Programming Language Ideas Escape the Lab:

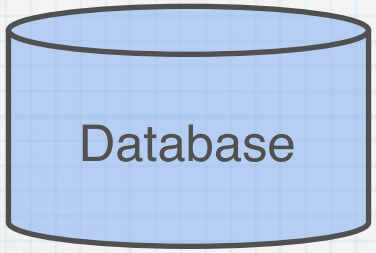
A Declarative Data Description Language

Kathleen Fisher
AT&T Labs Research
www.padsproj.org

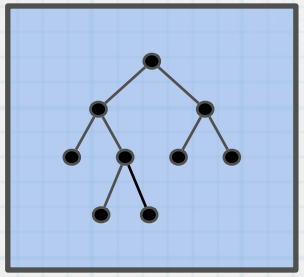
Data, Data, Everywhere!

Incredible amounts of data stored in well-behaved formats:

Databases:



XML:

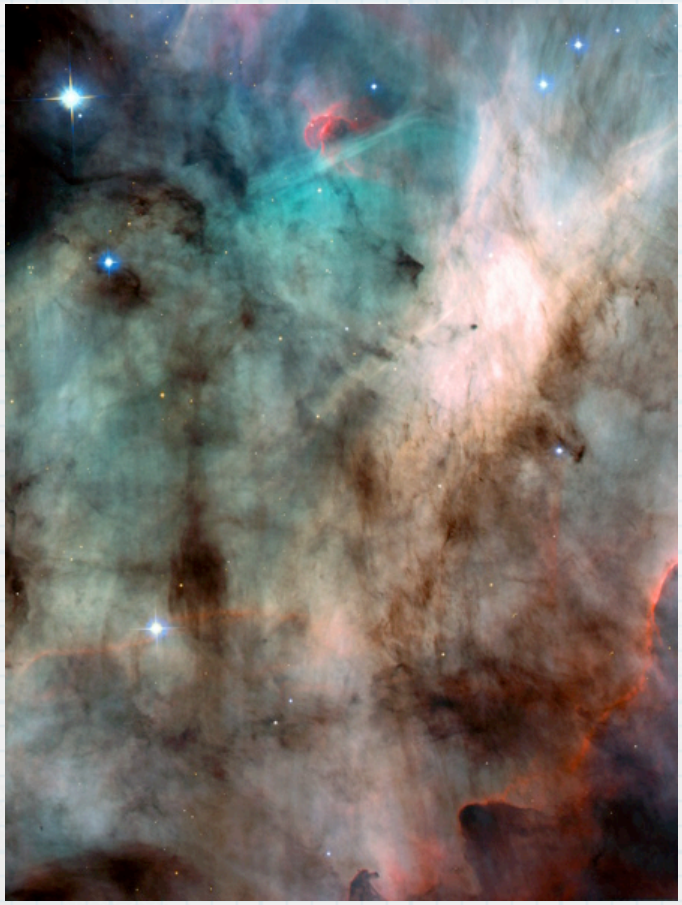


Tools

- Schema
- Browsers
- Query Languages
- Standards
- Libraries
- Books, documentation
- Training courses
- Conversion tools
- Vendor support
- Consultants...

We're not always so lucky!

Vast amounts of chaotic ad hoc data:



Tools

- Perl
- Awk
- C
- ...

Government Statistics

```
"MSN","YYYYMM","Publication Value","Publication Unit","Column Order"
"TEAJBUS",197313,-0.456483,Quadrillion Btu,4
"TEAJBUS",197413,-0.482265,Quadrillion Btu,4
"TEAJBUS",197513,-1.066511,Quadrillion Btu,4
"TEAJBUS",197613,-0.177807,Quadrillion Btu,4
"TEAJBUS",197713,-1.948233,Quadrillion Btu,4
"TEAJBUS",197813,-0.336538,Quadrillion Btu,4
"TEAJBUS",197913,-1.649302,Quadrillion Btu,4
"TEAJBUS",198013,-1.0537,Quadrillion Btu,4
```

