring Symposium Series, Argument and Belief,* 39–61. Palo Alto, CA.

Bing, J. (1987). Designing Text Retrieval Systems for "Conceptual Searching." *Proceedings First International Conference on Artificial Intelligence and Law (ICAIL-87)*, 43–51. Boston, MA. Association for Computing Machinery.

Bluebook. (1986). *A Uniform System of Citation* (14th ed.). Cambridge, MA: The Harvard Law Review Association.

Branting, L. K. (1991). Building Explanations from Rules and Structured Cases. *International Journal of Man-Machine Studies*, 34, 797–837.

Croft, W.B. (1979). *Organizing and Searching Large Files of Document Descriptions.* Ph.D. Dissertation, University of Cambridge, England.

Dasarathy, B. V. (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* Los Alamitos, CA: IEEE Computer Society Press.

Dick, J. P. (1987). Conceptual Retrieval and Case Law. *Proceedings First International Conference on AI and Law (ICAIL-87)*, 106–115. Boston, MA. Association for Computing Machinery.

Fikes R. E., Hart P., & Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence.* 3:251–288.

Gardner, A. vdl. (1987). *An Artificial Intelligence Approach to Legal Reasoning.* M.I.T. Press, Cambridge.

Gelbart, D. and Smith, J.C. (1991). Beyond Boolean Search: FLEXICON, A Legal Text-Based Intelligent System. *Third International Conference on Artificial Intelligence and Law (ICAIL-91)*, 225–234. Oxford, England. Association for Computing Machinery.

Gordon, T. F. (1991). An Abductive Theory of Legal Issues. *International Journal of Man-Machine Studies.* 35:95–118.

Hafner, C. D. (1981). *An Information Retrieval System Based on a Computer Model of Legal Knowledge.* Ph.D. thesis, University of Michigan, republished by UMI Research Press, Ann Arbor, MI.

Hafner, C. D. (1987). Conceptual Organization of Case Law Knowledge Bases. *Proceedings First International Conference on Artificial Intelligence and Law*, 35–42. Boston, MA. Association for Computing Machinery.

Hammond, K. J. (1989). *Case-Based Planning: Viewing Planning as a Memory Task.* Boston, MA: Academic Press.

Kolodner, J. L. (1983). Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7(4), 243–280.

Kunz, C. L., Schmedemann, D. A., Bateson, A. L., Downs, M. P. & Erlinder, C. P. (1992). *The Process of Legal Research* (Third Edition ed.). Boston, MA: Little, Brown.

Lakoff, G. (1987). *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind.* Chicago, IL: University of Chicago Press.

Lehnert, W. G. (1981). Plot units and Narrative Summarization. *Cognitive Science*, 5(4): 293–331.

Levi , E. H. (1949). *An Introduction to Legal Reasoning.* Chicago, IL: University of Chicago Press.

Loui, R. P., Norman, J., Olson, J., & Merrill, A. (1993). A Design for Reasoning with Policies, Precedents, and Rationales. *Proceedings Fourth International Conference on Artificial Intelligence and Law (ICAIL-93)*, 202–211. Amsterdam, The Netherlands. Association for Computing Machinery.

McCarty, L. T., & Sridharan, N. S. (1982). *A Computational Theory of Legal Argument.* Technical Report LRP-TR-13. Laboratory for Computer Science, Rutgers University.

Minton, S. (1988). *Learning Effective Search Control Knowledge: An Explanation-Based Approach.* Boston, MA: Kluwer Academic Publishers.

Nilsson, N. J. (1990). *The Mathematical Foundations of Learning Machines.* San Mateo, CA: Morgan Kaufmann.

Perelman, C. & Olbrechts-Tyteca, L. (1969). *The New Rhetoric: A Treatise on Argumentation.* Notre Dame, Indiana: University of Notre Dame Press.

