Producing BT's Yellow Pages with Formation

Gail Anderson, Andrew Casson-du Mont, Ann Macintosh and **Robert Rae**
AIAI, University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN, Scotland UK
email:aiai@aiai.ed.ac.uk
tel: +44 31 650 2732 fax: +44 31 650 6513

Barry Gleeson
Pindar Set Ltd.
Newlands Park Drive
Scarborough, North Yorkshire YO12 6DT
email:barry@pindar.com
tel: +44 1723 500455 fax: +44 1723 367852

Abstract

This case study illustrates how the adoption of AI technology can benefit smaller companies as well as major corporations.

Pindar Set is a small UK company which has originated the Yellow Pages directories for British Telecommunications plc since 1979. AIAI is a technology transfer organisation which has delivered innovative solutions to industrial clients since 1984. Together, AIAI and Pindar have developed a next-generation layout system, Formation.

Formation is fast, easy to use and flexible, and had already delivered benefits through marketing trials before being successfully deployed in production of the Yellow Pages in December 1997.

The heart of Formation is a 2D layout engine which formats input data according to styles written in LSSL, a domain-specific language developed at AIAI.

Through representing the layout knowledge in Formation explicitly in LSSL styles, and ensuring that it can easily be modified, Pindar has enabled itself to respond far better to its customer's present and future needs.

Background

Pindar's print works in Scarborough, NE England, were founded in 1836. In the 1960s, Pindar introduced photo typesetting to widen its market, and in the 1970s it became an early investor in computer technology. Pindar won the contract to originate the British Telecommunications Yellow Pages (BTYP) directories in 1979. Although a small company employing only 200 people, Pindar Set, the company within the Pindar group of companies which services BTYP, has invested in new technology which can provide faster, better services for its client.

AIAI at Edinburgh University was established in 1984 to promote the application of AI techniques. Since then, it has delivered solutions to many industrial and commercial clients. With its record of producing innovative solutions for industry – and, importantly, carrying out technology transfer to ensure that they work – AIAI was a natural choice to partner Pindar in producing a next-generation layout solution for BTYP, Formation.

Task Description

BTYP publishes 74 different regionally-based classified directories annually. Print runs vary from about 25,000 copies for the Isle of Man to over half a million for Glasgow South. A typical 1,500 page directory may contain 30,000 businesses under 3,000 classification headings, and 5,000 display advertisements. Pindar has used computers to lay out BTYP from the beginning, and today's tight timescales and large directory sizes make computer-based layout essential. However, Pindar had found the layout programs difficult to maintain in the face of changing requirements.

By the beginning of the 1990's, advertisers were becoming more demanding, and BTYP wanted to introduce new features which required extensive market testing. Pindar needed to be able to respond to changes such as the introduction of new advertisement

types, and be able to try out different layout scenarios quickly and reliably.

Pindar's increasing presence in worldwide classified telephone directory (CTD) production, and Pindar's future requirement to be able to use the new system to lay out other types of publication (such as newspapers or catalogues) meant that the system had to be flexible and configurable enough to produce publications with many different layout requirements. Layout requirements for CTDs are usually expressed in terms of page appearance, usability, and aesthetics. Increasingly, publishers also need to able to provide more specialised and customised services for their advertisers, who may want their entries to appear in particular positions on the page, and so on.

Pindar needed a knowledge-based system which allowed publishers to specify their requirements both naturally and precisely. The layout knowledge used by the system needed to be explicit rather than implicit, so that it could easily be modified. For commercial reasons, it was important that the system could run on standard PC hardware. Although knowledge-based layout systems did exist (Camara, Martins, & Jácome 1990; Chew & Liang 1994; Graf 1995), none of them met all Pindar's requirements for flexibility, speed, predictability, maintainability and price, so work began on Formation in 1994. Formation went into full production use at Pindar in December 1997; the first directory produced by the new system was for the Shrewsbury area – the same directory first produced by Pindar in 1980.

Application Description

Each BTYP directory covers a single geographical area. Sales representatives visit businesses throughout the area selling display advertising, and telephone sales representatives sell smaller, cheaper advertisements. Most display advertisements are produced by Pindar's and BTYP's graphics studios: once each has been approved by the advertiser, it is sent to Pindar's production site in Scarborough.

