Formal Models for Imaginal Deduction

George W. Furnas
Bell Communicaitons Research

Abstract

Systems with inherently spatial primitives have advantages over traditional sentential ones for representing spatial structure. The question is how such representations might be used in reasoning. This paper explores a simple kind of *deductive* reasoning where picture-like entities, instead of symbol-strings, are given first-class status. It is based on a model of deduction as the composition of mappings between sets, and allows generalized notions of unification and binding, which in turn permit the definition of various formal, "imaginal" deduction systems. The axioms and rules of inference are all pictures or fundamentally picture-based, and are used to derive pictorial "theorems". After sketching the generalized theory needed, several possible strategies are mentioned, and a prototype, the BITPICT computation system, is described in some detail.

1. Introduction

The role of imagery in reasoning, human or artificial, remains unsettled. In cognitive psychology Shepard[1], Kosslyn[2], and others have argued that imaginal representations exist and certain geometric transformations of them (rotations, scanning, zooming) can be used to answer spatial questions. Explicitly addressing inference, Lindsay[3] has argued that images are important because of their ability to implement geometric constraints automatically and that this is what distinguishes imaginal reasoning from "calculus-plus-proof-procedure," deductive systems. A different view is offered here: that imaginal *deductive* inference is possible, and specially empowered by the ability of images to implement spatial constraints.

Consider a simple example from a conventional deduction system, shown in Figure 1(a). This might be part of a system which manipulates arithmetic expressions while preserving their arithmetic value. The figure shows Axiom 1, representing the commutative law of addition, and Axiom 2, representing the *right*-zero-additive-identity law. These two axioms are put together in a deductive chain to yield the indicated Theorem, the *left*-zero-additive identity. In this algebra example, the rules link expressions that are one-dimensional arrangements of symbols. The goal here is to have the rules link picture-like objects. A domain that yields a fairly intuitive example is in classical Euclidean geometry proofs, illustrated in Figure 1(b). Axiom 1, expressed in English, would be something like, "If one angle of a transversal of two parallel lines is designated as being of interest, the corresponding second angle may be designated as its equal." Here, however, the axiom is meant to stand on its own as a statement in a formal system, with the rule linking the left-hand picture to the right-hand picture, and no underlying representation of the form "L1 parallel L2 transversed by L3..." The second axiom asserts in pictures the congruence of opposite angles of intersecting lines. The goal of the work here is to describe such formal systems, wherein picture axioms support formal deductions as rigorously as algebraic ones shown in Figure 1(a). Though the geometry rules of Figure 1(b) suggest what it might be like to do imaginal deductive inference, the example is very incomplete. In particular, the deduction requires that Axiom 2 be applied to the picture on the right-hand side of Axiom 1. No machinery has been defined that lets Axiom 2, which has only two lines in its picture, apply to the right-hand side of Axiom 1, where there are three lines. Somehow the left-hand side of Axiom 2 must stand for more than what meets the eye, more than the exact two-line picture on the paper. In conventional logics, like those illustrated in Figure 1(a), the corresponding problem is handled by the well-articulated mechanisms of variables, quantification,