Train Stations

```
Southern California Regional Railroad Authority,"Los Angeles, CA",
U,45,46,46,47,49,51,U,45,46,46,47,49,51
Connecticut Department of Transportation ,"New Haven, CT",
U,U,U,U,U,U,8,U,U,U,U,U,U,8
Tri-County Commuter Rail Authority ,"Miami, FL",
U,U,U,U,U,U,18,U,U,U,U,U,U,18
Northeast Illinois Regional Commuter Railroad Corporation,"Chicago, IL",
226,226,226,227,227,227,227,91,104,104,111,115,125,131
Northern Indiana Commuter Transportation District,"Chicago, IL",
18,18,18,18,18,18,20,7,7,7,7,7,7,11
Massachusetts Bay Transportation Authority,"Boston, MA",
U,U,117,119,120,121,124,U,U,67,69,74,75,78
Mass Transit Administration – Maryland DOT ,"Baltimore, MD",
U,U,U,U,U,U,42,U,U,U,U,U,U,22
New Jersey Transit Corporation ,"New York, NY",
158,158,158,162,162,162,167,22,22,41,46,46,46,51
```

Web Server Logs

```
207.136.97.49 – – [15/Oct/2006:18:46:51 -0700] "GET /turkey/amnty1.gif HTTP/1.0" 200 3013
207.136.97.49 – – [15/Oct/2006:18:46:51 -0700] "GET /turkey/clear.gif HTTP/1.0" 200 76
207.136.97.49 – – [15/Oct/2006:18:46:52 -0700] "GET /turkey/back.gif HTTP/1.0" 200 224
207.136.97.49 – – [15/Oct/2006:18:46:52 -0700] "GET /turkey/women.html HTTP/1.0" 200 17534
208.196.124.26 – Dbuser [15/Oct/2006:18:46:55 -0700] "GET /candatop.html HTTP/1.0" 200 -
208.196.124.26 – – [15/Oct/2006:18:46:57 -0700] "GET /images/done.gif HTTP/1.0" 200 4785
www.att.com – – [15/Oct/2006:18:47:01 -0700] "GET /images/reddash2.gif HTTP/1.0" 200 237
208.196.124.26 – – [15/Oct/2006:18:47:02 -0700] "POST /images/refrun1.gif HTTP/1.0" 200 836
208.196.124.26 – – [15/Oct/2006:18:47:05 -0700] "GET /images/hasene2.gif HTTP/1.0" 200 8833
www.cnn.com – – [15/Oct/2006:18:47:08 -0700] "GET /images/candalog.gif HTTP/1.0" 200 -
208.196.124.26 – – [15/Oct/2006:18:47:09 -0700] "GET /images/nigpost1.gif HTTP/1.0" 200 4429
208.196.124.26 – – [15/Oct/2006:18:47:09 -0700] "GET /images/rally4.jpg HTTP/1.0" 200 7352
128.200.68.71 – – [15/Oct/2006:18:47:11 -0700] "GET /amnesty/usalinks.html HTTP/1.0" 143 10329
208.196.124.26 – – [15/Oct/2006:18:47:11 -0700] "GET /images/reyes.gif HTTP/1.0" 200 10859
```

Genetic Data

```
((raccoon:19.19959,bear:6.80041):0.84600,((sea_lion:
11.99700,seal:12.00300):7.52973,((monkey:100.85930,cat:
47.14069):20.59201,weasel:18.87953):2.09460):3.87382,dog:
25.46154); (Bovine:0.69395,(Gibbon:0.36079,(Orang:0.33636,
(Gorilla:0.17147,(Chimp:0.19268,Human:0.11927):0.08386):
0.06124):0.15057):0.54939,Mouse:1.21460):0.10; (Bovine:
0.69395,(Hylobates:0.36079,(Pongo:0.33636,(G._Gorilla:
0.17147,(P._paniscus:0.19268,H._sapiens:0.11927):0.08386):
0.06124):0.15057):0.54939,Rodent:1.21460);
```