Porter, B. W., Bareiss, R. & Holte, R. C. (1990). Concept Learning and Heuristic Classification in Weak-Theory Domains. *Artificial Intelligence*, 45, 229–263.

Prakken, H. (1993). *Logical Tools for Modelling Legal Argument.* Ph. D. Dissertation, Vrije Universiteit, Amsterdam.

Rissland, E. L. (1977). *Epistemology, Representation, Understanding and Interactive Exploration of Mathematical Theories.* Ph.D. Dissertation. MIT, 1977.

Rissland, E. L. (1978). Understanding Understanding Mathematics. *Cognitive Science*, 2, 361–383.

Rissland, E. L. (1980). Example Generation. *Proceedings Third National Conference of the Canadian Society for Computational Studies of Intelligence*, Victoria, BC.

Rissland, E. L., Daniels, J. J., Rubinstein, Z. B. & Skalak, D. B. (1993). Case-Based Diagnostic Analysis in a Blackboard Architecture. *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, 66–72. Washington, DC. AAAI Press/MIT Press.

Rissland, E. L. & Skalak, D. B. (1991). CABARET: Rule Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies*, 34, 839–887.

Rissland, E. L., Skalak, D. B. & Friedman, M. T. (1993a). Case Retrieval through Multiple Indexing and Heuristic Search. *Proceedings of Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, Chambery, Savoie, France. International Joint Conferences on Artificial Intelligence.

Rissland, E. L., Skalak, D. B. & Friedman, M. T. (1993b). BankXX: A Program to Generate Argument through Case-Based Search. *Proceedings Fourth International Conference on Artificial Intelligence and Law (ICAIL-93)*, 117–124, Amsterdam, The Netherlands. Association for Computing Machinery.

Rissland, E.L., Skalak, D.B. & Friedman, M.T. (1995). Evaluating a Legal Argument Program: The BankXX Experiments. To appear in *The Journal of Artificial Intelligence and Law.* Also available as Technical Report, CMPSCI TR 95–30, Department of Computer Science, University of Massachusetts, Amherst, MA.

Rissland, E. L., Valcarce, E. M. & Ashley, K. D. (1984). Explaining and Arguing with Examples. *Proceedings of the National Conference on Artificial Intelligence (AAAI-84)*, 288–294. Austin, TX. American Association for Artificial Intelligence.

Rosch, E. & Mervis, C. B. (1975). Family Resemblances: Studies in the Internal Structure of Categories. *Cognitive Psychology*, 7, 573–605.

Rose, D. E. & Belew, R. K. (1991). A Connectionist and Symbolic Hybrid for Improving Legal Research. *International Journal of Man-Machine Studies*, 35, 1–33.

Rose, D. E. (1994). *A Symbolic and Connectionist Approach to Legal Information Retrieval.* Lawrence Erlbaum, Hillsdale, N.J.

Salton, G. (1989). *Automatic Text Processing.* Reading, MA: Addison-Wesley.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM J. Research and Development*, 3:210–229.

Samuel, A. L. (1967). Some studies in machine learning using the game of checkers. II – Recent Progress. *IBM J. Research and Development*, 11:601–617.

Schank, R. C. (1982). *Dynamic Memory,* Cambridge: Cambridge University Press.

Schank, R. C. (1990). *Tell Me a Story: A New Look at Real and Artificial Memory.* New York, NY: Scribner.

Shepard's. (1994). *Shepard's Federal Citations.* Colorado Springs, CO: Shepard's/McGraw-Hill.

Skalak, D. B. (1994). Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms. *Proceedings of the Eleventh International Conference on Machine Learning.* New Brunswick, NJ. Morgan Kaufmann.

Skalak, D. B. & Rissland, E. L. (1992). Arguments and Cases: An Inevitable Intertwining. *Artificial Intelligence and Law: An International Journal*, 1(1), 3–48.

Sullivan, T. A., Warren, E. & Westbrook, J. L. (1989). *As We Forgive Our Debtors.* New York, NY: Oxford University Press.