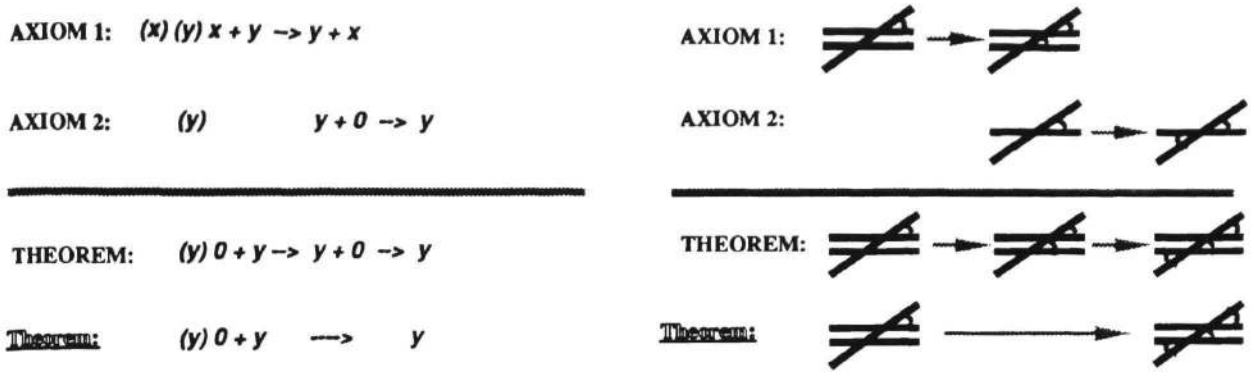

Figure 1. (a) A deductive system for algebra manipulation, showing the chaining of axioms for commutativity and zero-*right*-additive identity to get zero-*left*-additive identity. (b) A structurally analogous deduction in a hypothetical picture system to do Euclidean geometry proofs. In each case the first version of the resulting theorem preserves the intermediate link in the deductive chain, the second one omits it.

unification, bindings, etc. But where are the variables and quantifiers in the graphical axioms of Figure 1(b)? How can something like unification be accomplished? Since these mechanism have no direct analogs in the imaginal domain, it has been necessary to analyze what these mechanisms accomplish, to produce a list of generalized properties needed in a deductive formal system. These desired properties can

2. *Introduction to a General Theory of Deduction*

The difficulty for pictorial deduction in Figure 1(b) arises from the need to take the pictures in Axiom 2 other than literally. Consider the alternative, illustrated with the "Happy Face" system of Figure 2. Assume that *exactly* the picture appearing as the left-hand side [LHS] of Axiom 1 yields *exactly* the picture on the right-hand side [RHS] of Axiom 1, and *exactly* the picture appearing as the LHS of Axiom 2 yields *exactly* the picture on the RHS of Axiom 2. From these two axioms, one can conclude the "Happy Face Theorem", that *exactly* the LHS picture of Axiom 1 yields *exactly* the RHS picture of Axiom 2. Given a different frowning face, one of a different size or shape, even a single deductive step would be impossible, to say nothing of a chain of two.

2.1 The generative power of high-level rules

The literal deduction of the Happy Face Theorem is perfectly rigorous; it is also very impoverished. Many literal (*low-level*) rules would be needed to even approach some interesting behavior. What is required is a way to write *high-level rules*, i.e., a notational system whereby a whole set of low-level rules are specified at once. Such generative power is one of the important accomplishments of variables and quantifiers in standard sentential formalisms. Thus, the algebraic rule, $\forall x \forall y \;\; x + y \;\rightarrow\; y + x$, can really be considered as the name for a set of low-level arithmetic rules, $\{1 + 3 \rightarrow 3 + 1,\; 2 + 7 \rightarrow 7 + 2,\; 0 + 4 \rightarrow 4 + 0,\; \cdots\}$. Familiar machinery here generates the set of instance rules from the high-level algebraic expression using the Cartesian product of the instantiations of the universally quantified variables. For pictures, where it is not at all clear where or how to put in variables, a more general understanding of high-level rules is needed.

Low-level rules of a system associate ordered pairs ($a \rightarrow b$, $c \rightarrow d$, $e \rightarrow f$, $b \rightarrow f$, *etc.*) of "low-level" objects, for example, pairs of individual expressions of arithmetic or pairs of specific pictures of faces. The basic deductive inference is of the form, "given a and $a \rightarrow b$, conclude b," and enables deductive chaining of the low level rules, like,

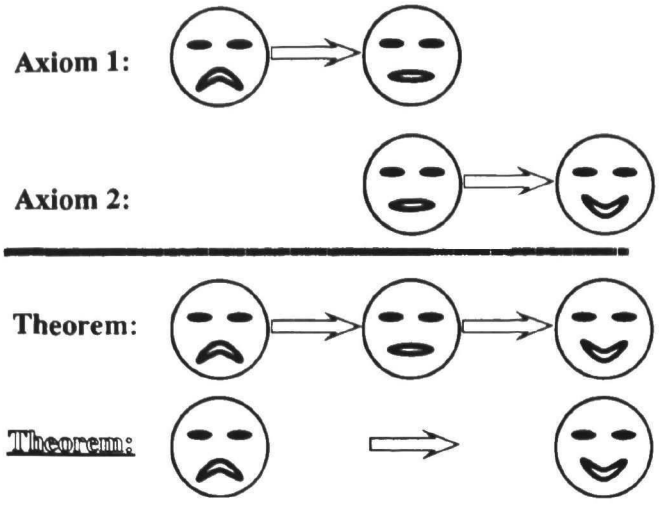

Figure 2. The "Happy Face Theorem". If the rules here are taken literally, i.e., as applying to only exact matches to the faces shown, a rigorous but impoverished deductive system results.

$$Given \quad a \rightarrow b \quad and \quad b \rightarrow f, \quad conclude \quad a \rightarrow f.$$

The happy-face deduction of Figure 2 , is an example of such a low-level chain.

