Diagrammatic Reasoning and Cases

Michael Anderson
Computer Science Department
University of Hartford
200 Bloomfield Avenue
West Hartford, Connecticut 06117
anderson@morpheus.hartford.edu

Robert McCartney
Department of Computer Science and Engineering
University of Connecticut
191 Auditorium Road
Storrs, Connecticut 06269-315
robert@cse.uconn.edu

Abstract

We believe that many problem domains that lend themselves to a case-based reasoning solution can benefit from an diagrammatic implementation and propose a diagrammatic case-based solution to what we term the n-queens best solution problem where the best solution is defined as that which solves the problem moving the fewest queens. A working system, based on a novel combination of diagrammatic and case-based reasoning, is described.

Introduction

Interest in computing with analogical representations is on the rise. This is evidenced by the attention given to them in recent symposia (e.g. [Narayanan 1992]), journals (e.g. [Narayanan 1993]), and books (e.g. [Glasgow, Narayanan, & Chandrasekaran 1995]). We believe that such attention is the natural outgrowth of the evolution of the currency of computing, the first two generations of which were numeric and symbolic.

Our particular interest lies in developing a general set of operators that can be used to reason with sets of related diagrams— *inter-diagrammatic reasoning*. This concept has been explored in [Anderson 1994; Anderson & McCartney 1995a; Anderson & McCartney 1995b] in which a heuristic for a game has been developed, musical notation and Venn diagrams have been reasoned with, and information from cartograms and various type of charts has been inferred using a general set of operators. Along these lines, we have been investigating the integration of inter-diagrammatic reasoning with case-based reasoning.

We contend that many problem domains that lend themselves to a case-based reasoning solution can benefit from an inter-diagrammatic implementation. For example, domains that deal with spatial configuration, navigation [Goel et al. 1994], and perception all might benefit from explicit representation of cases stored as diagrams, retrieved via a diagrammatic match, and modified diagrammatically.

We propose a diagrammatic case-based solution to what we term the *n-queens best solution problem* where the best solution is defined as that which solves the problem moving the fewest queens, leaving queens that are already in place untouched (versus a solution that solves the problem in the fewest moves). Further, we develop an inter-diagrammatic implementation of the *min-conflicts heuristic* [Gu 1989] to find solutions to randomly chosen *n-queens problems* [Stone & Stone 1986] themselves.

We present a syntax and semantics of inter-diagrammatic reasoning and then introduce the inter-diagrammatic operators and functions. Next, case-based reasoning is briefly overviewed. This is followed by a description of the diagrammatic solution of the *n*-queens problem and the diagrammatic case-based solution of the *n*-queens best solution problem. A brief discussion of related work follows and, finally, we offer our conclusions.

Inter-diagrammatic Reasoning

Most generally, one can syntactically define a *diagram* to be a tessellation of a planar area such that it is completely covered by atomic two dimensional regions or tesserae. The semantic domain will be defined as $\{v_0, \ldots, v_{i-1}\}$ denoting an i valued, additive gray scale incrementally increasing from a minimum value v_0 ,WHITE, to a maximum value v_{i-1} , BLACK. Intuitively, the gray scale values correspond to a discrete set of transparent gray filters that, when overlaid, combine to create a darker filter to a maximum of BLACK.

The following primitive unary operators, binary operators, and functions provide a set of basic tools to facilitate the process of inter-diagrammatic reasoning.

Unary Operators

NOT, denoted $\neg d$, is a unary operator taking a single diagram that returns a new diagram where each tessera's value is the difference between BLACK and its previous value.

Binary Operators

Binary operators take two diagrams, d_1 and d_2, of equal dimension and tessellation and return a new diagram where each tessera has a new value that is some function of the two corresponding tesserae in the operands.

OR, denoted $d_1 \vee d_2$, returns the *maximum* of each pair of tesserae where the maximum of two corresponding

tesserae is defined as the tessera whose value is closest to BLACK.

AND, denoted $d_1 \wedge d_2$, returns the *minimum* of each pair of tesserae where the minimum of two corresponding tesserae is defined as the tessera whose value is closest to WHITE.

OVERLAY, denoted $d_1 + d_2$, returns the *sum* of each pair of tesserae (to a maximum of BLACK) where the sum of values of corresponding tesserae is defined as the sum of their respective values' subscripts.