Haskell HI files

```
00000000: 0001 face 0000 0073 0400 0000 3600 0000  .......s....6...
00000010: 3000 0000 3500 0000 3000 0000 0000 0000  0...5...0.......
00000020: 0001 0000 0000 0100 0000 0043 0001 0000  ............C....
00000030: 0002 0200 0000 0200 0000 0300 0000 0200  ................
00000040: 0000 0400 0000 4800 0100 0000 0200 0000  ......H.........
00000050: 0502 0000 0000 0006 0000 0000 0007 0000  ................
00000060: 0001 0000 0000 6800 0000 0000 006f 0000  ......h......o..
00000070: 0000 0100 0000 0800 0000 0968 6173 6b65  ............haske
00000080: 6c6c 3938 0000 0007 4350 5554 696d 6500  ll98....CPUTime.
00000090: 0000 0462 6173 6500 0000 0847 4843 2e42  ...base....GHC.B
000000a0: 6173 6500 0000 0e47 4843 2e46 6f72 6569  ase....GHC.Forei
000000b0: 676e 5074 7200 0000 0e53 7973 7465 6d2e  gnPtr....System.
000000c0: 4350 5554 696d 6500 0000 0a67 6574 4350  CPUTime....getCP
000000d0: 5554 696d 6500 0000 1063 7075 5469 6d65  UTime....cpuTime
000000e0: 5072 6563 6973 696f 6e                   Precision
```

Ad hoc data from AT&T

Name & Use	Representation	Size
Web server logs (CLF): Measure web workloads	Fixed-column ASCII records	≤ 12 GB/week
Sirius data: Monitor service activation	Variable-width ASCII records	2.2 GB/week
Call detail: Detect fraud	Fixed-width binary records	~7GB/day
Altair data: Track billing process	Various Cobol data formats	~4000 files/day
Regulus data: Monitor IP network	ASCII	≥ 15 sources, ~15 GB/day
Netflow: Monitor IP network	Data-dependent number of fixed-width binary records	>1Gigabit/second

And many others...

* Gene ontology data
* Cosmology data
* Financial trading data
* Telecom billing data
* Router config files
* System logs
* Call detail data
* Netflow packets
* DNS packets
* Java JAR files
* Jazz recording info
* ...

Technical Challenges

Technical Challenges

* Data arrives "as is" in many encodings and formats.

Technical Challenges

* Data arrives "as is" in many encodings and formats.
* Documentation is often out-of-date or nonexistent.
  * Hijacked fields.
  * Undocumented "missing value" representations.

Technical Challenges

* Data arrives "**as is**" in many encodings and formats.
* Documentation is often **out-of-date** or **nonexistent**.
  * Hijacked fields.
  * Undocumented "missing value" representations.
* Data is **buggy**.
  * Missing data, human error, malfunctioning machines, race conditions on log entries, "extra" data, ...
  * Processing must detect **relevant errors** and respond in **application-specific** ways.
  * Errors are sometimes the **most** interesting portion of the data.

Technical Challenges

* Data arrives "**as is**" in many encodings and formats.
* Documentation is often **out-of-date** or **nonexistent**.
  * Hijacked fields.
  * Undocumented "missing value" representations.
* Data is **buggy**.
  * Missing data, human error, malfunctioning machines, race conditions on log entries, "extra" data, ...
  * Processing must detect **relevant errors** and respond in **application-specific** ways.
  * Errors are sometimes the **most** interesting portion of the data.
* Data sources often have **high volume**.

Conventional Approaches

* Lex/Yacc
  * Target PL syntax, not data description.
  * Overkill & Underkill for data descriptions.
* Perl/C
  * Code brittle with respect to changes in format.
  * Analysis ends up interwoven with parsing, precluding reuse.
  * Error code, if written, swamps main-line computation. If not written, errors can corrupt "good" data.
  * Everything has to be coded by hand.

Types to the Rescue!

Relational and XML data are easier to manage (partly) because schema exist to describe the data.

Relational Data	Relational Schema
XML	XML Schema
Ad Hoc Data	???

Types to the Rescue!

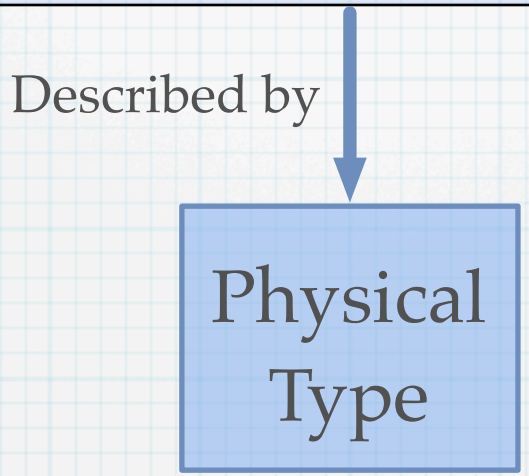
Relational and XML data are easier to manage (partly) because schema exist to describe the data.

