PADS: A Domain-Specific Language for Processing Ad Hoc Data

talk based on paper by

Kathleen Fisher[1] Robert Gruber[2]

[1] AT&T Labs Research [2] Google

November 10, 2005

Outline

1. Why worry about ad hoc data?
2. Current options for processing ad hoc data
3. What PADS does for you

Outline

Data and formatting

Massive amounts of data is generated and collected every day.

Formatted data is easier to understand and process.

Formats can either be standardized, or not standardized.

Standard vs nonstandard formats

Boundary not so clear, but...

there is certainly a difference in the size of the expected producer/user-base. Standard formats tend to have many applications that generate and process them. Standardized formats should be designed to expect a growing user-base.

Examples of data in standard formats:

- webpages in HTML
- pictures in JPEG
- XML
- data in databases
- The billions of lines of code you write every day?

Ad hoc data (data in nonstandard formats)

Claim/(fact?): Most data is actually stored in nonstandard formats.
Here are some examples:

- AT&T accumulates 250-300GB/day of billing data
- Netflow data arrives at Cisco routers at over a GB/sec.

Common Log Format (CLF) for web servers

One record per line, with 7 fields:

```
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" 200 941
```

The '-' character indicates missing data for that field

Investment data

Date: 3/21/2005 1:00PM PACIFIC
Investor's Business Daily ®
Stock List Name: DAVE

Stock Symbol	Company Name	Price	Price Change	Price % Change	Volume % Change	EPS Rating	RS Rating
AET	Aetna Inc	73.68	-0.22	0%	31%	64	93
GE	General Electric Co	36.01	0.13	0%	-8%	59	56
HD	Home Depot Inc	37.99	-0.89	-2%	63%	84	38
IBM	Intl Business Machines	89.51	0.23	0%	-13%	66	35
INTC	Intel Corp	23.50	0.09	0%	-47%	39	33

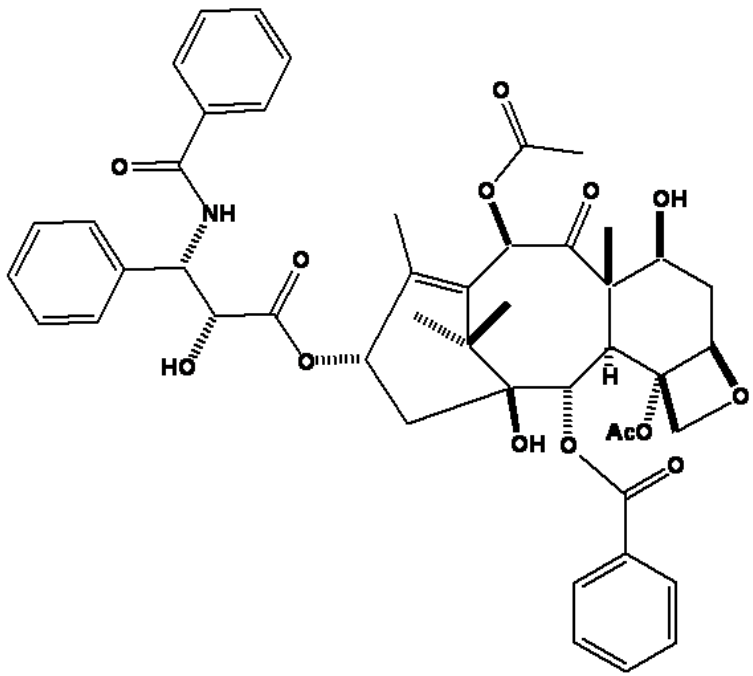
All prices are delayed at least 20 minutes.

