VRule - A Rule Based Pixel Rewriting System for End-User Modeling

Rick L. Vinyard, Jr.

Dept. of Computer Science, New Mexico State University, Las Cruces NM, USA
rick.vinyard@gmail.com

Abstract. Numerous approaches to pixel rewriting systems (PRSs) have been investigated since Furnas' Bitpict. VRule is a new rule-based pixel rewriting framework designed to support investigation of new formalisms for end-user modeling with PRSs, and is implemented as a Pattern Directed Inference System (PDIS) with specialized classes of antecedent and consequent Pattern Directed Modules (PDM) components. This overview briefly examines VRule's key elements resulting from exploration of the architectural aspects of rule-based engines to include matching, rule resolution, and execution in a spatial environment, as well as numerous new PDM formalisms. Examples representing classes of modeling domains have been developed to include Turing machines, 1D/2D/3D cellular automata, a game of Fox and Hounds, Langton's Ant (turmites), a climatological gas model originally written in C#, and a virological infection model originally written in Python.

Keywords: pixel · rewriting · modeling · end-user · inference engine.

1 Diagrammatic Reasoning and Pixel Rewriting Systems

The expressiveness of diagrammatic reasoning systems have been the subject of many approaches created to explore not only whether reasoning can be performed with images and diagrams, but also what spatial relationships can be leveraged for programming environments. Although ancient mathematicians in China, India, and Greece [19] used diagrams to support and explain geometric theorems, later debate has focused on the role of diagrams in reasoning with some taking the position that diagrams have no role in formal reasoning [30] and others arguing that diagrams can be used in formal reasoning and that diagrams have both advantages and weaknesses distinct from formal sentential methods [3, 27, 9, 18, 29].

Early attempts to leverage diagrams in computer based reasoning systems include Gelernter's Geometry Machine [13, 12] which demonstrated the first use of diagrams in computer reasoning and utilized backward chaining logic and diagrams to prune the tree, and also used diagrams as to accept heuristic facts as true from the diagrammatic representation which used a coordinate representation system.

Pygmalian [28] was another early system created in 1975 to explore writing computer programs with diagrams and images. Pygmalion was an early programming by demonstration system that employed icons instead of variables, data structures, and functions. A Pygmalion program was developed by specifying an initial image, and through a series of edits the operations would be recorded for later execution. Other systems such a Tinker [24] and NetLogo [32] rely heavily on text based languages, and augment the programming environment with spatial aspects such as mouse events or image annotations, and differ from those systems that explicitly use diagrammatic properties as part of the knowledge representation system.

Lindsay proposed an alternative approach to the frame problem utilizing images as inference-making representations [25] and proposed the idea of using images to draw inferences without explicitly listing rules of deduction following Haugeland's observation that "the beauty of images is that (spatial) side effects take care of themselves[16]." Lindsay further claimed that "visual images possess properties... not possessed by *deductive* propositional representation," and that a representation system could "make inferences without the explicit rules of deduction but simply by virtue of the properties of the knowledge representation system alone."

Extending upon this idea Barwise and Etchemendy used diagrammatic and sentential representations in the same semantic framework when they demonstrated Hyperproofs. However, Hyperproofs separated visual and linguistic forms into distinct rules of inference that operated on each form of representation, and provided for rules to allow information to move between the representation forms [2, 4].

As noted by Blackwell[5] graphical rewrite systems are "an influential family of development environments for end-users." Furnas' BitPict [6, 8] was the first diagrammatic reasoning system to demonstrate a completely graphical image rewriting system where reasoning occurred from image to image through pixel rewriting without any sentential representation, and demonstrated the ability to perform image to image deduction with a key example demonstrating the solution to a spatial problem of counting disconnected components in a tangled forest. In this example Furnas demonstrated a set of three graphical rules capable of reducing the bifurcating trees to a set of distinct pixels representing the disconnected components, and used an additional set of rules to represent the number of disconnected components in Roman numerals, and as pointed out by Furnas "the critical computation point is that the evolving state of the blackboard is governed by picture-to-picture mapping rules, with no need for underlying sentential representation."