PEEL, denoted $d_1 - d_2$, returns the *difference* of each pair of tesserae (to a minimum of WHITE) where the difference of values of corresponding tesserae is defined as the difference of their respective values' subscripts.

ASSIGNMENT, denoted $d_1 \Leftarrow d_2$, modifies d_1 such that each tessera has the value of the corresponding tessera in d_2. (Note that non-diagrammatic assignment will be symbolized as := and the equality relation as =.)

Functions Over Diagrams

NULL, denoted NULL(*d*), is a one place Boolean function taking a single diagram that returns TRUE if all tesserae of *d* are WHITE else it returns FALSE.

NONNULL, denoted NONNULL(*d*), is a one place Boolean function taking a single diagram that returns FALSE if all tesserae of *d* are WHITE else it returns TRUE.

Functions Over Sets of Diagrams

ACCUMULATE, denoted ACCUMULATE (*d*, *ds*, *o*), is a three place function taking an initial diagram, *d*, a set of diagrams of equal dimension and tessellation, *ds*, and the name of a binary diagrammatic operator, *o*, that returns a new diagram which is the accumulation of the results of successively applying *o* to *d* and each diagram in *ds* .

MAP, denoted MAP(*f*, *ds*), is a two place function taking a function *f* and a set of diagrams of equal dimension and tessellation, *ds*, that returns a new set of diagrams comprised of all diagrams resulting from application of *f* to each diagram in *ds*.

FILTER, denoted FILTER(*f*, *ds*), is a two place function taking a Boolean function, *f* and a set of diagrams of equal dimension and tessellation, *ds*, that returns a new set of diagrams comprised of all diagrams in *ds* for which *f* returns TRUE.

RANDOM, denoted RANDOM([*x*,]*s*), is a one or two place function that returns a set of *x* unique elements of *s* at random, *x* defaulting to 1 if not present.

CARDINALITY, denoted CARDINALITY(*s*), is a one place function taking a finite set that returns the number of elements in *s*.

Case-based Reasoning

Case-based reasoning [Kolodner 1993] is the use of previous problem solving episodes (with solutions) to solve a new problem. Cases can be used for two purposes: 1) to support plausible inferencing in the absence of a complete domain theory [McCartney 1993], and 2) to increase efficiency by either providing partial solutions or providing focus and direction to problem solving efforts [Kambhampati & Hendler 1992]. For both of these purposes, case-based reasoning provides an obvious learning mechanism: as problems are solved, new episodes are incorporated into the case base, which can later be used in future problem solving.

Implementing a case-based reasoning system requires answering a number of fundamental questions.

- *representation*: What is a case and how is it represented?
- *indexing*: How is a case stored and retrieved?
- *similarity*: How do we determine which case is most appropriate to use in solving a given problem?
- *adaptation*: How do we use an appropriate case once we get it?

These questions have obvious general answers given our interest in diagrammatic reasoning. Cases will be diagrams, represented in a way consistent with the proposed syntax and semantics, and algorithms used for indexing, similarity, and adaptation of a case will be defined in terms of diagrammatic operators. As we are working with a complete domain theory and no uncertainty, we are using case-based reasoning to increase efficiency and provide a mechanism to improve performance over time.

Diagrammatic Constraint Satisfaction

Diagrammatic reasoning can be used to solve *constraint satisfaction problems*— problems in the form of a set of variables that must satisfy some set of constraints. The *n*-queens problem, for example, can be viewed as a constraint satisfaction problem that can be solved diagrammatically.

A solution to the *n*-queens problem is any configuration of *n* queens on an *n* by *n* chessboard in which no queen is being attacked by any other queen. Figure 1 shows a diagram of a solution to the problem when $n = 8$. When the location of each queen is considered a variable that must meet the constraint that no other queen can attack that location, a constraint satisfaction perspective of the problem arises. The min-conflicts heuristic, which advocates selecting a value for a variable that results in the minimum number of conflicts with other variables, can be

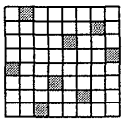

Figure 1: n-queen solution where $n = 8$

implemented diagrammatically to solve the n-queens problem.