Records within another kind of record

Ad hoc data in chemistry

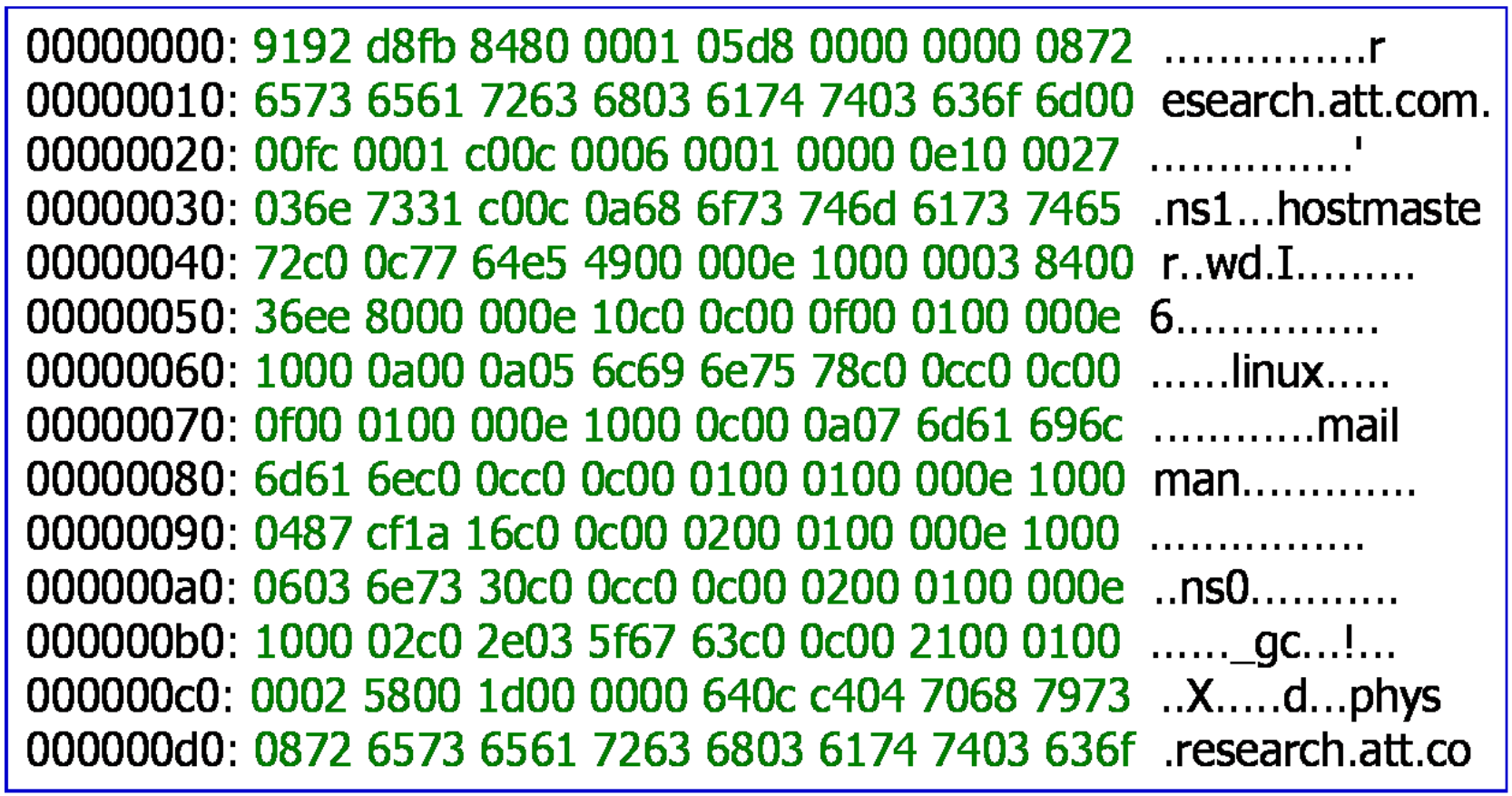
```
O=C([C@@H]2OC(C)=O)[C@@]3(C)[C@]([C@](CO4)
(OC(C)=O)[C@H]4C[C@@H]3O)([H])[C@H]
(OC(C7=CC=CC=C7)=O)[C@@]1(O)[C@@](C)(C)C2=C(C)
[C@@H](OC([C@H](O)[C@@H](NC(C6=CC=CC=C6)=O)
C5=CC=CC=C5)=O)C1
```

Context-free?