Visulan [33, 34] further extended the groundbreaking concept of image-to-image deduction with the introduction of logical conjunction (extended pairs) and disjunction (abstract/concrete images) concepts and also demonstrated Turing completeness.

Later work on BitPict [7, 10] examined the used of intermediate constructions to facilitate problem solving in PRS environments. Employing an approach which uses intermediate diagrams Furnas, et. al. demonstrated the ability of PRSs to not only discover the shortest path between two points around an obstacle, but also demonstrated a limited case of digit recognition and other forms of shape manipulation.

SOAR+SVI [22, 23] extended the SOAR [21, 20] architecture and further built upon the PRS framework of BitPict employing an object model and a visual buffer, combining algebraic image processing with a pixel rewriting system, introducing a number of new formalisms including wildcard antecedents, sentential rule set data descriptions, and attention windows.

2 VRule Architecture and Work Completed

VRule has been designed to support investigation into pixel rewriting environments and has been informed by many of the aforementioned graphical systems. Like many of its predecessors, the goal of VRule is to explore aspects of the inherent properties of diagrammatic representation systems with a particular focus on those properties that can facilitate end-user programming. VRule has been implemented as a Pattern Directed Inference System (PDIS) where a "program is better viewed as a loosely organized collection of pattern-directed modules"[17] and employs a familiar *match* $\rightarrow$ *resolve* $\rightarrow$ *act* cycle. Building upon concepts introduced by BitPict, VRule's workspaces and rules are sets of PNG (Portable Network Graphics) images, and is implemented in Java/CUDA with an object oriented design to allow rapid development of exploratory engine components and PDMs. Output of the intermediate computation is either raw PNG images or scaled images with grid lines and other annotations as seen in the latter images generated directly by VRule.

The similarity between raster based pixel rewriting systems and raster based cellular automata (CAs) was noted by Furnas and Qu [11], but they also distinguish the two approaches as "classical CAs are, however, often considered hard to program to achieve desired (as opposed to explore emergent) results" and conclude "the patterns in rules of PRSs are often explicitly relevant to the desired resulting functionality, we find them easier to program than CAs."

VRule embraces both pixel rewriting systems and CAs and introduces new formalisms for PRSs that enable the programming of CAs and agent based models, and improves upon the existing notational forms to demonstrate the potential of image based computation for some forms of end-user modeling. Also examined is the natural alignment between the parallelized architecture of *General-Purpose computing on Graphics Processing Units* (GPGPU) environments such as CUDA and OpenCL, and the execution engines of PRSs.

Adoption of the PDIS architecture has significantly influenced the design of VRule's extensible framework which can be broadly divided into the categories of extending the image-to-image deductive reasoning framework of BitPict, Visulan, et. al. with rule-based engine components or PDM components. To facilitate investigation components such as PDM antecedents or consequents can be extended to create new formalisms that leverage diagrammatic properties such as spatial relationships or color attributes, and can be used interchangeably with other extensions. Examples of VRule's engine innovations include a new spatial resolver capable of using a multitude of dimensional strategies when resolving antecedent matches.

Another VRule resolver variant provides Visulan's extended pairs with a total ordering not only between rules, but also between spatial matches and introduces intra-rule matches with independent spatial priority. As observed by Furnas, et. al. [7] "[n]ote that rules here use only local, 2D rewrites on rectilinear grids of pixels. There is a large space of related systems as yet unexplored, including variants with multiple layers and non-local pixel rewrites." VRule's Key/Value workspace layers embody one form of exploration of this space allowing rules to match and act against specific layers. Other areas of examination include boundary matches and hierarchical rules for specification of intra-rule strategies.

While Anderson's Inter-Diagrammatic Reasoning [1] evolved a macro level approach to BitPict, VRule's PDM approach takes a diametrical evaluation of the rules to the micro level of the pixels involved during the match and act phases while preserving BitPict's image to image computation. New PDM formalisms have been developed with the goal of supporting paradigms common to classes of modeling domains, and to demonstrate the utility of these new antecedent and consequent formalisms examples representative of domain classes have been developed.