A diagram in the n-queens domain is represented as an n by n tessellation of gray-scale valued tesserae. A set of n by n diagrams comprised of all possible single queen positions (denoted *Queens*) must be defined. Each of these diagrams represents one possible position of a queen (in a medium level gray) and the extent of its attack (in GRAY1). Figure 2 shows a diagram of n OVERLAYed queen diagrams and each of the corresponding diagrams from *Queens* that represent the individual queens in question where $n = 8$. Given a random selection of queen positions, the strategy is to move iteratively the most attacked queen to a position on the board that currently is the least attacked until a solution is discovered.

Discovering a Solution

After a random selection of queens is made, all the corresponding diagrams from *Queens* (denoted *SelectedQueens*) are OVERLAYed onto a single diagram (denoted *CurrentBoard*). This process can be more formally represented using the proposed diagrammatic operators as

CurrentBoard⇐
ACCUMULATE (∅, *SelectedQueens*, +)

The *CurrentBoard* is checked to see if it is a solution by PEELing from it a diagram that is completely covered in the same gray level that represents a queen (denoted *QueenGrayBoard*). Only if the result of this operation is a diagram with all WHITE tesserae (denoted *NullDiagram*) has a solution been found. More formally stated, a solution will return TRUE for

NULL (*CurrentBoard* - *QueenGrayBoard*)

As long as the gray level representing queens is greater than any gray level achievable by simply OVERLAYing the GRAY1 tesserae representing queen attack extents, a tessera will only take on a gray level greater than that representing queens if one or more GRAY1 tesserae is OVERLAYed upon a queen. If such a level of gray is found, a queen is under attack. Therefore, if the previous PEEL operation does not remove all gray from a diagram, it cannot be a solution. If a solution has yet to be found, an attacked queen is PEELed from the current diagram and a

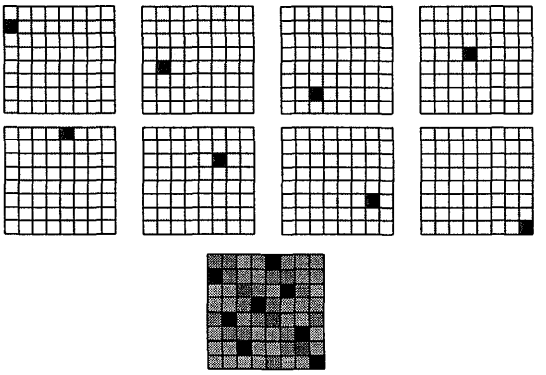

Figure 2: OVERLAYing 8 queen diagrams

new queen is OVERLAYed at a minimally attacked location.

An attacked queen (denoted *AttackedQueen*) is found by ANDing a GRAY1-PEELed version of all diagrams from *SelectedQueens* with the results of the solution test and randomly selecting from those queens that do not produce the *NullDiagram* (i.e. those queens that correspond with NON-WHITE tesserae in the diagram resulting from the solution test). More formally:

AttackedQueen⇐
RANDOM
(FILTER
(NULL,
MAP (λ(*x*) ((*x* - *Gray1Board*)
∧
(*CurrentBoard* - *QueenGrayBoard*)),
SelectedQueens)))

AttackedQueen is PEELed from the *CurrentBoard* and a minimally attacked queen is OVERLAYed in its place. By definition, the minimally attacked queen (denoted *MinimalQueen*) on the current diagram will be the queen at the location that is the lightest gray level. These locations are found by ANDing a GRAY1-PEELed version of all unused diagrams from *Queens* (denoted *UnusedQueens*) with the current diagram and randomly selecting from those queens that produce the *NullDiagram* (i.e. those queens that correspond with WHITE tesserae in *CurrentBoard*). More formally:

MinimalQueen⇐
RANDOM
(FILTER
(NONNULL,
MAP (λ(*x*) ((*x* - *Gray1Board*)
∧
(*CurrentBoard* - *AttackedQueen*)),
UnusedQueens)))

If no such queen is found, a diagram that is completely covered in GRAY1 (denoted *Gray1Board*) is iteratively PEELed from the current diagram, making all tesserae one gray level lighter, and the process repeated. More formally:

CurrentBoard ⇐ *CurrentBoard* - *Gray1Board*

MinimalQueen is then OVERLAYed upon the current diagram. More formally:

CurrentBoard ⇐
CurrentBoard - *AttackedQueen* + *MinimalQueen*

This new diagram is checked to see if it is a solution and the process continues until such a solution is discovered.

