Dissertation Proposal

Constraints for Interactive Layout

Greg Badros
Department of Computer Science and Engineering
University of Washington

16 October 1998

1 Background and Introduction

From the inception of graphical user interfaces, systems have tried to use constraints to maintain relationships among on-screen entities [Sut63]. Constraints permit the designers or users of a system to express what they wish to hold true, rather than detail how to maintain the invariant procedurally. This declarative specification of desired relationships is the fundamental strength of using constraints. Constraints are especially natural in managing user interfaces [BD86, MMM+97, MGD+90], drawing [Sut63, HN94], and other applications involving geometrical layout [CSI86, HM96, RMS97].

Since any constraint system is limited by the expressibility and performance of the underlying constraint solver, increasing the power of solvers is a popular research area. Early systems embedded solvers based on iterative numerical techniques and local propagation which dealt only with acyclic constraints [Sut63, Bor79]. More sophisticated constraint systems (e.g., those with cycles, inequalities, or simultaneous linear equalities) require correspondingly more advanced techniques for finding solutions. Though batch numerical techniques are applicable for solving constraint systems in isolation, the demand for interactive use of constraints required algorithmic improvements to increase solver efficiency by exploiting previous solutions [FBM89, BFB98, BMSX97].

There is substantial tension between the expressiveness of the constraints a solver can manage and the efficiency in finding a solution. (For example, permitting arbitrary nonlinear constraints prohibitively increases the computational complexity of the solver algorithm [VHM95].) Because of this fragile balance between expressiveness and performance, system implementors typically hand-tune the tradeoff for each specific application involving constraints—reuse of constraint solvers in interactive applications has been relatively limited. The implementation challenge has certainly hindered widespread acceptance of constraint solving technology.

After over thirty years of research, general-purpose constraint technology remains largely unsuccessful commercially in interactive applications. Numerous difficulties have contributed

to this lack of success. User interfaces for specifying and managing constraints are often immature or incomplete, failing to hide the complexities and idiosyncrasies of the underlying solver from the user. To be most useful, constraints must be expressed at an appropriate level of abstraction for the application and user's sophistication. The benefit of the constraint abstraction may disappear if the solving engine must be understood in detail to use a system effectively. Debugging constraints at a reasonable level of abstraction is particularly challenging—with existing systems, unexpected solutions can often be difficult to explain, and few tools, if any, are provided to aid the user's understanding.

As hinted earlier, another problem relates to domain-specific needs of individual constraint-based applications. When an application needs to express a constraint that exceeds the embedded solver's abilities, the temptation is to find an ad-hoc means of providing just that additional capability. This complicates the solver with edge cases instead of extending the solving infrastructure more generally.

I intend to attack these problems with current constraint technology within the context of interactive layout applications. The next section discusses my plans for using constraints in two such applications: 1) a web browser and authoring environment for creating and rendering a document containing text, figures, tables, and other entities; and 2) a window manager for arranging and manipulating application windows on a user's desktop. Both of these applications will build on existing solver technology, and in Section 3 I discuss my plan for creating a generalized and extensible constraint solver infrastructure to permit easier addition of the domain-specific relationships. Section 4 briefly summarizes my expected contributions and Appendix A details a time schedule.

2 Interactive Layout

Interactive layout is rich with beneficial possible applications of constraints. For both web browsers and window managers, constraints in some form are already in use; my thesis will push the envelope of what is expressed within the solver while investigating what interface metaphors and paradigms best aid the user. Since linear equalities and inequalities are especially useful for object-layout applications, the Cassowary [BB98] constraint solver will be used as a starting point for each application.

2.1 Constraints and Style Sheets for the Web

The World Wide Web has exploded in popularity in the last seven years. However, publishing for the web still suffers from some significant problems which stem from the lack of separation of semantic information and visual formatting. Sites often maintain multiple versions of the same document for various different browsers, screen size, or bandwidth limitations. The duplication of content complicates a web site's structure [FFK+98]. Additionally, not all users' needs will be met by the limited set of chosen presentation formats—differently-abled users may be ignored. Page designers regularly mangle the semantic content of a document to achieve a desired visual layout. The splicing of text fragments into tables or images

complicates tools' use of the contained information (e.g., search engines and other agents).