In addition to spatial aspects, VRule has also been designed to explore the use of color, hue, saturation, and transparency in a diagrammatic representation system, and new forms of PDMs introduced by VRule include transparent pixel PDMs that can be used to specify punctured lattices such as CA Moore neighborhoods. Although SOAR+SVI demonstrated wildcard matching in the antecedent, VRule extends the transparency concept to the consequent. Another new form of PDM are VRule's anchors that allow the programmer to specify transforms between the antecedent match and the consequent application on the same or even different layers.

Other new VRule PDMs include null consequents, negation antecedents, border matching, and extended affine transforms. Non-image PDMs have also been examined to include event based antecedents, halt consequents, and snapshot consequents. Event based antecedents provide a mechanism for user interaction with the executing model, while snapshots allow programming specific points in the model evolution that can be used for further numerical analysis beyond the scope of VRule.

Moving beyond purely diagrammatic forms, new stochastic elements have been introduced as well that preserve BitPict's original goal of image to image computation while providing stochasticism for modeling domains. Trilobite notation, inspired by the Schrodinger notation of Spider diagrams[15], was developed to specify concepts such as "no more than n Moore neighbors," or "at least n Von Neumann neighbors with range less than r" used to program the SEIR model, various CAs, and Conway's Game of Life.

2.1 Implementing the First Turing Machine in VRule

Turing's first a-machine described in his pioneering work [31] is shown in figure 1a, and results in computation of the sequence 010101... with each digit separated by a blank. This example was selected as it demonstrates a direct comparison to the differing approaches of Visulan and VRule, and the benefit of VRule's transparent pixel PDMs.

This machine can be implemented in VRule using antecedents and consequents with transparent pixels. The transparent antecedent pixels equate to Turing's actions when no symbol is scanned, and the transparent consequent pixels allow rules to be created that consider only the current tuple being created. Rather than use symbols for the states, colors will represent the various states using a relation of {(b, red), (c, green), (e, blue), (f, magenta)}. Turing's alphabet will be related to colors using a relation of {(*blank*, yellow), (0, white), (1, black)}. The movement of the head is spatially represented in the consequent relative to the antecedent, as is the next state. Thus, each rule contains all information of the corresponding 5-tuple. Figure 1b demonstrates the four rules needed to program Turing's first machine using VRule's transparency rules for antecedents and consequents.

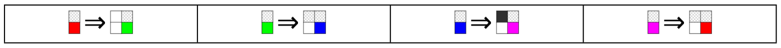
This approach is significantly different from either that of Visulan or BitPict in that the rules of those systems needed to encode not only for the current symbol under the read/write head, but also for the symbol under the read/write head after it moved. Using VRule's transparent pixel consequents allows the rules and operation to be specified directly in accordance with Turing's original design. Figure 1c depicts the first twelve cycles of Turing's first a-machine executing in VRule.

2.2 Further Examples

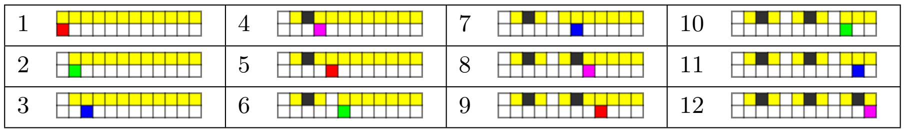
Figure 2 contains two examples of VRule programs in different modeling domains created with VRule's new formalisms, selected for their computational characteristics in the domain. Depicted in figure 2a are the bubble toppling rules of the Model of Ebullition and Gas storAge (MEGA) [26] (originally in C#) reimplemented in VRule, and uses VRule's new border PDMs to control the cycle counter and introduction of gas bubbles to the executing model, as well as stochastic PDMs to control random left/right toppling. Figure 2b demonstrates a Human Immunodeficiency Virus (HIV) SEIR cellular model [14] (originally in Python) progressing from a healthy state, to an infected state, to a delayed immune response over seven weeks. Also depicted is a trilobite antecedent specifying a Moore neighborhood that excludes the central cell and specifies a match of at least one, but no more than two cells. The second image of the SEIR model demonstrates a random initial population generated by VRule's stochastic PDMs. Stochastic PDMs are also employed in determining cellular death and