A *high-level* rule specifies a whole set of such low level rules at once. This set of low-level associations defines a mapping between two sets of low-level objects. So a high-level rule is fundamentally a mapping,

$$f: A \rightarrow B, \quad i.e., \quad a_1 \rightarrow b_1, \; a_2 \rightarrow b_2, \; \cdots$$

between two sets (its Left- and Right-Hand sets), and as such may admit specification mechanisms other variables and quantifiers. Thus for a deductive system in a given domain of low-level objects (e.g., pictures), one needs devices for

(P1) the specification of sets of low-level objects, and
(P2) the specification of mappings between those sets,

that are suitable for the objects in the domain, and computationally tractable. Suitability, for example, is at issue for the straightforward application of the device of variables and quantifiers for the specification of sets of pictures. Computational tractability is required for the operations implicit in the application of the function, f , to a candidate object x. In particular, both the applicability of the function (the test of whether $x \in A$) and the actual result of the function must be computable with reasonable resources.

2.2 The deductive power of high-level rules

The generative power of high-level rules, i.e., their ability to specify a whole set of low-level rules at once, is only one important desideratum of a deductive system; a second is the ability to do high-level deductive chaining. A high-level deductive chain takes two high-level rules f and g and derives a third high-level rule h, subject to the condition that the low-level rules in h are all implied by those of f and g. That is, whole sets of low-level rules are deductively chained by chaining high-level rules. In this way, high level axioms are chained, as shown in Figure 1, to derive new "theorems". In terms of treating high-level rules as mappings, one wants to take two mappings, $f: A \rightarrow B$ and $g: C \rightarrow D$ and say as much as possible about applying one to the output of the other, i.e., forming a function composition. If the sets B and C are equal, the composition is straight forward,

$$Given \quad f: A \rightarrow B \quad and \quad g: C \rightarrow D \quad and \quad B = C, \quad conclude \quad g \circ f: A \rightarrow (B = C) \rightarrow D.$$

However if $B \neq C$ then the maximal composition is mediated by the intersection, $B \cap C$. Writing a bit

loosely (i.e., applying f and g to sets in the obvious way), we have

Given $f: A \rightarrow B$ *and* $g: C \rightarrow D$, *conclude* $f^{-1}(B \cap C) \rightarrow (B \cap C) \rightarrow g(B \cap C)$.

The computation of maximal partial composition may be thought of as having three critical steps. First is the computation of the intersection $B \cap C$. In standard deduction, this is accomplished by *unification* and yields information on variable bindings (e.g., y is bound to 0 in the algebra example of Figure 1(a)). The second and third steps map this intersection backwards by f^{-1} into A and forwards by g into D. In standard deduction, this is accomplished by substituting the variable bindings resulting from the unification, into the LHS of the first rule and the RHS of the second rule. Thus in any system where (P1) and (P2) above were defined, any mechanism for

(P3) the computation of $B \cap C$, *and*
(P4) the computation of $f^{-1}(B \cap C)$ *and* $g(B \cap C)$,
i.e., a way to test whether $a_i \in f^{-1}(B \cap C)$, *and if so to find* $d_i \in g(B \cap C)$,

would then enable high-level deductive chaining.

While the ideal is to have a system that supports (P3), one could settle for less. For example, if it were only possible to compute subsets of $B \cap C$, one could still make valid, though not maximal, high-level deductions. Note also that systems with no high-level deductive chaining could still be useful. For example, most current expert systems never prove theorems. They use only the generative power of high-level rules, taking some particular low-level instance from the world and cranking it through a rich system of high-level rules to deduce new low-level conclusions. Thus it is quite possible that a picture system that only satisfied (P1) and (P2) could still be quite important.

2.3 Candidate mechanisms for (P1)-(P4)

There are many strategies that can be used to satisfy various of the desiderata, particularly (P1) and (P2) (specifying sets of pictures and picture-to-picture mappings). A few of the more promising general approaches, fairly different in character from the use of quantified expressions of variables, are sketched below. Versions of the first two of these have been used in the the BITPICT system (see below), a version of the third in another system we are working on.