DNS data

Binary data



Small audience, big expectations

Groups that work with ad hoc data, no matter how small, still need tools that

- parse the data to load into applications
- visualize the data
- allow queries over the data
- convert the data (maybe to load into a database)
- detect errors in the data
- correct errors
- filter data
- combine data from multiple sources

Outline

Perl/AWK/Shell scripts/C

Pros

- flexible

Cons

- time investment
  - hand-code parser
  - hand-code error handling (if you even bother)
  - hand-code other tools (e.g., converters, viewers)
- error-prone
- difficult to maintain in the face of format changes

Lex + Yacc

Pros

- ?

Cons

- specify lexer and parser separately
- error handling inflexible
- still need to hand-code other tools (e.g., converters, viewers)

People don't use Lex + Yacc for ad hoc data.

Outline

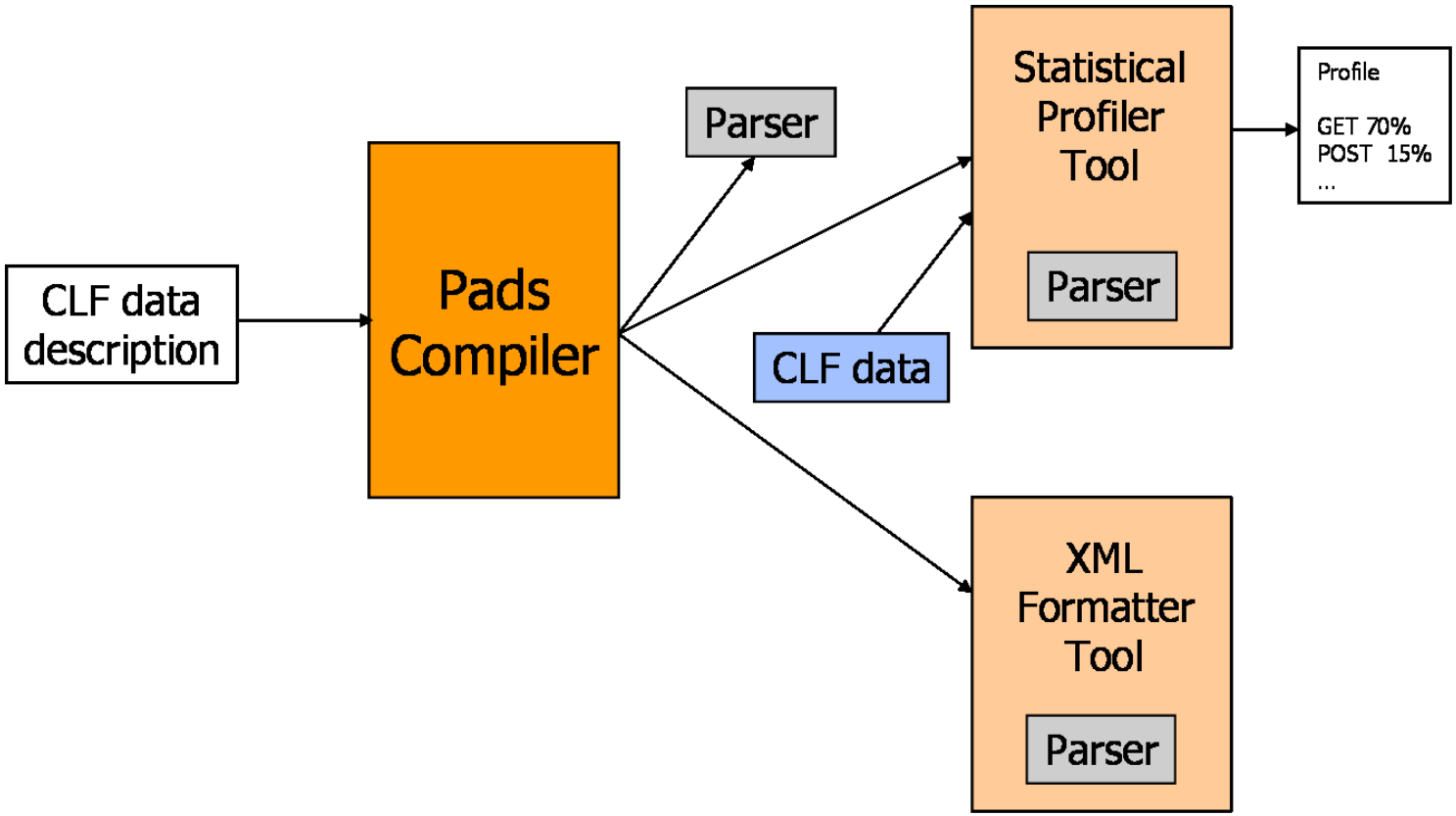
What we want