An Example

Figures 2 through 4 graphically display an example of the solution finding process where $n = 8$. Figure 2 shows the queen diagrams selected from *Queens* as well as the diagram that results from OVERLAYing these diagrams.

Figure 3 displays one iteration of this process. 3a shows the solution check, *QueenGrayBoard* is PEELed from the current diagram. This diagram is not a solution because the result is not the *NullDiagram*. In 3b, one of the attacked queens is selected and PEELed from the current diagram. Since there are no WHITE tesserae, *Gray1Board* is PEELed from the result in 3c. In 3d, a queen diagram is randomly selected from the set of queen diagrams that correspond to the WHITE tesserae in the result and OVERLAYed on the current diagram.

Figure 4 shows the next two iterations of the solution finding process. 4a displays the solution check for the current diagram created by the last iteration. This is also found not to be a solution, so an attacked queen's diagram is PEELed from the current diagram in 4b. Since there is a WHITE tesserae in the result, PEELing *Gray1Board* from it is not required. The only possible new queen diagram is then OVERLAYed on the current diagram in 4c. 4d shows the solution check for the third iteration and, as this is found to be a solution (i.e. the check results in the *NullDiagram*), processing stops. The result of the entire process is the 8-queen problem solution presented in 4e.

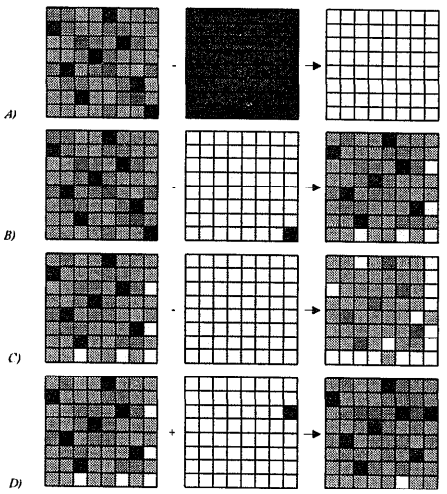

Figure 3: 8-queens example, 1st iteration

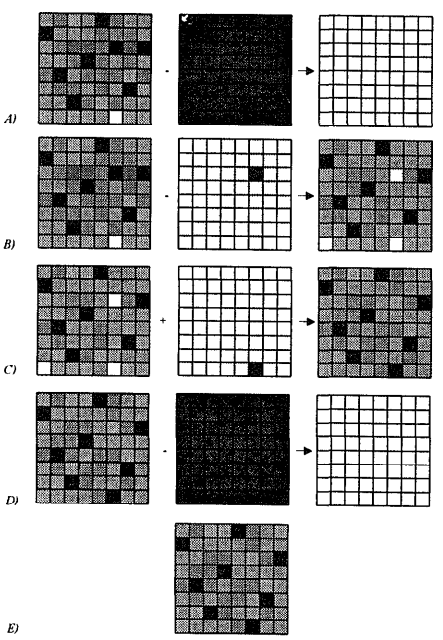

Figure 4: 8-queens example, 2nd and 3rd iterations

Diagrammatic Case-based Best Solution

A solution to an *n-queens best solution problem* is an n-queens placement obtained by moving the fewest queens from some initial placement. Although finding this minimal solution can only be achieved at great computational cost, we have implemented a system that improves its performance at this task by making use of previous solutions it has developed. Solutions to previous problems can be used to provide partial solutions to the current problem.

These previous solutions form the *cases* of our case-based reasoning solution. Case *representation* is defined diagrammatically as an OVERLAYed solution set of n queens without attack extent information. Case *similarity* is defined as cases that have the most number of queens in common with the current problem. This matching is accomplished diagrammatically by ANDing the

current problem board (PEELed with *QueenGrayBoard*) with each of the stored solutions, counting all non-WHITE tesserae and retrieving those solutions with the highest count. A partial solution to the current problem has then been found; all queens in common can be exempted from further consideration as they are already in place. Case *adaptation* is the arrangement of those queens that are not yet in place to form a complete solution without disturbing the positions of the exempted queens. Lastly, case *indexing* is expedited by diagrammatically comparing a new candidate case with existing cases and rejecting duplicates.