When the directory is "closed" (that is, when no more orders will be accepted from advertisers), BTYP produces compiled data containing classification headings, display and semi-display advertisements, and listings. This data is presented in sequence, with advertisers appearing alphabetically within their class. The task of the layout system is to place all the entries within the directory in as close a sequence as possible to the original alphabetical sequence, while minimising the number of pages used and hence the amount of "filler" material generated. Maintaining a good visual balance is also increasingly important.

Formation lays out input data according to particular layout styles, and a BTYP style has been implemented. This specifies things like where particular types of item can be placed on the page, and what changes to the input sequence are allowed. Layout styles are implemented in the Layout Style Specification Language (LSSL), an object-oriented domain-specific language designed by AIAI. It is intended to enable the LSSL programmer to describe the layout process in a natural manner, while remaining precise.

Once layout is complete, Pindar carries out post-processing of the data, and combines the layout produced using Formation with the graphics produced by the studio to produce film ready for printing.

Formation Overview

The diagram in figure 1 shows how the Formation system can be delivered to a particular publisher. At its core is the general 2D layout engine, the LSSL interpreter. In addition to this, there is a library of generic style elements, implemented in LSSL.

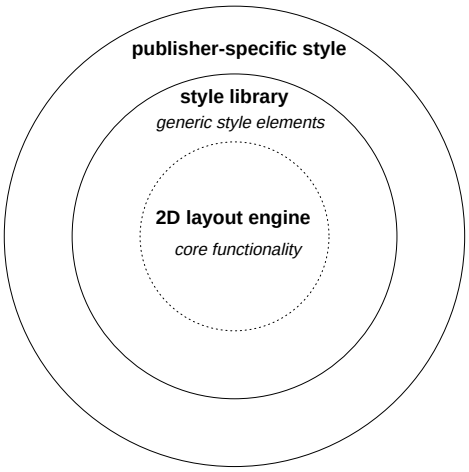

Figure 1: Formation design

At present, the style library contains elements which deal with layout in general and elements which are useful specifically for layout of classified telephone directories. As our experience with using Formation to lay out different documents grows, we are adding to the style library.

Using these generic elements and LSSL primitives, a style is implemented to specify the publisher's in-house requirements. Usually, the style will provide parameters through which the user can control the finished layout. For example, one of the parameters in the BTYP style specifies how groups of line entries can be split, and the CTD style described in an earlier paper (Anderson *et al.* 1996), which was produced for use in production of certain US telephone directories, has

a parameter which specifies whether or not a display advertisement has to appear amongst the line entries of its own classification.

The publisher can use the Formation graphical user interface to configure the house style, setting the parameters to specify exactly how each publication should be laid out, with no programming required.

Once the publisher is happy, the same interface can be used in production to control and monitor layout. Figure 2 shows the Formation user interface laying out some pages from a BTYP directory.

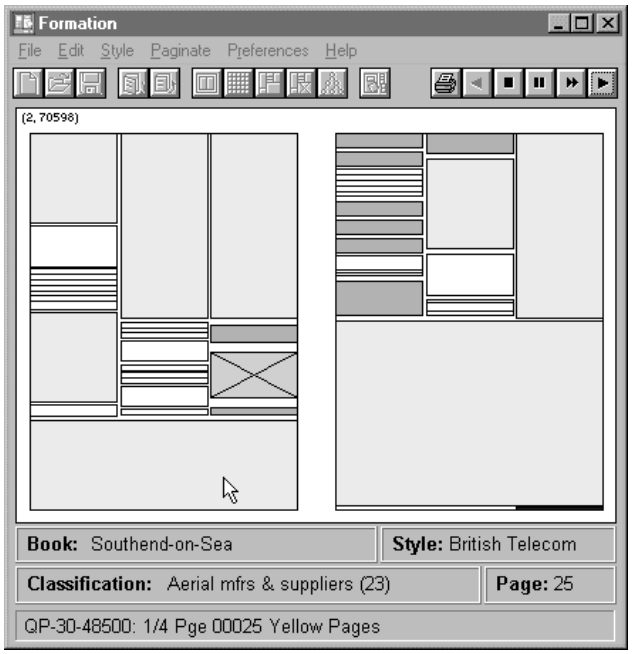

Figure 2: The Formation GUI under Windows95

Through the judicious provision of style parameters, the publisher can therefore be offered a great deal of flexibility in choosing the layout of his documents. At Pindar, the Formation user interface also provides a natural and accessible way of demonstrating the power of the layout engine to Pindar's clients.