Numerous attempts are underway to separate the desired visual structure of documents into reusable style sheets. Cascading style sheets, as specified by the W3 Consortium, are a popular approach to alleviating this problem [LB96, BLLJ98]. I will extend CSS1 (and later CSS2) to permit expressing general constraints about the layout and other properties of the page. In collaboration with Håkon Wium Lie and Bert Bos of the W3 Consortium, some work has been done to enhance CSS2 with constraints for box layout [Mic98]. I will build on that specification and also on the work done by Alan Borning and others during his sabbatical at Monash University in Melbourne, Australia (where I will be visiting next quarter) [BLM97]. To permit authors to easily use the constraint-enhanced CSS model, I will extend a web-page authoring environment with a user interface for managing and debugging the constraints.

Document designers will be able to selectively specify only the relative importance of the various aspects of the visual appearance of a page. Since the browser recognizes explicitly-stated desired relationships for a given page it will have the freedom to generate layouts that may be more useful to a wider range of users. Visually-challenged users will be able to specify lower bounds on font sizes to keep text readable. The possibilities are endless–web page authors could even link the page background color to the age of the document, so the document yellows as it ages.

To permit interactive viewing and reformatting of the delivered pages with their visual formatting specifications, I will embed the same solver in Amaya, the W3 Consortium's test-bed WWW browser [Con98]. Additionally, I will permit the end-user to specify desired layout preferences using the same expressiveness of constraints. Though the competing constraints could result in an over-constrained system, the theory of constraint hierarchies [BMMW89] provides an elegant solution whereby the solver can be seen as an arbiter of the conflicting preferences between the viewer and the author of page.

By providing a general constraints-based solution instead of a specific set of pre-defined possibilities, the resulting system will be far more flexible and powerful than if CSS were repeatedly extended with more and more ad-hoc capabilities. Ideally, our collaborations with the W3 Consortium will result in adding constraints to the CSS specification so the "C" may come to stand for "Constraints."

2.2 Scheme Constraints Window Manager

In some windowing systems, the arrangement of windows on screen is managed by a distinguished user-level application called a window manager. Numerous window managers exist, each enforcing different window layout policies. Various relationships among application windows are easily expressed using linear equality and inequality constraints. For example, "this window should be above that window," or "these two windows' heights should sum to the display height."

Window managers provide an application with especially dynamic (and thus challenging) interactive layout needs: users constantly add, remove, rearrange and otherwise manipulate windows to help them work better. Convenient window layout improves users' overall performance, as long as the placement of windows is not time consuming [KS96]. Window layout

and the other visual properties managed by a window manager are becoming more and more related to web page layout: dynamic applets and addition, removal, and resizing of web-page objects will become commonplace as web browsers and our desktop environment continue to merge. Window managers may become the web browser of the future, or web browsers may turn into desktop managers as the boundaries between the two applications fade away.

I have embedded the Cassowary constraint solver in an original window manager called Scwm—the Scheme Constraints Window Manager [BS]. Users have the full generality of the constraint solver for specifying the relationships among windows and a rich programmable scheme environment for additional arbitrary computation. I will use Scwm as a test-platform for studying user interfaces for manipulation and visualization of constraints. Additionally, I will investigate ways to enable users to debug constraint systems interactively, without exposing the computational details of the underlying solver.

Numerous features of existing window managers can also be expressed as constraints. For example "sticky" windows which are visible on all viewports of a virtual display, and "always-on-top" windows which remain unoccluded by (in front of) other regular windows. These constraints will be satisfied via the local-propagation subsolver. The resulting constraint system will be expressive enough to permit very flexible, automatic window layout, and subsume numerous ad-hoc constraint-like features already provided. Throughout development, I will perform user studies to better understand how users layout their windows and to learn how they employ the constraint system and its user interface to aid them.

3 Extensible Constraint Solving

To encourage wider use of constraint-solving technology, and improve the quality and reusability of constraint solving toolkits, I will create an extensible constraint solving architecture (or perhaps extend or enhance constraint handling rules [Fru98]). My approach will be to first integrate the solving technologies of Cassowary—the efficient, incremental simplex-based solver—with Ultraviolet [BFB98]—a local-propagation based arbitrary-domain solver. This will extend Ultraviolet's hybrid nature and permit other subsolvers.

Another important extension of the underlying solver is to add support for expressing disjunctions. I will extend the solver architecture with support for CLP-style [JL87] backtracking to permit satisfying systems with disjunctions.