- Simple data description language
- Parser
  - should handle various encodings (ASCII, binary, etc.)
  - should not halt on errors, and
  - should track errors (syntactic + semantic)
  - generate useful in-memory representation
- Other tools
  - converter to other formats (e.g., XML, CSV)
  - statistical profiler
  - query interface
  - more?
- High performance

What we get from PADS

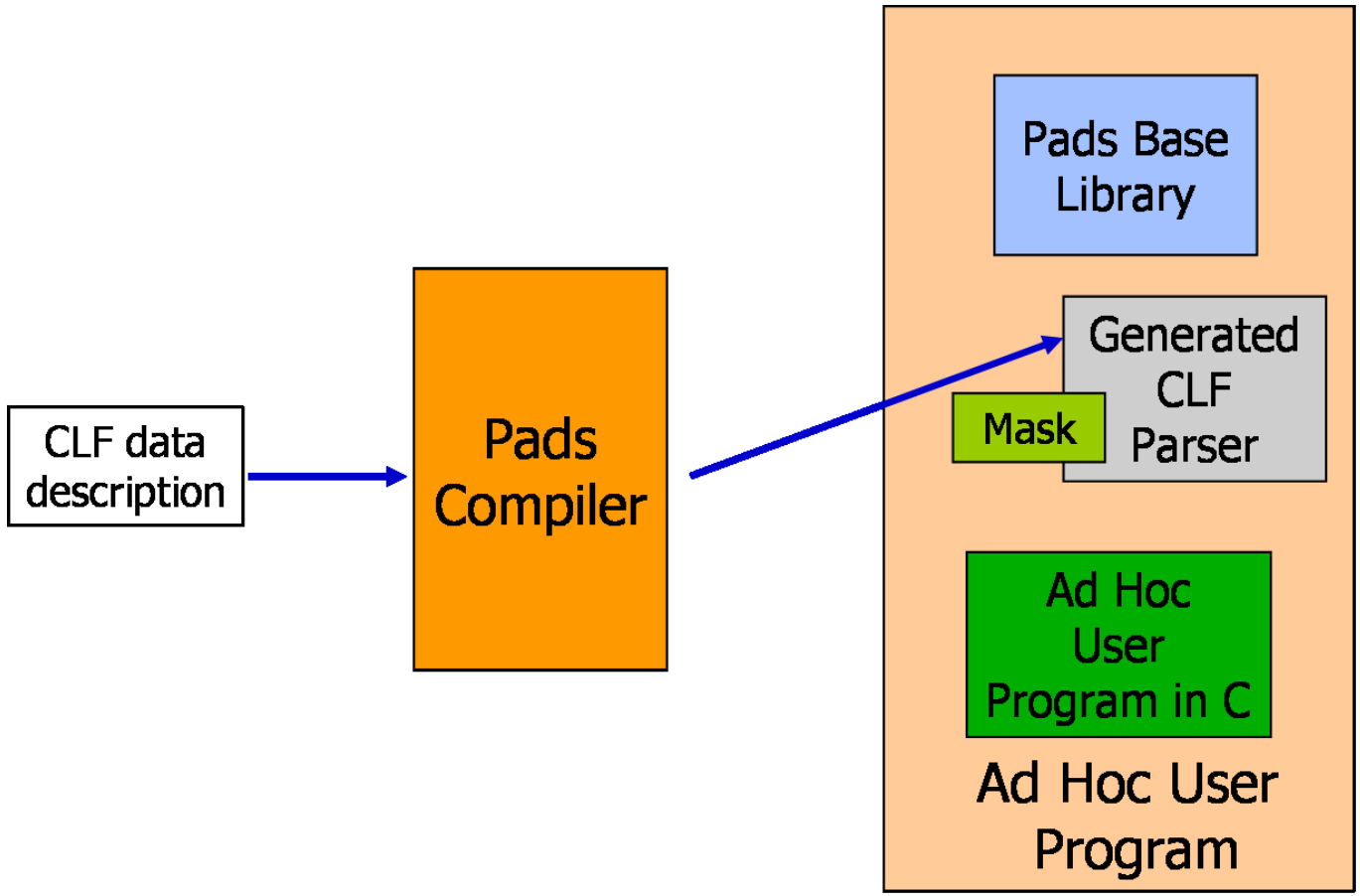
- Simple data description language
- Parser
  - handles ASCII, EBCDIC, and binary
  - does not halt on errors, and
  - tracks errors (syntactic + semantic) in a parse-descriptor struct
  - generates C structs, unions, etc., for in-memory rep.
- Other tools
  - convert/print to other formats (XML + XML Schema, CSV)
  - statistical profilers via accumulator programs
  - query interface
  - more in the future
- Faster than Perl programs
- Can stream data, rather than read it all into memory at once