However, should the customer want greater flexibility than is offered through configuring an existing style, he can have it, because the full power of LSSL is available to the style programmer.

There are inevitably practical difficulties in fully and unambiguously describing how to handle every interaction that can result from every possible combination of input data, so very occasionally the operators need to make manual changes to the finished layout. More commonly, the publisher has to make late changes – such as the inclusion of new advertisers, or corrections to the text of line entries – so support for human intervention is a continuing requirement. When it is used to produce the BTYP directories at Pindar, the LSSL interpreter is harnessed to a page editor developed by Pindar.

LSSL and the Layout Engine

The 2D layout engine is a LSSL interpreter. It is implemented in Allegro Common Lisp, and runs on a Pentium-based Windows PC in production at Pindar. Because it is in Common Lisp, it is portable, and it also runs under Macintosh and UNIX operating systems.

LSSL is a small, object-oriented language for describing general 2D layout requirements. It has dynamic typing, simple object semantics, and provides mechanisms for event-driven programming which support precise specification of the layout process.

The representation of a LSSL document is based on the concepts of *block* and *grid*. A block is a rectangle; blocks are sub-typed into *regions*, which are rectangles which can be filled by the layout process, and *items*, which represent the data to be laid out. Regions can be divided into smaller regions by *divider* grids, enabling the style programmer to specify different logical parts of the page, while *page* grids are used to specify the positions in which items can be placed.

Using these concepts, then, the geometry of the page is described. The document is represented as a queue of *spread* regions. A spread is usually – but not necessarily – two facing pages (see figure 3).

LSSL provides primitives for defining styles and documents, and for controlling pagination. Layout is usually carried out through calling the built-in `paginate` procedure, as follows. First, there is some pre-layout preparation: input and output files are opened, and various objects needed for layout are initialised. Then the document is laid out in a well-defined process. At the end, things are tidied up, and `paginate` compiles and writes out a layout report.

The Style Library

The style library provides generic and reusable style elements, implemented in LSSL, which can be customised by a style programmer. Through the use of the inheritance mechanisms provided by LSSL the style programmer can define ever more specific requirements. At present, the style library contains completely generic elements for 2D layout, plus elements which are of use within the more specialised domain of CTD layout.

For example, we said that a spread is usually *but not necessarily* two facing pages. The concept of a spread is built in to the style library, and by default, a spread will contain a 2 by 1 divider grid. This type of grid is

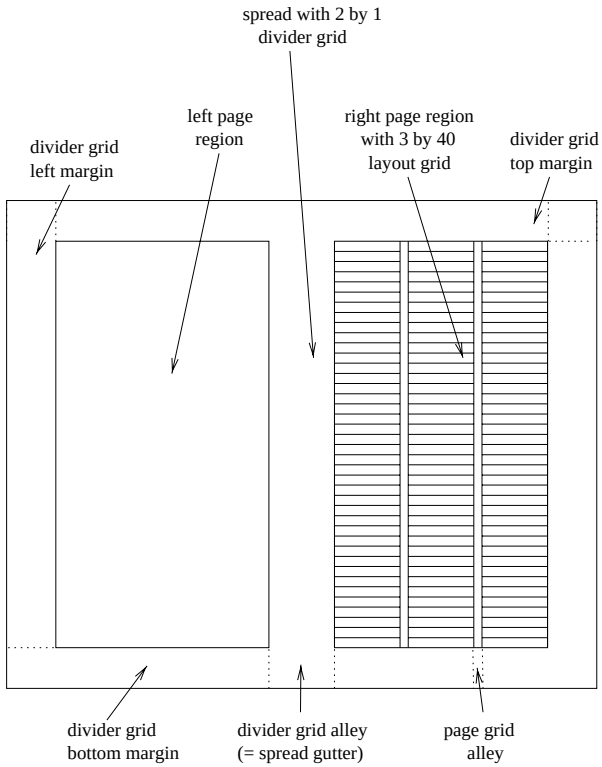

Figure 3: Spread structure with regions and grids

the norm for CTDs in general, and BTYP directories are no exception. However, if we wanted to write a style for laying out a single-page flier, we could do so by defining a single-spread page, overriding the default. Figure 4 shows how the style library is developing.

The Formation User Interface and the Page Editor

The Formation user interface (see figure 2) is written using the Allegro Common Lisp GUI Builder, and runs only on a PC under Windows. The Page Editor, however, is implemented in C++ using Microsoft Foundation Classes. This gives Pindar portability across the PC, Apple and Sun platforms, as well as allowing easy access to PC dependent facilities such as OLE.

