LAUNCHPADS: A System for Processing Ad Hoc Data

Mark Daly
Princeton University
mdaly@Princeton.EDU

Mary Fernández
Kathleen Fisher
AT&T Labs Research
mff,kfisher@research.att.com

Yitzhak Mandelbaum
David Walker
Princeton University
yitzhakm,dpw@CS.Princeton.EDU

An Introduction to PADS. Ideally, any data we ever encounter will be presented to us in standardized formats, such as XML. Why? Because for formats like XML, there are a whole host of software libraries, query engines, visualization tools and even programming languages specially designed to help users process their data. However, we do not live in an ideal world, and in reality, vast amounts of data is produced and communicated in *ad hoc formats*, those formats for which no data processing tools are readily available. Figure 1 presents a small selection of ad hoc data sources. As one can see, ad hoc data exists in a very wide variety of fields and the users range from network administrators to computational biologists and genomics researchers to physicists, financial analysts and everyday programmers.

Programmers often deal with this data by whipping up one-time Perl scripts or C programs to parse and analyze their data. Unfortunately, this strategy is slow and tedious, and often produces code that is difficult to understand, lacks adequate error checking, and is brittle to format change over time. To expedite and improve this process, we developed the PADS data description language and system [2, 3]. Using the PADS language, one may write a declarative description of the structure of almost any ad hoc data source. The descriptions take the form of types, drawn from a dependent type theory. For instance, PADS base types describe simple objects including strings, integers, floating-point numbers, dates, times, and ip addresses. Records and arrays specify sequences of elements in a data source, and unions, switched unions and enums specify alternatives. Any of these structured types may be parameterized and users may write arbitrary semantic constraints over their data as well.

Once a programmer has written a description in the PADS language, the PADS compiler can generate a collection of format-specific libraries in C, including a parser, printer, and verifier. In addition, the compiler can compose these libraries with generic templates to create value-added tools such as an ad hoc-to-XML format conversion tool, a histogram generator, and a statistical analysis and error summary tool. Finally, PADS has been composed with the GALAX query engine [6, 4, 5] for XQuery to create PADX [1], a new system that allows users to query and transform any ad hoc data source as if it was XML, without incurring the performance penalty that usually results when one converts ad hoc data into a much more verbose XML representation.

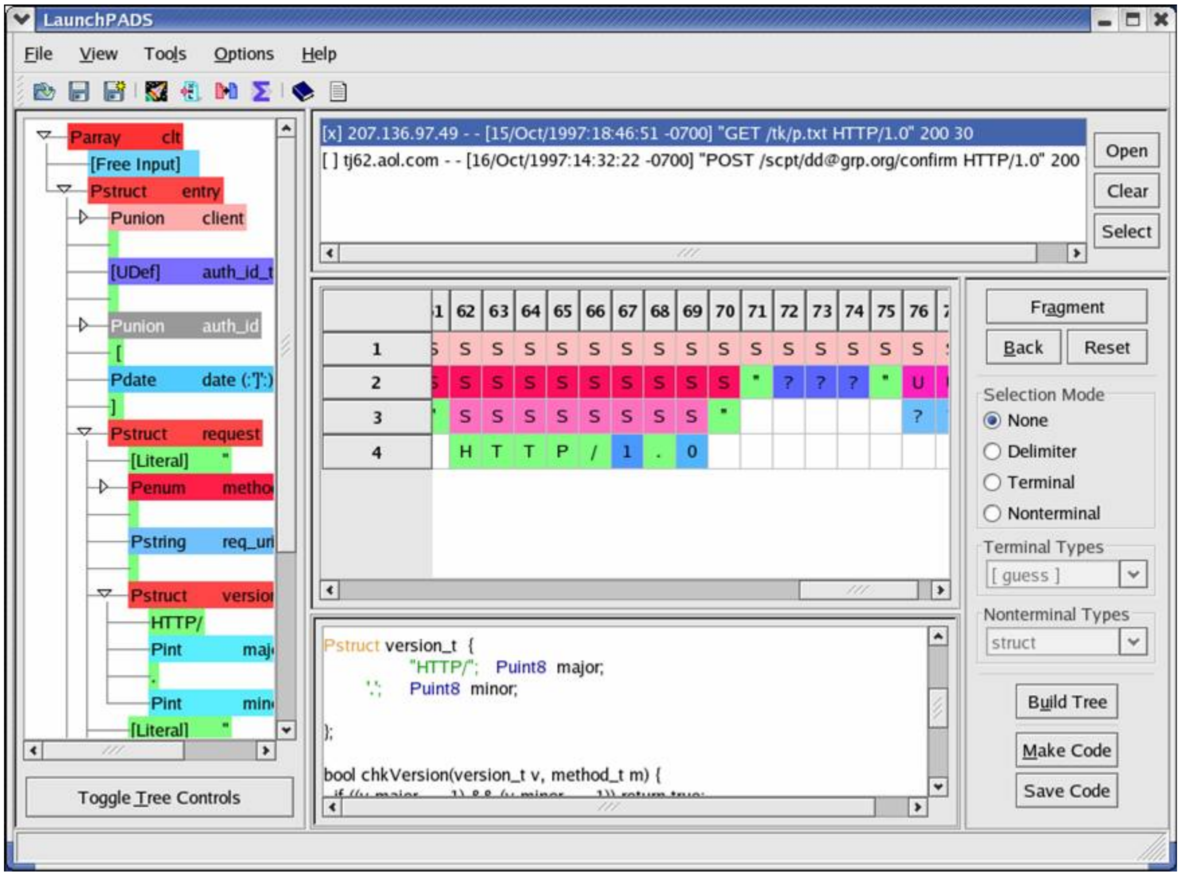
While the PADS language provides an extremely versatile means of creating tools for processing ad hoc data, it is nevertheless a *new* language and learning a new language is time-consuming for anyone, especially for computational biologists or other scientists for whom programming is not their primary area of expertise. To ease the way for novice PADS users, we developed LAUNCHPADS, a new tool that provides access to the PADS system without requiring foreknowledge of the PADS language itself. Hence, LAUNCHPADS graphic interface will also help more experienced PADS users to shorten their development cycle and provides a convenient way for experts to quickly create any of the data processing tools they need.