Sycara, K. P. (1989). Argumentation: Planning Other Agents' Plans. *Proceedings Eleventh International Joint Conference on Artificial Intelligence (IJCAI-88),* 517–523. Detroit, MI.

Sycara, K. P. & Navinchandra, D. (1991). Influences: A Thematic Abstraction for Creative Use of Multiple Cases. *Proceedings Case-Based Reasoning Workshop 1991*, 133–144. Washington, DC. San Mateo, CA: Morgan Kaufmann.

Toulmin, S. (1958). *The Uses of Argument.* Cambridge University Press.

Twining, W. & Miers, D. (1991). *How To Do Things With Rules.* London: Weidenfeld and Nicolson.

Turner, R. (1988). Organizing and Using Schematic Knowledge for Medical Diagnosis. *Proceedings Case-Based Reasoning Workshop 1988*, 435–446. Clearwater Beach, FL. San Mateo, CA: Morgan Kaufmann.

Utgoff, P. E., & Clouse, J. A. (1991). Two kinds of training information for evaluation function learning. *Proceedings Ninth National Conference on Artificial Intelligence (AAAI-91)*, 596–600. Anaheim, CA. MIT Press.

Voorhees, E.M. (1985). *The Effectiveness and Efficiency of Agglomerative Clustering in Document Retrieval.* Ph.D. Dissertation, Cornell University, Ithaca, NY.

West (1992). WestLaw. St. Paul, MN: West Publishing Co.

Words and Phrases (1994). *Words and Phrases*. St. Paul, MN: West Publishing.

Appendix A. The Cases in the BankXX Corpus

ADAMU	*In re Adamu*, 82 B.R. 128 (Bkrtcy. D.Or. 1988)
AKIN	*Matter of Akin*, 54 B.R. 700 (Bkrtcy. 1985)
ALI	*In re Ali*, 33 B.R. 890 (Bkrtcy. 1983)
ASHTON	*In re Ashton*, 85 B.R. 766 (Bkrtcy. S.D.Ohio 1988)
BAEZ	*In re Baez*, 106 B.R. 16 (Bkrtcy. D.Puerto Rico 1989)
BARNES	*Barnes v. Whelan*, 689 F.2d 193 (1982)
BELLGRAPH	*Matter of Bellgraph*, 4 B.R. 421 (1980)
BOYD	*In re Boyd*, 57 B.R. 410 (Bkrtcy. N.D.Ill. 1983)
BROWN	*In re Brown*, 56 B.R. 293 (Bkrtcy. N.D.Ill. 1985)
BURRELL	*In re Burrell*, 2 B.R. 650 (1980)
CALDWELL	*In re Caldwell*, 895 F.2d 1123 (6th Cir. 1990)
CANDA	*In re Canda*, 33 B.R. 75 (Bkrtcy. 1983)
CHURA	*In re Chura*, 33 B.R. 558 (Bkrtcy. 1983)
CRUZ	*Matter of Cruz*, 75 B.R. 56 (Bkrtcy. D.Puerto Rico 1987)
DEANS	*Deans v. O'Donnell*, 692 F.2d 968 (1982)
DOS-PASSOS	*In re Dos Passos*, 45 B.R. 240 (Bkrtcy. 1984)
EASLEY	*In re Easley*, 72 B.R. 948 (Bkrtcy. M.D.Tenn. 1987)
EPPERS	*In re Eppers*, 38 B.R. 301 (Bkrtcy. 1984)
ESTUS	*In re Estus*, 695 F.2d 311 (1982)
FLYGARE	*Flygare v. Boulden*, 709 F.2d 1344 (1983)
GIBSON	*In re Gibson*, 45 B.R. 783 (Bkrtcy. 1985)
GIRDAUKAS	*In re Girdaukas*, 92 B.R. 373 (Bkrtcy. E.D.Wis. 1988)
GOEB	*In re Goeb*, 675 F.2d 1386 (1982)
GUNN	*In re Gunn*, 37 B.R. 432 (Bkrtcy. 1984)
HAWKINS	*Matter of Hawkins*, 33 B.R. 908 (Bkrtcy. 1983)
HEARD	*In re Heard*, 6 B.R. 876 (1980)
IACOVONI	*In re Iacovoni*, 2 B.R. 256 (1980)
KITCHENS	*In re Kitchens*, 702 F.2d 885 (1983)
KULL	*Matter of Kull*, 12 B.R. 654 (1981)
MAKARCHUK	*In re Makarchuk*, 76 B.R. 919 (Bkrtcy. N.D.N.Y. 1987)