As well as allowing the user to configure a style through changing and specifying its parameters (and in particular, changing the parameters of the strategy and methods), the user interface provides facilities for describing the page geometry and the types of item which can appear in the book.

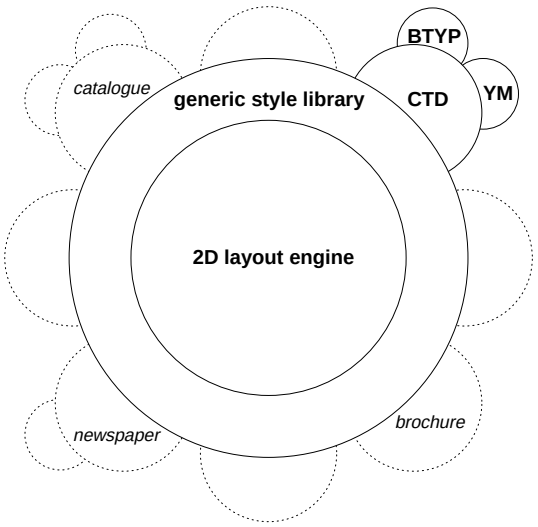

Figure 4: Development of the style library

The Contents of a Style

A layout style comprises: the page geometry (regions and grids); the different types of item which can appear on a page (a hierarchy of types, usually specifiying sizes, and required spacing); a specification of the required layout process, or *strategy* (e.g. lay out a spread at a time, or consider a whole classification at once); and detailed knowledge about how to position items on a page, in the form of layout *methods* and *rules*.

Note that in LSSL a layout method is simply an object which encapsulates knowledge about a particular aspect of page layout. For example, in the BTYP style, there is a particular method which specifies how to keep entries in sequence, and another which specifies how to position multi-column display advertisements. Each layout method is implemented as a package of event-driven layout rules; a rule defines a set of actions which is to be carried out when a particular event happens.

Layout Rules

LSSL provides the programmer with the ability to control the layout of a page through rules which are defined for certain objects and are executed when particular events occur. LSSL layout rules are therefore not production rules. While production rules encode their conditions explicitly, LSSL layout rules do not. Instead, the conditions under which LSSL rules are fired are defined in terms of LSSL events. Built-in LSSL procedures – such as *get-space*, which finds a space in which an item can fit, and *align*, which tests to see whether the item can be positioned in that space given

the layout requirements – signal events, but the LSSL programmer can also signal his own events.

Since every rule defined for an item at a particular event is executed when that event occurs, it is important that rules do not conflict. Some care is necessary to package rules together in discrete methods which can be enabled or disabled on demand.

For example, the method for keeping entries in sequence in BTYP contains a rule to check that an entry is aligned following the last one already placed on a page:

```
;;; Keep entries in sequence

(a method eis "Entries in sequence"
   ("Sequenced entry type" entry))

;;; Align a sequenced entry following the
;;; last one on the page

(a rule eis.follows-last? (for eis.entry)
                          (at align)
                          :(it gd _ _) ->
   (or (not gd.last-entry)
       (eis.follows? it gd.last-entry)))
```

The rule is part of the `eis` method, and is stored in a slot in that method. Note that the type of item for which it is defined can be configured through the method parameter `eis.entry`. `eis.follows-last?` will be executed when any item of this type is aligned. It carries out a simple test to see whether there are any entries on `gd`, the page grid on which this item is being aligned, and, if there are, it runs a method-specific procedure `eis.follows?` to check whether this entry is aligned following the last. Of course, the definition of `eis.follows?` could be changed to accommodate the requirements of a different publisher.

Through encapsulating the knowledge about the look of a finished page in methods and rules, then, we are able build up a library of configurable, reusable components.

Design Criteria

Formation was designed to satisfy two important requirements: flexibility in describing or modifying the details of a particular style of layout, and high throughput. It was required to produce CTDs for BTYP, but was always intended to be more general. It therefore has a modular design, and is based on the general-purpose framework provided by LSSL. Unlike other systems (Camara, Martins, & Jácome 1990; Graf 1995), the emphasis was on carrying out correctly and predictably what has been specified, rather than selecting an optimal solution from several candidates.