Finally, the experience of providing two useful extensions to the solver architecture will provide insight for the last important goal for the infrastructure: specification of a clean API to permit other subsolvers to be integrated with minimal non-essential affect to the performance and complexity of the resulting combined solver. The two practical layout applications described in Section 2 will exercise my constraint solver architecture substantially with their demanding needs for sophisticated and varied relationships.

4 Expected contributions

To summarize, the primary expected contributions of my thesis are:

- An extension to the cascading style sheets specification and an implementation of a web browser and authoring environment for a new constraint style sheets standard supporting generalized constraints that are dynamically maintained;
- interactive graphical user interfaces for manipulation, visualization, and debugging of advanced constraint systems for web page rendering and window manager layout;
- user study results reporting on the impact of constraints in enhancing web browsing and document authoring, and automatic layout and manipulation of application windows by a window manager; and
- an extensible architecture for constraint solving systems with a fully-specified API for user extension and standard subsolvers for the architecture which simultaneously solve linear equalities, inequalities, arbitrary domain constraints, and disjunctions.

By framing the research problems in two real-world applications, I expect the contributions to be of immediate practical value. Web page authors will have the necessary tools to separate the visual layout desires for their content from the semantic information. Web site maintenance will be simplified, web surfers will have more freedom in overriding formatting that hinders their capability to understand or navigate pages, and automated tools will reason more reliably about the pages they manage. As the desktop continues to merge with the global web-space, we will be prepared for the increasingly dynamic layout needs of integrated "everything browsers."

A Time Schedule

My proposed thesis work is already underway. I am currently supporting C++ and Java implementations of the Cassowary constraint solving toolkit. Work is underway to embed the solver in the Amaya web browser. The below table lists various activities and my anticipated completion dates.

Milestone	**Anticipated completion**
Initial design of generalized constraint extensions for CSS1	December 1998
Basic constraint solver embedded in Amaya	January 1999
Cassowary/Ultraviolet solver integration	April 1999
User interface for web authoring environment, browser	June 1999
Preliminary user studies on browser	July 1999
Extensions to solver to support disjunctions	August 1999
Embedding of advanced constraint solver in Amaya	September 1999
Revised constraint extensions, use of CSS2	December 1999
Second generation web authoring environment	February 2000
Final user studies with Amaya, authoring environment	April 2000
Basic constraint solver embedded in SCWM	Completed
Preliminary constraints interface	February 1999
Preliminary user studies on window layout with constraints	July 1999
Embedding of advanced constraint solver in SCWM	October 1999
Revised constraints visualization and manipulation interface	January 2000
Final user studies with SCWM	March 2000
Initial complete draft of dissertation	May 2000
Final dissertation submission and defense	June 2000

References

[BB98] Greg Badros and Alan Borning. The cassowary linear arithmetic constraint solving algorithm: Interface and implementation. Technical Report UW-CSE-98-06-04, University of Washington, Seattle, Washington, June 1998.

[BD86] Alan Borning and Robert Duisberg. Constraint-based tools for building user interfaces. *ACM Transactions on Graphics*, 5(4):345–374, October 1986.

[BFB98] Alan Borning and Bjorn Freeman-Benson. Ultraviolet: A constraint satisfaction algorithm for interactive graphics. *Constraints: An International Jounal*, 3:1–26, 1998.

[BLLJ98] Bert Bos, Håkon Wium Lie, Chris Lilley, and Ian Jacobs. Cascading style sheets, level 2. W3C Working Draft, January 1998. http://www.w3.org/TR/WD-css2/.

[BLM97] Alan Borning, Richard Lin, and Kim Marriott. Constraints for the web. In *Proceedings of 1997 ACM Multimedia Conference*, 1997.

[BMMW89] Alan Borning, Michael Maher, Amy Martindale, and Molly Wilson. Constraint hierarchies and logic programming. In *Proceedings of the Sixth International Conference on Logic Programming*, pages 149–164, Lisbon, June 1989.

[BMSX97] Alan Borning, Kim Marriott, Peter Stuckey, and Yi Xiao. Solving linear arithmetic constraints for user interface applications. In *Proceedings of the 1997 ACM Symposium on User Interface Software and Technology*, October 1997.

[Bor79] Alan Borning. *ThingLab—A Constraint-Oriented Simulation Laboratory*. PhD thesis, Stanford, March 1979. A revised version is published as Xerox Palo Alto Research Center Report SSL-79-3 (July 1979).