An Example

Figure 5 details this case-based approach. 5a PEELs *QueenGrayBoard* from the current diagram resulting in a diagram that is gray only where queens are placed on the current diagram (denoted *QueenPlacement*). More formally:

QueenPlacement $\Leftarrow$ *CurrentBoard* - *QueenGrayBoard*

5b shows the process of ANDing *QueenPlacement* with each stored solution in the *CaseBase*, 5c, resulting in a set of diagrams, 5d, that each display their similarity with *QueenPlacement* via the number of gray tessera they have (denoted *SimilaritySet*). More formally:

SimilaritySet :=
MAP($\lambda(x)$ (*QueenPlacement* $\wedge$ x), *CaseBase*)

In this example, one case's queen placement matches six of the current diagram's, 5e. Such counting of certain valued tessera is accomplished diagrammatically as well (see [Anderson & McCartney 1995b]). This case is chosen, then, and the placement of the remaining two queens proceeds as described previously with the stipulation that the six matched queens are not to be moved.

Although this system cannot guarantee an optimal solution, it learns over time by storing previous solutions and, therefore, becomes progressively better at providing near optimal solutions at reasonable computational cost.

Related Research

Research in diagrammatic reasoning is just beginning to flourish after a long dormancy it experienced being virtually abandoned after a brief flirtation in the early days of AI (e.g. [Gelernter 1959; Evans 1962]). See, for instance, [Larkin & Simon 1987; Narayanan & Chandrasekaran 1991; Narayanan 1992; Chandrasekaran, Narayanan, & Iwasaki 1993; Narayanan 1993; Glasgow 1993; Glasgow, Narayanan, & Chandrasekaran 1995] for a representative sample of this work. We have previously proposed inter-diagrammatic reasoning as one way of using diagrammatic representations to solve problems

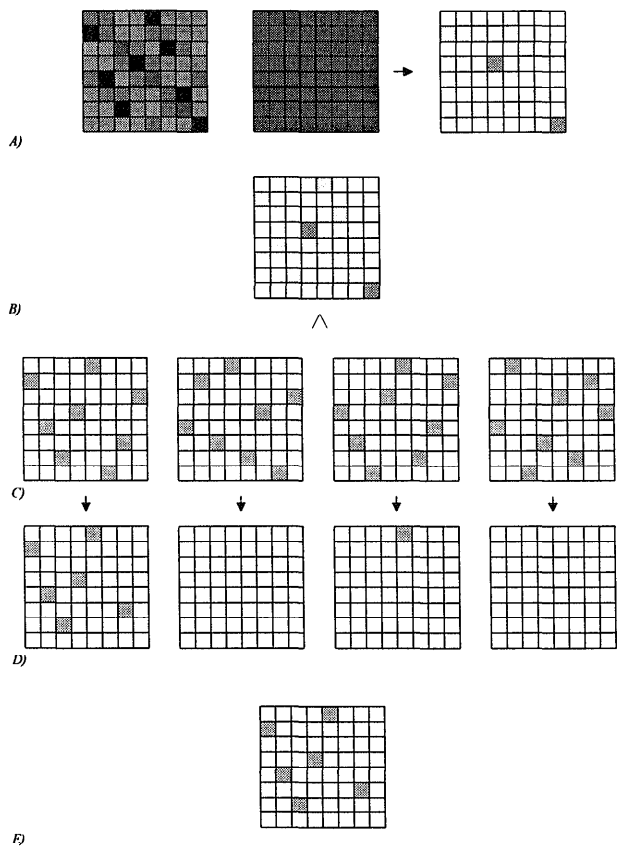

Figure 5: 8-queens example, case matching

[Anderson 1994; Anderson & McCartney 1995a; Anderson & McCartney 1995b]. The earliest work in diagrammatic reasoning can be considered the first example of inter-diagrammatic reasoning as well [Evans 1962]. Bieger and Glock [1985; 1986] and Willows and Houghton [1987] have done work in human use of sets of related diagrams.

Case-based reasoning has generated a good deal of interest: much work has been and is being done in this area. See [Kolodner 1993] for an overview. Interestingly, case-based reasoning has been previously used to increase efficiency in solving constraint satisfaction problems in [Purvis 1995].

[Narayanan & Chandrasekaran 1991] discuss what they term "visual cases" for diagrammatic spatial reasoning but we believe that we are the first to successfully integrate diagrammatic and case-based reasoning.