Current State	Scanned Symbol	Print Symbol	Move Head	Next State
`b` 🟥	`None` 🟨	`0` ⬜	`R`	`c` 🟩
`c` 🟩	`None` 🟨	🟨	`R`	`e` 🟦
`e` 🟦	`None` 🟨	`1` ⬛	`R`	`f` 🟪
`f` 🟪	`None` 🟨	🟨	`R`	`b` 🟥

(a) Turing's first *a*-machine with color mappings



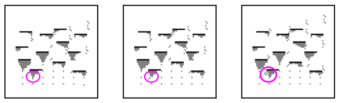
(b) Four rules implementing Turing's first machine in VRule



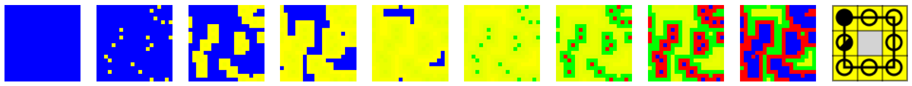
(c) Execution of Turing's first machine in VRule - First 12 cycles

Fig. 1: Turing's First *a*-Machine

dead cell replacement in accordance with the original Python model. This SEIR example employs many of VRule's new formalisms such as trilobite notation to specify a Moore neighborhood for candidate cellular infections, layers to control weekly cellular aging, and boundary PDMs to control the model cycle.



(a) MEGA bubble toppling rules and Wolfram CA 82



(b) HIV SEIR model

Fig. 2: Examples of programs executing in VRule

3 Future Work

Initial work has been done upon tessellations of hexagons to support macroscopic Navier-Stokes fluid flows. Another current area of investigation includes antecedent color component thresholds and consequent color-based operators such as dodge, burn, etc. It is anticipated that these forms may prove useful on workspaces derived from photographic images to perform tasks such as annealing hot pixels or edge detection. Implementation of the matching antecedents and consequent actions as OpenCL kernels has also been investigated with promising early results warranting further examination to quantify the performance benefits related to executing the match and action phases in a parallelized graphical environment.

While VRule has been demonstrated to be Turing complete, and is capable of supporting many modeling domains including cellular automata, climate change, and epidemiology there remains a multitude of possibilities to extend VRule with additional formalisms supporting new and emerging modeling approaches.

References

1. Anderson, M., Furnas, G.: Relating two image-based diagrammatic reasoning architectures. In: International Conference on Theory and Application of Diagrams. pp. 128–143. Springer (2010)
2. Barwise, J., Etchemendy, J.: Information, infons, and inference. In: Cooper, R., Mukai, K., Perry, J. (eds.) Situation Theory and its Applications. vol. I, pp. 33–78. University of Chicago Press (1990)
3. Barwise, J., Etchemendy, J.: Visual information and valid reasoning. In: Visualization in teaching and learning mathematics. pp. 9–24. Mathematical Association of America (1991)
4. Barwise, J., Etchemendy, J.: Hyperproof: Logical reasoning with diagrams. Tech. rep., AAAI Spring Symposium Technical Report SS-92-02 (1992)
5. Blackwell, A.F.: Psychological issues in end-user programming. In: End user development, pp. 9–30. Springer (2006)
6. Furnas, G.: Formal models for imaginal deduction. In: Erlbaum, L. (ed.) Proceedings of the Twelfth Annual Conference of the Cognitive Science Society. pp. 622–669 (Jul 1990)
7. Furnas, G., Qu, Y., Shrivastava, S., Peters, G.: The use of intermediate graphical constructions in problem solving with dynamic, pixel-level diagrams. In: International Conference on Theory and Application of Diagrams. pp. 314–329. Springer (2000)
8. Furnas, G.W.: New graphical reasoning models for understanding graphical interfaces. In: CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 71–78. ACM Press, New York, NY, USA (1991)
9. Furnas, G.W.: Reasoning with diagrams only. In: AAAI Spring Symposium Series, Symposium: Reasoning with Diagrammatic Representations. pp. 118–123 (1992)
10. Furnas, G.W., Qu, Y.: Shape manipulation using pixel rewrites. Proc. Visual Computing 2002 pp. 630–639 (2002)
11. Furnas, G.W., Qu, Y.: Using pixel rewrites for shape-rich interaction. In: CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 369–376. ACM Press, New York, NY, USA (2003)
12. Gelernter, H., Hansen, J.R., Loveland, D.W.: Empirical explorations of the geometry theorem machine. In: Papers Presented at the May 3-5, 1960, Western Joint IRE-AIEE-ACM Computer Conference. pp. 143–149. IRE-AIEE-ACM '60 (Western), ACM, New York, NY, USA (1960). https://doi.org/10.1145/1460361.1460381