[BS] Greg Badros and Maciej Stachowiak. Scwm—the scheme constraints window manager. Web page. http://huis-clos.mit.edu/scwm/.

[Con98] W3 Consortium. Amaya web browser software. Web page, October 1998. http://www.w3.org/Amaya.

[CSI86] Ellis S. Cohen, Edward T. Smith, and Lee A. Iverson. Constraint-based tiled windows. *IEEE Computer Graphics and Applications*, pages 35–45, May 1986.

[FBM89] Bjorn Freeman-Benson and John Maloney. The deltablue algorithm: An incremental constraint hierarchy solver. In *Proceedings of the Eighth Annual IEEE Phoenix Conference on Computers and Communications*, Scottsdale, Arizona, March 1989. IEEE.

[FFK+98] Mary Fernandez, Daniela Florescu, Jaewoo Kang, Alon Levy, and Dan Suciu. Catching the boat with strudel: experience with a web-site management system. In *Proceedings of SIGMOD*, 1998.

[Fru98] Thom Fruhwirth. Theory and practice of constraint handling rules. *Journal of Logic Programming*, 37(1-3):95–138, October 1998. http://www.pst.informatik.uni-muenchen.de/∼fruehwir/chr-intro.html.

[GW94] Michael Gleicher and Andrew Witkin. Drawing with constraints. *Visual Computer*, 11(1):39–51, 1994.

[HM96] Weiqing He and Kim Marriott. Constrainted graph layout. In S. North, editor, *Proceedings of 1996 Graph Drawing Conference*, pages 217–232, Berkeley, CA, September 1996. Springer Verlag.

[HN94] Allan Heydon and Greg Nelson. The Juno-2 constraint-based drawing editor. Technical Report 131a, Digital Systems Research Center, Palo Alto, California, December 1994.

[JL87] Joxan Jaffar and Jean-Louis Lassez. Constraint logic programming. In *Proceedings of the Fourteenth ACM Principles of Programming Languages Conference*, Munich, January 1987.

[KS96] Eser Kandogan and Ben Shneiderman. Elastic windows: Improved spatial layout and rapid multiple window operations. Web page, May 1996. http://www.cs.umd.edu/users/kandogan/papers/avi96/paper4.html.

[LB96] Håkon Wium Lie and Bert Bos. Cascading style sheets, level 1. W3C Recommendation, December 1996. http://www.w3.org/pub/WWW/TR/PR-CSS1/.

[MGD+90] Brad A. Myers, Dario Giuse, Roger B. Dannenberg, Brad Vander Zanden, David Kosbie, Philippe Marchal, Ed Pervin, Andrew Mickish, and John A. Kolojejchick. The Garnet toolkit reference manuals: Support for highly-interactive graphical user interfaces in Lisp. Technical Report CMU-CS-90-117, Computer Science Dept, Carnegie Mellon University, March 1990.

[Mic98] Brian Michalowski. A constraint-based specification for box layout in css2. Technical Report UW-CSE-98-06-03, University of Washington, June 1998.

[MMM+97] Brad A. Myers, Richard G. McDaniel, Robert C. Miller, Alan S. Ferrency, Andrew Faulring, Bruce D. Kyle, Andrew Mickish, Alex Klimovitski, and Patrick Doane. The Amulet environment: New models for effective user interface software development. *IEEE Transactions on Software Engineering*, 23(6):347–365, June 1997.

[RMS97] Kathy Ryall, Joe Marks, and Stuart Shieber. An interactive constraint-based system for drawing graphs. In *Proceedings of 1997 UIST Conference*, Banff, Alberta Canada, October 1997.

[Sut63] Ivan Sutherland. *Sketchpad: A Man-Machine Graphical Communication System.* PhD thesis, Department of Electrical Engineering, MIT, January 1963.

[TMM+98] Shin Takahashi, Satoshi Matsuoka, Ken Miyashita, Hiroshi Hosobe, and Tomihisa Kamada. A constraint based approach for visualization and animation. *Constraints: An International Jounal*, 3:61–86, 1998.

[VHM95] Pascal Van Hentenryck and Laurent Michel. Newton: Constraint programming over nonlinear real constraints. Technical Report CS-95-25, Brown University, Providence, Rhode Island, August 1995.