Uses of AI Technology

Since Pindar required a system which would run reliably in production, AIAI chose to use mature technology. Common Lisp was chosen as the development language because it is particularly well-suited for manipulating symbolic data, and because features such as automatic storage management improve programmer productivity. AIAI staff already had Lisp expertise. Franz Inc's Allegro Common Lisp was used because it provides a high-quality, well-supported development environment.

In developing Formation, AIAI's skills and experience in knowledge acquisition, knowledge representation and symbolic computation were particularly useful. A disciplined knowledge-engineering approach was taken from the beginning, and in the early stages of development AIAI carried out structured interviews with Pindar staff. Through these we were able to capture: BTYP's current requirements; expertise resulting from the continuing need to edit pages after layout is complete; and previous experience of laying out CTDs by hand before computerised layout was possible. The results were used to produce a layout ontology from which LSSL and the style library have developed.

It was important to Pindar that the delivered system be maintainable by Pindar staff, so modularity has been important to the design and development of Formation. We decided early on that given Pindar's requirements, including tight timescales and the requirement for mature technology, the available constraint-based technologies would not suit. However, we originally expected to use a very simple representation of layout constraints in the layout engine.

Early experiments with these proved that in order to meet BTYP's requirement to be able to specify (and monitor) layout very precisely, we needed a much more deterministic approach. The event-driven rules in LSSL were the result, and we found it easy to incorporate them into the general and modular framework we had developed. Should there be a requirement for a more constraint-based alternative approach in the future, Formation's design, and the object-oriented nature of LSSL, will accommodate this.

Pindar chose to implement the page editor in C++ for business reasons, including the ability to take advantage of in-house skills in C++. The choice of Microsoft Foundation Classes provided portability, and delivery on a PC made it easy and cheap to deliver hardware for production as well as permitting off-site demonstration on laptop computers.

Application Use and Payoff

Formation is used in production at Pindar in batch mode. The text listings to be included in a book – such as the classification headers and plain text listings – are typeset; when typesetting is complete, the space needed on the printed page for each text entry is known. LSSL object definitions for these entries are then combined with definitions for the graphical entries – such as the display advertisements – to produce a single input stream. Pindar production staff then run Formation in batch mode. When layout is complete, the output stream of LSSL object definitions is combined with the typesetting information and graphics to produce printed pages for proof-reading. On the occasions when editing is necessary, perhaps because of last-minute changes from the publisher, these are made using the page editor developed by Pindar. The whole process, from the point when the compiled data arrives at Pindar till the finished film is sent to the printers, typically takes only 3 days.

Long before Formation went into full production use for laying out BTYP at Pindar in December 1997, its high throughput and great flexibility had already brought Pindar clear business benefits.

The ability to describe layout knowledge in a concise and modular way has enabled us to make changes requested by BTYP very quickly, and Pindar has been very pleased with the speed at which styles can be updated. Changes can be made to the layout produced by Formation in hours or days; with previous systems, even apparently minor changes might take weeks. The introduction of new advertisement types has proved painless.

Using Formation, Pindar has been able to produce special sections such as Eating-Out Guides which were previously compiled manually by the operators. The ability to trial different layouts has helped Pindar to produce samples which BTYP have used for market research, with the confidence that should BTYP decide to produce directories with these new features, Pindar can deliver quickly and easily.

The ability to collect statistics such as the total percentage of filler space in the book, or the number of full-page display advertisements, and to control through LSSL styles which of these are reported to the operator or user, is an additional benefit. It provides Pindar with a mechanism for measuring the impact of changes to layout quantitatively, and for identifying those which will have a direct impact on costs.

Although the ability to program styles in LSSL makes Formation extremely flexible, the provision of the Formation GUI makes the system very easy for a non-programmer to use and understand. Apart from reducing training costs, this ease of use means that Pindar can readily demonstrate Formation to existing and prospective customers. Furthermore, since Pindar has the ability to reconfigure a style's parameters and lay its example document out again and again in front of a customer, it is easy for the customer to see and to understand the impact of changes and the potential benefits they will bring him.

As the detailed knowledge which controls the 'look and feel' of the finished document is readily identifiable, it can also be used to inform advertisers about the rules that are applied to determine the positioning of their advertisements. This is important to BTYP, as their staff have to be in a position to explain to advertisers why their entries appear where they do; the ability to give precise answers could eventually reduce costs incurred by BTYP through customer complaints.