2.3.1 Instance-plus-transformation strategies. A set can be specified by a canonical instance a_0 and a set $T=\{t_i\}$ of permissible transformations of that instance. A rule can similarly be defined by giving a canonical low-level rule, $a_0 \rightarrow b_0$, and a set of transformations, T. The corresponding high-level rule is taken to map $t_i(a_0) \rightarrow t_i(b_0)$. For example, if T is the set of size rescaling transformations, and $a_0 \rightarrow b_0$ is the low-level, "sad-face" rule (Axiom 1) of Figure 2, then the result is a high level rule which maps the sad face at size i to the neutral face resized to size i. Similarly taking T to be the set of affine transformations of the plane[1] and the axioms of the Euclidean Geometry example as canonical rules, would yield a rigorous high-level picture deduction system (though still providing only part of the needed generality).

2.3.2 Feature-set strategies. One class of systems that fulfill all of (P1)-(P4) quite naturally arises if low-level objects can be characterized by sets of features. Then subsets of features can be used to specify sets of objects possessing the given features. There is a natural partial ordering of both object and feature sets by containment, forming a dual lattice structure upon which the intersections required for unification can be easily calculated, and for which there are natural notions of generalized binding.

1. Note that affine transformations are typically discussed in conventional sentential algebraic terms (e.g., by certain matrix multiplications), but that is not inherent. They can be represented and implemented otherwise (e.g., optically).

2.3.3 Grammar-based strategies. Another way to specify a set is as the language of a grammar. Thus, for example, most standard logics have a context-free string grammar that specifies the well-formed formulas of the formalism. The axioms map between certain specified subsets of the well-formed formulas. In particular, the left-hand side of many rules can be thought of as generated by a sub-grammar (e.g., "X + Y" as "expression1 : plus-operator : expression2"). A parser builds a parse-tree over the input to the left-hand side of the rule, and gets the object on the right side of the rule by manipulating the parse tree (e.g., flipping the two descendants of the "+" operator, to get "Y + X").

Such an exercise need not be restricted to string systems. There are various grammars for spatially structured objects, for example array and matrix grammars [4] [5] or picture grammars[6]. Collaborations with Karen Lochbaum led towards a system that specifies sets of simple pictures for the LHS of rules using context-free array grammars,with high-level rules operating by transformig the resulting spatially structured parse-trees.

3. The BITPICT deduction system

Several of the strategies just outlined have been combined in a particular system, the BITPICT system, and a prototype virtual machine now exists, running on a Symbolics Lisp Machine. It is a particularly simple formal system which, though not powerful enough to handle the Euclidean geometry example in full generality, is sufficient to give a better sense of what picture deduction systems might be like.

The architecture of the BITPICT system is similar to that of standard production systems. That is, like classical production systems [7] [8] [9], it has a set of rules that interact via a shared blackboard. Each rule has a left-hand side that constantly looks at the blackboard for a match. When a rule finds a match, its right-hand side directs how to change the contents of the blackboard. The change is made, and the process iterates. The rules are engineered so the evolution of the blackboard solves the intended problem.

Unlike standard production systems, where the structures on the board and in the rules are typically strings of symbols, in the BITPICT system they are simple picture fragments. Picture fragments are matched and replaced and problems are solved by the fragment-by-fragment evolution of the picture on the blackboard.