MARSCH	*In re Marsch*, 11 B.R. 514 (1981)
MEMPHIS	*Memphis Bank & Trust Co. v. Whitman*, 692 F.2d 427 (1982)
MURALLO	*Matter of Murallo*, 4 B.R. 666 (1980)
MYERS	*Matter of Myers*, 52 B.R. 248 (Bkrtcy. 1985)
NEUFELD	*Neufeld v. Freeman*, 794 F.2d 149 (4th Cir. 1986)
OKOREEH-BAAH	*In re Okoreeh-Baah*, 836 F.2d 1030 (6th Cir. 1988)
OWENS	*In re Owens*, 82 B.R. 960 (Bkrtcy. N.D.Ill. 1988)
PONANSKI	*In re Ponanski*, 11 B.R. 661 (1981)
RASMUSSEN	*In re Rasmussen*, 888 F.2d 703 (10th Cir. 1989)
RIMGALE	*In re Rimgale*, 669 F.2d 426 (1982)
SANABRIA	*In re Sanabria*, 52 B.R. 75 (D.C. 1985)
SANDERS	*In re Sanders*, 28 B.R. 917 (Bkrtcy. 1983)
SCHAITZ	*In re Schaitz*, 913 F.2d 452 (7th Cir. 1990)
SCHONGALLA	*In re Schongalla*, 4 B.R. 360 (1980)
SCHYMA	*In re Schyma*, 68 B.R. 52 (Bkrtcy. D.Minn. 1985)
SELLERS	*In re Sellers*, 33 B.R. 854 (Bkrtcy. 1983)
SEVERS	*In re Severs*, 28 B.R. 61 (Bkrtcy. 1982)
SHEETS	*In re Sheets*, Bkrtcy., 26 B.R. 523 (1983)
SILVA	*In re Silva*, 82 B.R. 845 (Bkrtcy. S.D.Ohio 1987)
SOTTER	*In re Sotter*, 28 B.R. 201 (Bkrtcy. 1983)
STRONG	*Matter of Strong*, Bkrtcy., 26 B.R. 814 (1983)
TAUSCHER	*In re Tauscher*, Bkrtcy., 26 B.R. 99 (1982)
TERRY	*In re Terry*, 630 F.2d 634 (1980)
TRAMONTO	*In re Tramonto*, Bkrtcy., 23 B.R. 464 (1982)
VALENTINE	*In re Valentine*, 29 B.R. 366 (Bkrtcy. 1983)

Appendix B. BankXX's Legal Theory Space

The following legal theories are represented in BankXX:

ABUSE-OF-CHAPTER-13-LEGAL-THEORY
ALL-THE-FACTS-AND-CIRCUMSTANCES-LEGAL-THEORY (ALSO CALLED CASE-BY-CASE-BASIS)
BEST-INTERESTS-OF-CREDITORS-TEST-LEGAL-THEORY
DEANS-LEGAL-THEORY
EASLEY-16-FACTORS-LEGAL-THEORY
ESTUS-LEGAL-THEORY
FLYGARE-LEGAL-THEORY
JOHNSON-ANALYSIS-DISCHARGE-STUDENT-LOANS-LEGAL-THEORY