Conclusion

We have shown how a diagrammatic cased-based approach is useful in providing near optimal solutions to the

n-queens problem. It is straight forward to generalize our approach to involve objects of various sizes and extents. This is the first step towards applying this approach to spatial configuration problems and other domains.

We believe that, in general, a diagrammatic approach to case-based reasoning can help provide answers to the questions of case representation, similarity, indexing, and adaptation in many interesting real world domains. Further, a case-based reasoning approach to diagrammatic reasoning provides a framework that enables the effectiveness of diagrammatic operators to emerge.

References

Anderson, M. 1994. Reasoning with Diagram Sequences. In Proceedings of the Conference on Information-Oriented Approaches to Logic, Language and Computation (Fourth Conference on Situation Theory and its Applications).

Anderson, M. and McCartney, R. 1995a. Developing a Heuristic via Diagrammatic Reasoning. In Proceedings of the Tenth Annual ACM Symposium on Applied Computing.

Anderson, M. and McCartney, R. 1995b. Inter-diagrammatic Reasoning. In Proceedings of the 14th International Joint Conference on Artificial Intelligence.

Bieger, G. and Glock, M. 1985. The Information Content of Picture-Text Instructions. *The Journal of Experimental Education*, 53(2), 68-76.

Bieger, G. and Glock, M. 1986. Comprehending Spatial and Contextual Information in Picture-Text Instructions. *The Journal of Experimental Education*, 54(4), 181-188.

Chandrasekaran, B., Narayanan, N. and Iwasaki, Y. 1993. Reasoning with Diagrammatic Representations, *AI Magazine*, 14(2).

Evans, T. G. 1962. A Heuristic Program to Solve Geometry Analogy Problems. MIT AI Memo 46. (also in *Semantic Information Processing* as "A Program for the Solution of a Class of Geometric-analogy Intelligence-test Questions", 271-353, Minsky, M. L., ed. MIT Press, 1968).

Feigenbaum, E. A. and Feldman, J., eds. 1963. *Computers and Thought*, McGraw-Hill.

Gelernter, H. 1959. Realization of a Geometry Theorem Proving Machine. In Proceedings of an International Conference on Information Processing, 273-282. UNESCO House. (also in [Feigenbaum & Feldman 1963]).

Glasgow, J. 1993. The Imagery Debate Revisited: A Computational Perspective in [Narayanan 1993].

Glasgow, J., Narayanan, N., and Chandrasekaran, B. 1995. *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, AAAI Press.

Goel, A., Ali, K., Donnellan, M., de Silva Garza, A., and Callantine, T. 1994. Multistrategy Adaptive Path Planning, *IEEE Expert*, 9:6, 57-65.

Gu, J. 1989. Parallel Algorithms and Architectures for Very Fast AI Search, Ph.D. diss., University of Utah.

Kambhampati, S., and Hendler, J. 1992. A Validation Structure Based Theory of Plan Modification and Reuse. *Artificial Intelligence,* 55: 193-258.

Kolodner, J. L. 1993. *Case-based Reasoning*. Morgan Kaufmann, San Mateo.

Larkin, J. and Simon, H. 1987. Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science*, 11, 65-99.

McCartney, R. 1993. Episodic Cases and Real-time Performance in a Case-based Planning System. *Expert Systems with Applications,* 6:9-22.

Narayanan, N., ed. 1992. Working Notes of AAAI Spring Symposium on Reasoning with Diagrammatic Representations.

Narayanan, N., ed. 1993. Taking Issue/Forum: The Imagery Debate Revisited. *Computational Intelligence*, 9(4).

Narayanan, N. H. and Chandrasekaran, B. 1991. Reasoning Visually about Spatial Interactions. In Proceedings of the 12th International Joint Conference on Artificial Intelligence.

Purvis, L. 1995. Constraint Satisfaction Combined with Case-Based Reasoning for Assembly Sequence Planning, Technical Report CSE-TR-93-20, University of Connecticut.

Stone, H. S. and Stone, J. 1986. Efficient Search Techniques: an Empirical Study of the n-Queens Problem, Technical Report RC 12057, IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

Willows, D. and Houghton, H. 1987. *The Psychology of Illustration*, Springer-Verlag.