The universe of pictures considered by the BITPICT system are bitmaps, i.e., regular grids of picture elements (pixels) that are either black or white. A *bitpict* is a small such bitmap, and is meant to be the specifier for a whole set of other bitmaps, in this way performing a function analogous to quantified algebraic expressions specifying sets of arithmetic ones. A bitpict specifies the set of all bitmaps which contain that bitmap fragment somewhere. A *bitpict rule* is an ordered pair, LHS->RHS, of conformal bitpicts, and is taken to specify a mapping between the two corresponding sets of pictures. The mapping simply associates left- and right-hand pictures that differ only in that the LH bitpict has been replaced by the RH one. (See Figure 3.) There are two components to how bitpicts specify sets of pictures: (1) the use of a piece of a picture to specify the set of pictures containing that piece and (2) translation invariance, i.e., fact that the piece may occur at any vertical or horizontal position. The first is an example of a feature-set strategy: a pixel being black or white is a feature, and picture sets are specified by a spatially defined collection of such features. As mentioned earlier, notions of unification fall out easily from the resulting lattice. Let A and B be two bitpicts (but with translation disallowed, i.e. each at fixed positions). The *intersection* of their corresponding picture sets is a set whose specification is the pixel-wise *union* of A and B (or nil if they disagree at points of overlap). The second aspect, translation invariance, is an example of the instance-plus-transformation strategy: the i,j-th translation of the left-hand bitpict is mapped into the corresponding i,j-th translation of the right-hand bitpict. In fact, the BITPICT system can variously allow translations, rotations (0, 90, 180, or 270 degrees), and reflections. (The small crossed vertical and horizontal lines appearing like "+" sign, under the circle bitpict is an icon indicating that this particular rule holds for vertical and horizontal translations only.)

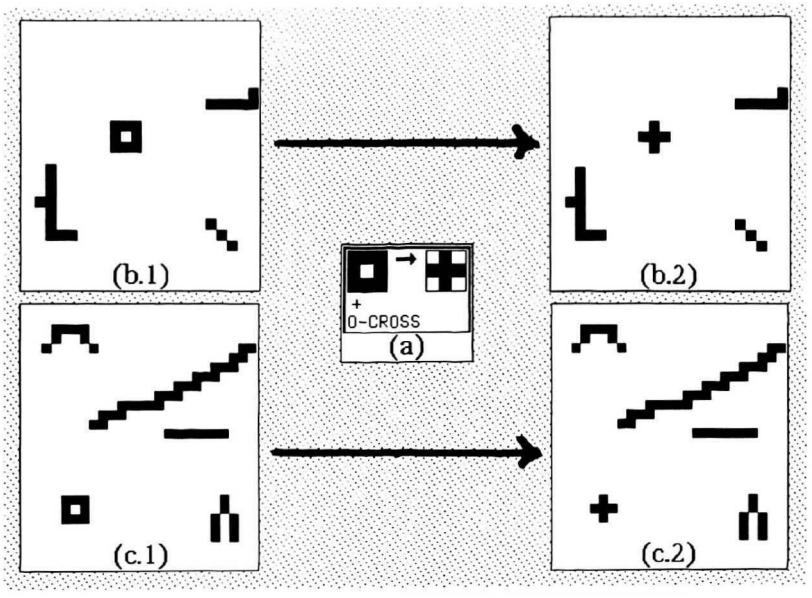

Figure 3. (a) A BITPICT rule mapping the 3x3 grid of pixels containing a circle to the 3x3 grid containing a cross. (b.1) and (c.1) show pictures in the left-hand set of the rule. (b.2) and (c.2) show the corresponding pictures in the right-hand set.

3.1 A BITPICT example: Counting the tangled forest

An example of the the BITPICT system solving a spatial problem with graphical deductions is presented in Figure 4. The problem, counting the disconnected components in a tangled forest of bifurcating trees, is not easy to represent in a sentential way, yet has a very natural solution with bitpicts.

Part (a) of the figure shows a sample tangled forest. The first set of rules, (b), simplify the problem, basically by nibbling back the tips of the branches, nibbling at straight, corner and "T" sections respectively. The rules neither create nor destroy connected components, and so each component is gradually reduced, (b.1), to a dot (b.2).

The next rules, (c), move the dots down, and when they hit the bottom of the window, to the right, as shown in (c.1), until they are all canonically aligned in the lower right corner, (c.2). Again the invariant of number of components is preserved, and the problem of counting trees has been reduced to counting these neatly arrayed dots.

The final rules, (d) count the aligned dots by converting each to a vertical bar (an "I") and from there to Roman numerals, with rules for simplifying (e.g., five I's to a V), maintaining alignment (e.g., shifting V's and X's right as needed), and sorting (e.g., VX to XV). The final result is the Roman numeral, XII, indicating the number of trees in the original tangled forest.

The critical computational point is that the evolving state of the blackboard is governed by the picture-to-picture mapping rules, with no need for underlying sentential representation.

4. Discussion

In addition to solving certain spatial problems and implementing graphical symbol manipulation algorithms (e.g., Roman numeral counting and arithmetic), the BITPICT system is also useful for auto-animation (e.g., cartoon figures that move themselves according to rules), for certain kinds of naive physics models (e.g.,Figure 5), and for the analysis of graphical computer interfaces.