KITCHENS-KULL-LEGAL-THEORY
LITTLE-INDEPENDENT-MEANING-LEGAL-THEORY
MAKARCHUK-PRINCIPAL-PURPOSE-STUDENT-LOAN-DISCHARGE-LEGAL-THEORY
MEMPHIS-LEGAL-THEORY
OKOREEH-BAAH-LEGAL-THEORY
OLD-BANKRUPTCY-ACT-GOOD-FAITH-DEFINITION-LEGAL-THEORY
OWENS-3-FACTORS-LEGAL-THEORY
PER-SE-MINIMUM-PAYMENT-REQUIREMENT-LEGAL-THEORY
RIMGALE-LEGAL-THEORY
SUBSTANTIAL-OR-MEANINGFUL-REPAYMENT-LEGAL-THEORY

The following theory links are represented in BankXX's Legal Theory Space:
ALL-THE-FACTS-AND-CIRCUMSTANCES – *agrees-with* – FLYGARE
CH-13-ABUSE – *overlaps-with* – FACTS-AND-CIRCUMSTANCES
ESTUS – *conflicts-with* – MAKARCHUK
ESTUS – *derives* – MAKARCHUK
ESTUS – *overlaps-with* – WITH-KITCHENS
ESTUS – *rejects* – PER-SE-MINIMUM
ESTUS-THEORY – *is-equivalent-to* – FLYGARE-THEORY
FLYGARE – *agrees-with* – ALL-THE-FACTS-AND-CIRCUMSTANCES
FLYGARE – *rejects* – PER-SE-MINIMUM
FLYGARE-THEORY – *is-equivalent-to* – ESTUS-THEORY
KITCHENS – *overlaps-with* – ESTUS
LITTLE-INDEPENDENT-MEANING – *rejects* – ESTUS-THEORY
LITTLE-INDEPENDENT-MEANING – *rejects* – SUBSTANTIAL-REPAYMENT
MAKARCHUK – *conflicts-with* – ESTUS
MAKARCHUK – *derived-from* – ESTUS
RIMGALE – *overlaps-with* – ESTUS
RIMGALE – *overlaps-with* – FACTS-AND-CIRCUMSTANCES
RIMGALE – *overlaps-with* – OLD-BANKRUPTCY-ACT
RIMGALE – *rejects* – PER-SE-MINIMUM
SUBSTANTIAL-REPAYMENT – *conflicts-with* – ESTUS

Appendix C. BankXX Domain Factors

The following 26 domain factors are employed in BankXX:
percent-surplus-of-income-factor
employment-history-factor
earnings-potential-factor
plan-duration-factor
plan-accuracy-factor

preferential-creditor-treatment-factor
secured-claims-modified-factor
debt-type-factor
nondischarge-7-factor
special-circumstances-factor
frequency-relief-sought-factor
motivation-sincerity-factor
trustee-burden-factor
relative-timing-factor
relative-total-payment-amount-factor
relative-monthly-payment-amount-factor
use-of-skills-gained-factor
relative-educational-loan-debt-factor
de-minimis-payments-factor
attempts-to-pay-factor
repayment-unsecured-debt-factor
necessary-expenses-minus-plan-payments-factor
unfair-manipulation-factor
inaccuracies-to-mislead-factor
other-relevant-considerations-factor
likelihood-income-increase-factor

Appendix D. BankXX Story Prototypes

We identified the following 19 factual story prototypes in the "good faith" domain. An asterisk $*$ indicates that it is used in BankXX:

automobile-debtor$*$
bankruptcy-repeater$*$
civil-judgment-lien$*$
consumer-debt (credit card junkie)$*$
desperate-economic-trouble-unrealistic-plan
dishonest-debtor$*$
divorce
entrepreneur
family-farm$*$
flatbroke
homeowner
honest-debtor$*$
interrupted-income$*$
irresponsible-debtor
medical-calamity
single-woman

slimy-middle-class-manipulator
*student-loan**
widow

Appendix E. The Hand-Coded *Estus* Opinion

The following is the hand-coded representation of case and theory information found in the actual opinion of the *Estus* case (*In re Estus*, 695 F.2d 311 (8th Cir. 1982)).