Speed of layout is an important aspect of the system due to both the number of directories that have to be processed and the relatively short timescales involved in their production and printing. Formation can lay out a typical directory of 1500 pages at a speed of well over 1,000 pages per hour on a 100 MHz Pentium-based PC.

Although Formation was intended specifically to produce classified telephone directories, it is a much more general system and can be used for the two-dimensional layout of general shapes based on rectangles. Because of this general framework, Pindar can see enormous potential for marketing Formation in Europe, the USA and the Far East.

Application Development and Deployment

Formation was developed at AIAI by a project team of five. The skills of the individuals in the team complemented each other, and covered requirements capture, knowledge elicitation and modelling, AI software design and implementation, and project management. The initial development of the LSSL interpreter and the BTYP style took approximately 1 man year, and was carried out over a period of 10 months, during which regular review meetings were held with Pindar staff. The system was delivered on time and to specification.

In order to ensure that the system met operational requirements, Pindar's Production Operations Manager was involved, from specifying the requirements through to accepting the delivered system. His input was essential, as he has overall responsibility for the typesetting and formatting process by which the BTYP directories are produced.

Throughout its development, a high priority has been to ensure that Pindar is able to maintain and

modify Formation in the future. As a technology transfer organisation, AIAI aimed not to deliver a 'black box' product, but to provide its client with a long-term solution. Pindar's technical staff visited AIAI regularly to work with the project team, and they were involved in discussion and comment at all stages of design and implementation. Following project delivery, members of Pindar's support staff spent further time at AIAI learning about the system.

Since January 1996 Formation has been used by Pindar in development and for market research, and it has proved very flexible. During this time it was also used in production trials, and we gained a great deal of expertise about layout specification. It became clear early on that page layout systems are very liable to incomplete and inaccurate specification. Some of AIAI's efforts in technology transfer, therefore, have centred on providing tutorial material which will help Pindar to write clear and concise style specifications. A great deal of effort has gone into ensuring that the BTYP style is very well documented and that Pindar technical staff can modify it.

By contrast, since Formation is used in production in batch mode, very little training in its use has been necessary for production staff.

Formation successfully replaced the previous system in production use in December 1997, together with the Page Editor. The first book produced with it was the 1998 directory for the Shrewsbury area.

Maintenance

It is very important to Pindar that maintenance of the layout system is not dependent on the developers; AIAI delivers tutorials to ensure that Pindar staff understand new developments, as well as producing reference and tutorial documentation.

The strategic partnership between AIAI and Pindar enables continuing development, and current efforts are concentrated on further development of the style library.

The provision of style parameters which can be modified though the user interface has enabled even beginners to change Formation's behaviour. The additional facilities in the interface for specifying page geometry and item types allow the user to make changes to the declarative knowledge defined by the style without ever needing to program.

These facilities, together with the growing style library and the ability to program entirely new features in a modular and natural fashion, are particularly useful in this time of rapid change in publishing and advertising. BTYP's layout requirements have changed several times since work on Formation began in 1994, but it has proved easy to accommodate those changes. Pindar expects to make even greater use of Formation's flexibility in future, as new features currently under discussion start to appear in BTYP directories.

Conclusion

The successful development and deployment of Formation illustrates how a relatively small, privately owned company can benefit from using applied AI technology. Through working with AIAI to ensure that its layout knowledge is understandable and easily modified, Pindar Set has enabled itself to respond far better to its customer's needs, both present and future.

References

Anderson, G.; Casson, A.; Macintosh, A.; Rae, R.; Gleeson, B.; and Carter, S. 1996. Formation: Knowledge-based layout of classified telephone directories. In *Applications and Innovations in Expert Systems IV: Proceedings of Expert Systems 96.* British Computer Society.

Camara, J. A.; Martins, J. F.; and Jácome, M. S. 1990. ATENA: A knowledge based system for automatic pagination of Yellow Directories. In *Research and development in expert systems VII: Expert Systems 90, Proceedings of the Tenth Annual Technical Conference of the British Computer Society specialist group on Expert Systems.*

Chew, H.-G., and Liang, M. 1994. ALEXIS: An Intelligent Layout Tool for Publishing. In *Proceedings of the 6th Innovative Applications of Artificial Intelligence Conference.* Seattle, Washington: AAAI.

Graf, W. H. 1995. The Constraint-Based Layout Framework LayLab and its Applications. In *Proceedings of the Workshop on Effective Abstractions in Multimedia Layout, Presentations, and Interaction.* San Francisco: ACM.