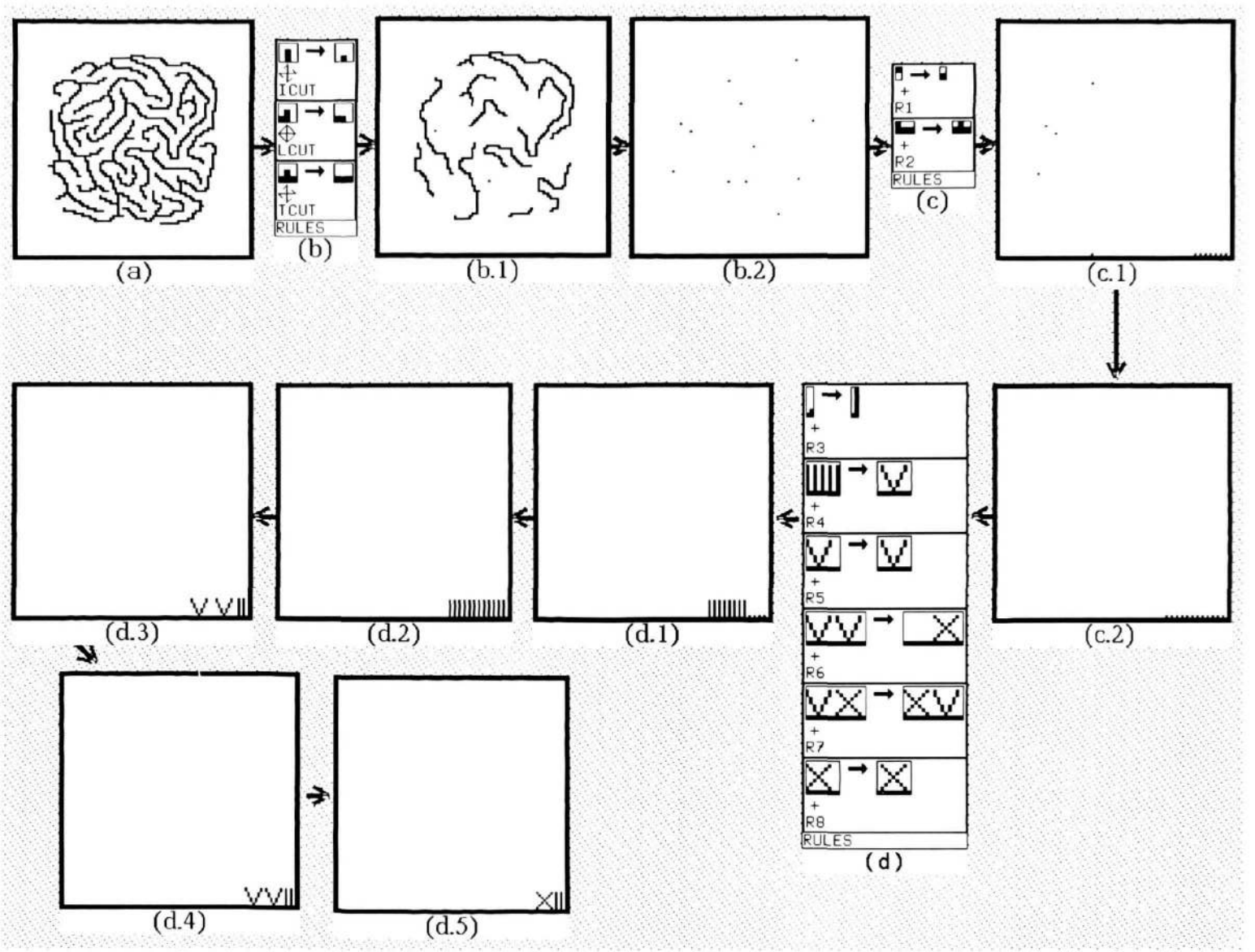

Figure 4. Counting the tangled forest. (a) A tangled forest of bifurcating trees. The rules, (b), nibble at the tips of straight, corner "L", and "T" sections, reducing each component, (b.1), to a dot (b.2). The rules, (c), move the dots down, and to the right, (c.1), into the lower right corner, (c.2). The rules, (d), count the aligned dots in Roman numerals, yielding the answer, XII.