Name : Use	Representation
Web server logs (CLF): Measure web workloads	Fixed-column ASCII records
CoMon data: Monitor PlanetLab Machines	ASCII records
Call detail: Fraud detection	Fixed-width binary records
AT&T billing data: Monitor billing process	Various Cobol data formats
Netflow: Monitor network performance	Data-dependent number of fixed-width binary records
Newick: Immune system response simulation	Fixed-width ASCII records in tree-shaped hierarchy
Gene Ontology: Gene-gene correlations	Variable-width ASCII records in DAG-shaped hierarchy
CPT codes: Medical diagnoses	Floating point numbers

Figure 1. Selected ad hoc data sources.

LaunchPads. LAUNCHPADS combines mechanisms for graphically defining structure and semantic properties of ad hoc data, for translation of this definition into PADS code, and for compilation/execution of the generic tools that operate over ad hoc data. More specifically, LAUNCHPADS breaks definition of an ad hoc data format and generation of data processing tools into the following steps. Figure 2 presents a screenshot of LAUNCHPADS being used to construct a data description for a web-server log format.

1. **Selection of sample data.** Creation of a definition within LAUNCHPADS begins when a user loads sample data into the graphical interface. In Figure 2, web log data (beginning with the IP address `207.136.97.49 ...`) appears in the top right hand corner of the picture. A user then selects a row of data to work on in the LAUNCHPADS *gridview*.
2. **Iterative refinement in the gridview.** Once in the gridview, which appears just below the data selection window in Figure 2, users may specify descriptions for regions of text using a highlighting scheme. The color assigned to a region represents the description class (base or composite) and region boundaries. Structure within a definition is represented through a series of refinement steps: composite regions are broken down level after level, thereby allowing for nested elements (Figure 2 shows four nesting levels). The refinement process bottoms out when one reaches an atomic description such as a character string, IP address or date. Once all regions have been given a base type in the gridview, LAUNCHPADS will generate a *treeview* of the definition for further processing.



```
[x] 207.136.97.49 - - [15/Oct/1997:18:46:51 -0700] "GET /tk/p.txt HTTP/1.0" 200 30
[ ] tj62.aol.com - - [16/Oct/1997:14:32:22 -0700] "POST /scpt/dd@grp.org/confirm HTTP/1.0" 200
```

	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76
1	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
2	S	S	S	S	S	S	S	S	S	"	?	?	?	"	U
3	S	S	S	S	S	S	S	S	"						?
4	H	T	T	P	/	1	.	0							

Selection Mode: [✓ None] Delimiter Terminal Nonterminal

Terminal Types: [guess]

Nonterminal Types: struct

```
Pstruct version_t {
            "HTTP/";   Puint8  major;
      '.';     Puint8  minor;

};

bool chkVersion(version_t v, method_t m) {
```

Figure 2. LaunchPads Interface.

3. **Customization in the treeview.** The treeview, which appears on the left-hand side of Figure 2, is a graphical representation of the abstract syntax of a PADS description. In this view, programmers can manipulate definitions with a high degree of precision: definition elements may be created, destroyed, and renamed; type associations for existing elements may be changed (within limitations); element ordering may be altered; user defined types may be added to the definition and applied to elements; content-aware error constraints may be imposed. Indeed, from within the tree view it is possible to access the "expert" functions of PADS directly if one so chooses, or to completely avoid them in lieu of a simpler definition and/or faster development time.
4. **PADS code generation, tool compilation and use.** When the user is satisfied with their PADS definition in the treeview, they may generate PADS code. Any such generated code is guaranteed to be syntactically correct so the user need not worry about fussing with concrete PADS syntax if they do not want to. Figure 2 shows the generated code in the window at the bottom of the interface. By using the pulldown menus at the top and a set of "wizards" (not shown), the user may now issue commands to compile the generated code and create data processing tools including the XML converter and statistical analyzer. As development of LAUNCHPADS continues, we will add further tools and corresponding wizards to the interface.

Conclusions In summary, in this demonstration, we will explain the many challenges that ad hoc data pose and how the PADS language is structured to meet these challenges. In addition, we will explain how LAUNCHPADS provides further support for processing ad hoc data by demonstrating both features for helping users construct data descriptions and features for creating and invoking tools that operate over data. We believe that both expert programmers and novices alike can benefit from this simple system for manipulating ad hoc data.

References

[1] M. Fernández, K. Fisher, and Y. Mandelbaum. PADX: Querying large-scale ad hoc data with XQuery. Submitted to PLAN-X 2006.

[2] K. Fisher and R. Gruber. PADS: A domain-specific language for processing ad hoc data. In *Proceedings of the ACM SIGPLAN 2005 conference on Programming language design and implementation*, June 2005.

[3] K. Fisher, Y. Mandelbaum, and D. Walker. The next 700 data description languages. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, Jan. 2006. To appear.

[4] Galax user manual. `http://www.galaxquery.org/doc.html#manual`.

[5] C. Ré, J. Siméon, and M. Fernández. A complete and efficient algebraic compiler for XQuery. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)*, April 2006.

[6] J. Siméon and M. F. Fernández. Build your own XQuery processor. EDBT Summer School, Tutorial on Galax architecture, Sept 2004.

.