13. Gelernter, H.L.: Realization of a geometry theorem proving machine. In: IFIP congress. pp. 273–281 (1959)
14. Giabbanelli, P.J., Freeman, C., Devita, J.A., Rosso, N., Brumme, Z.L.: Mechanisms for cell-to-cell and cell-free spread of hiv-1 in cellular automata models. In: Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. pp. 103–114. ACM (2019)
15. Gil, J., Howse, J., Kent, S.: Formalizing spider diagrams. In: Proceedings 1999 IEEE Symposium on Visual Languages. pp. 130–137. IEEE (1999)
16. Haugeland, J.: Artificial Intelligence: The Very Idea. Massachusetts Institute of Technology, USA (1985)
17. Hayes-Roth, F., Waterman, D., Lenat, D.B.: Principles of pattern-directed inference systems. In: Pattern-directed inference systems, pp. 577–601. Elsevier (1978)
18. Jamnik, M.: Mathematical reasoning with diagrams. University of Chicago Press (2001)
19. Kulpa, Z.: Diagrammatic representation and reasoning. In: Machine GRAPHICS & VISION. vol. 3, pp. 411–418. Polish Academy of Sciences (1994)
20. Laird, J.E.: The Soar cognitive architecture. MIT press (2012)
21. Laird, J.E., Newell, A., Rosenbloom, P.S.: Soar: An architecture for general intelligence. Artificial intelligence **33**(1), 1–64 (1987)
22. Lathrop, S.D.: Extending cognitive architectures with spatial and visual imagery mechanisms. Computer science and engineering, The University of Michigan, Ann Arbor (2008)
23. Lathrop, S.D., Laird, J.E.: Extending cognitive architectures with mental imagery. In: Proceedings of the 2nd Conference on Artificial General Intelligence (2009). Atlantis Press (2009)
24. Lieberman, H.: Tinker: A programming by demonstration system for beginning programmers. Watch what I do: programming by demonstration **1**, 49–64 (1993)
25. Lindsay, R.K.: Images and inference. Cognition **29**(3), 229–250 (1988)
26. Ramirez, J.A., Baird, A.J., Coulthard, T.J., Waddington, J.M.: Testing a simple model of gas bubble dynamics in porous media. Water Resources Research **51**(2), 1036–1049 (2015)
27. Shin, S.J.: The logical status of diagrams. Cambridge University Press (1994)
28. Smith, D.C.: Pygmalion: An executable electronic blackboard. In: Watch what I do. pp. 19–48. MIT Press (1993)
29. Stapleton, G., Jamnik, M., Shimojima, A.: What makes an effective representation of information: A formal account of observational advantages. Journal of Logic, Language and Information **26**(2), 143–177 (2017)
30. Tennant, N.: The withering away of formal semantics? Mind & language **1**(4), 302–318 (1986)
31. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London mathematical society **2**(1), 230–265 (1937)
32. Wilensky, U.: Netlogo (1999), http://ccl.northwestern.edu/netlogo/, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
33. Yamamoto, K.: Visulan: A visual programming language for self-changing bitmap. In: Proceedings of International Conference on Visual Information Systems. pp. 88–96. Springer (1996)
34. Yamamoto, K.: A module system for bitmap-based languages [translated from japanese]. Transactions of the Information Processing Society of Japan **38**(12), 2544–2551 (1997)