The BITPICT system has a number of limitations, arising from its tie to the rectangular pixel grid, its current 2-D character, and the flatness of its representations (without the structural depth of grammar approaches), but it has been important in showing what imaginal deduction might be like.

REFERENCES

1. Shepard, R.N., and Metzler, J., Mental rotation of three-dimensional objects, *Science*, **171**, 1971, 701-703.
2. Kosslyn, S.M., *Image and Mind*, Cambridge, MA: Harvard University Press, 1980.
3. Lindsay, R.K., "Images and inference" *Cognition*, **29**, 1988, 229-250.
4. Siromoney, G., Siromoney,R. and Krithivasan, K., Picture languages with array rewriting rules, *Information and Control*, **22**, 1973, 447-470.
5. Rosenfeld, A., Array and web grammars: an overview. In A. Lindenmayer and G. Rozenberg (Eds.), *Automata, Languages, Development*, North-Holland, 1976, 517-529.
6. Lakin, F., Spatial Parsing for Visual Languages. In S.K.Chang, T.Ichikawa, and P.Ligomenides (Eds.), *Visual Languages*, New York: Plenum, 1986,35-85.
7. Post, E.L., Formal reductions of the general combinatorial decision problem, *American Journal of Mathematics*, **65**, 1943,197-268.
8. Newell, A., and Simon, H., *Human Problem Solving*, Englewood Cliffs, NJ: Prentice-Hall, 1972.
9. Waterman, D.A. and Hayes-Roth, F., *Pattern-Directed Inference Systems*, NewYork: Academic Press, 1978.

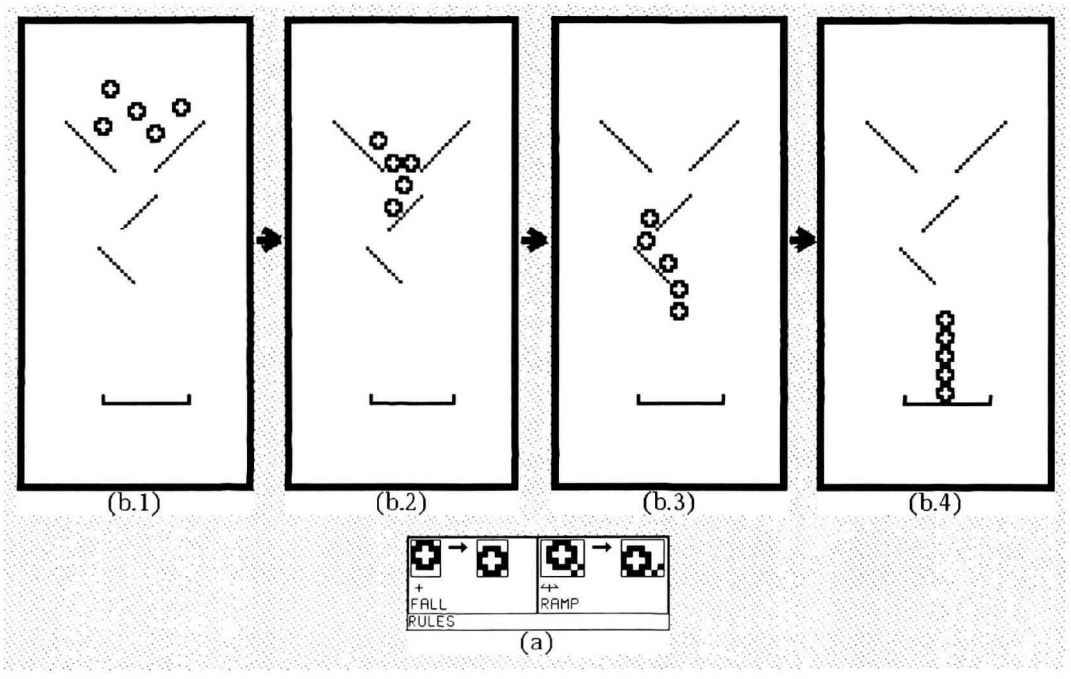

Figure 5. Balls and ramps - a naive physics model. (a) Rules for straight fall (translation invariant) and rolling down a piece of ramp (translation and reflection invariant). (b.1) An initial configuration. (b.2),(b.3) Samples from intermediate states in the deduction. (b.4) Final configuration.