AGGREGATED THEORIES:
ALL-THE-FACTS-AND-CIRCUMSTANCES-LEGAL-THEORY
ABUSE-OF-CHAPTER-13-LEGAL-THEORY
BEST-INTERESTS-OF-CREDITORS-TEST-LEGAL-THEORY
PER-SE-MINIMUM-PAYMENT-REQUIREMENT-LEGAL-THEORY
SUBSTANTIAL-OR-MEANINGFUL-REPAYMENT-LEGAL-THEORY
ESTUS-LEGAL-THEORY

LEADING-CITED-CASES:
RIMGALE, GOEB, DEANS, IACOVONI

AGGREGATED-SUPPORTING-CASES:
HEARD, IACOVONI, KULL, TERRY

AGGREGATED-CONTRARY-CASES:
RIMGALE, GOEB, DEANS, BARNES, BELLGRAPH

FACTUAL-PROTOTYPE-STORY:
STUDENT-LOAN

Appendix F. The Estus-problem Case under Node-Type and Argument-Factor Evaluation Functions

The following is aggregated partial output of BankXX run on the Estus-problem case with the node-type evaluation function. Post-1982 items that would be deleted in the post-processing date-filtering are also shown.

AGGREGATED-THEORIES: **DELETED:**
ABUSE-OF-CHAPTER-13-LEGAL-THEORY (before 1980) FLYGARE-LEGAL-THEORY (1983)
PER-SE-MIN-PAYMENT-REQUIREMENT-LEGAL-THEORY (before 1980)
ESTUS-LEGAL-THEORY (1982)
KITCHENS-KULL-LEGAL-THEORY (1981)
OLD-BANKRUPTCY-ACT-GOOD-FAITH-DEF'N-LEGAL-THEORY (before 1980)

LEADING-CITED-CASES:
RIMGALE (1982), GOEB (1982), DEANS (1982), IACOVONI (1980)

AGGREGATED-SUPPORTING-CASES: **DELETED:**
BURRELL (1980), IACOVONI (1980), HEARD (1980) AKIN (1985), RASMUSSEN (1989), SANDERS (1983), CHURA (1983), MAKARCHUK (1987)

AGGREGATED-CONTRARY-CASES: **DELETED:**
DEANS (1982), BARNES (1982), GOEB (1982), RIMGALE (1982) ALI (1983), CRUZ (1987), BAEZ (1989), ASHTON (1988), SCHYMA (1985), FLYGARE (1983)

The following is aggregated partial output of BankXX run on the Estus-problem case with the argument-factor evaluation function. Post-1982 items that would be deleted in the post-processing date-filtering are also shown.

AGGREGATED-THEORIES: **DELETED:**
PER-SE-MIN-PAYMENT-REQUIREMENT-THEORY (before 1980) FLYGARE-LEGAL-THEORY (1983)
ESTUS-LEGAL-THEORY (1982)
KITCHENS-KULL-LEGAL-THEORY (1981)

LEADING-CITED-CASES:
RIMGALE (1982), GOEB (1982), DEANS (1982), IACOVONI (1980)

AGGREGATED-SUPPORTING-CASES: **DELETED:**
BURRELL (1980), IACOVONI (1980), HEARD (1980) AKIN (1985), RASMUSSEN (1989), CHURA (1983), MAKARCHUK (1987)

AGGREGATED-CONTRARY-CASES: **DELETED:**
DEANS (1982), BARNES (1982), GOEB (1982), RIMGALE (1982) ALI (1983), CRUZ (1987), BAEZ (1989), ASHTON (1988)