PADS architecture



(Not to scale)

PADS architecture: a closer look



Note: Use masks to select only the features that are relevant to the application $\Rightarrow$ increased performance

Data Description Language

Primitives (`Pint32, Pdate, Pstring, Pip`).

Complex data (`Pstruct, Punion, Parray, Popt, Ptypedefs, Penum`).

Parameterized types (e.g., `Puint16_FW(:3:)`).

Can attach predicates for error checking (see following example).

CLF: Checking HTTP version conformance

```
Penum method_t {
    GET,    PUT,  POST,  HEAD,
    DELETE, LINK, UNLINK
};

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

int chkVersion(version_t v,
method_t m) {
  if ((v.major == 1) && (v.minor ==
1)) return 1;
  if ((m == LINK) || (m == UNLINK))
return 0;
  return 1;
};
```

```
Pstruct request_t {
  '\"';   method_t        meth;
  ' ';    Pstring(:' ':) req_uri;
  ' ';    version_t      version :

chkVersion(version, meth);
  '\"';
};
```

Siruis Data: Check that timestamps increase monotonically

```
Pstruct event_t {
    Pstring(:'|':) state;
    '|';
    Puint32        tstamp;
};

Parray eventSeq {
    event_t[] : Psep('|') &&
Pterm(Peor);
} Pwhere {
   Pforall(i Pin [0..length-2] :
                  (elts[i].tstamp <=
elts[i+1].tstamp));
};
```

Conclusion

- They evaluate the work by showing how to construct parsers for a few formats and comparing the performance of generated tools to hand-coded perl programs. PADS generated tools are no slower (actually faster in all 6 runs and 2x faster in 3 of the runs).
- Rather than standardize a format (e.g., HTML), then create tools, PADS can automatically generate tools as the format evolves. The only cost is the time to update the PADS description.

If you want to try it, go to **http://www.padsproj.org**

Discussion

- What other methods of evaluation would you like to see for work of this kind?
- The PADS group plans to add more auxiliary tool generators. Is it really useful, or do you imagine people building custom tools using only the generated parser?
- If people build custom tools, does that also build up data-format inertia?
- Do you feel that the PADS compiler has *more* opportunities to optimize the generated parser and tools than compiled hand-coded tools?