Relational Data	Relational Schema
XML	XML Schema
Ad Hoc Data	Physical Types

Thesis: Types can facilitate ad hoc data management. Familiar types from programming languages are suited to the task.

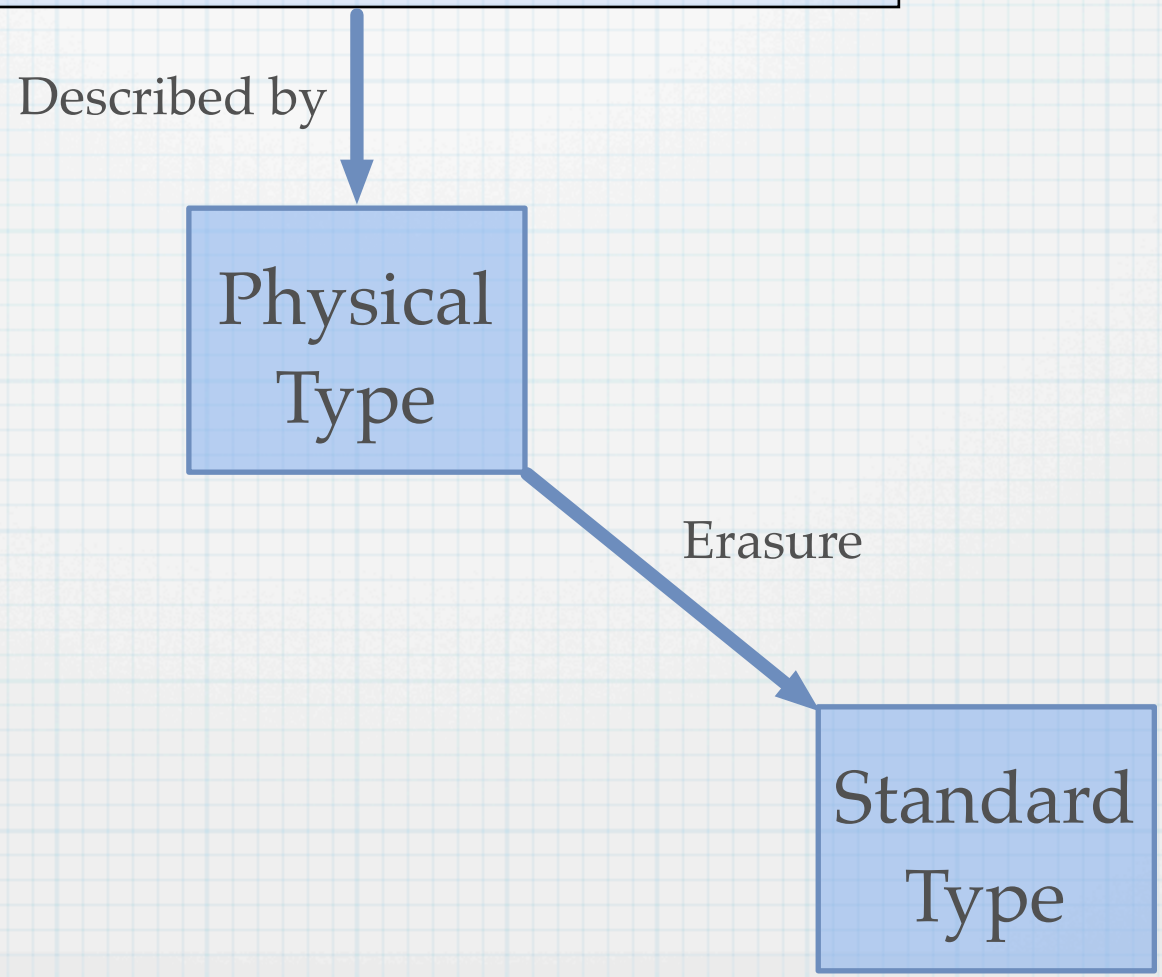
Typing Ad hoc Data

```
"TEAJBUS",197713,-1.948233,Quadrillion Btu,4
"TEAJBUS",197813,-0.336538,Quadrillion Btu,4
"TEAJBUS",197913,-1.649302,Quadrillion Btu,4
"TEAJBUS",198013,-1.0537,Quadrillion Btu,4
```



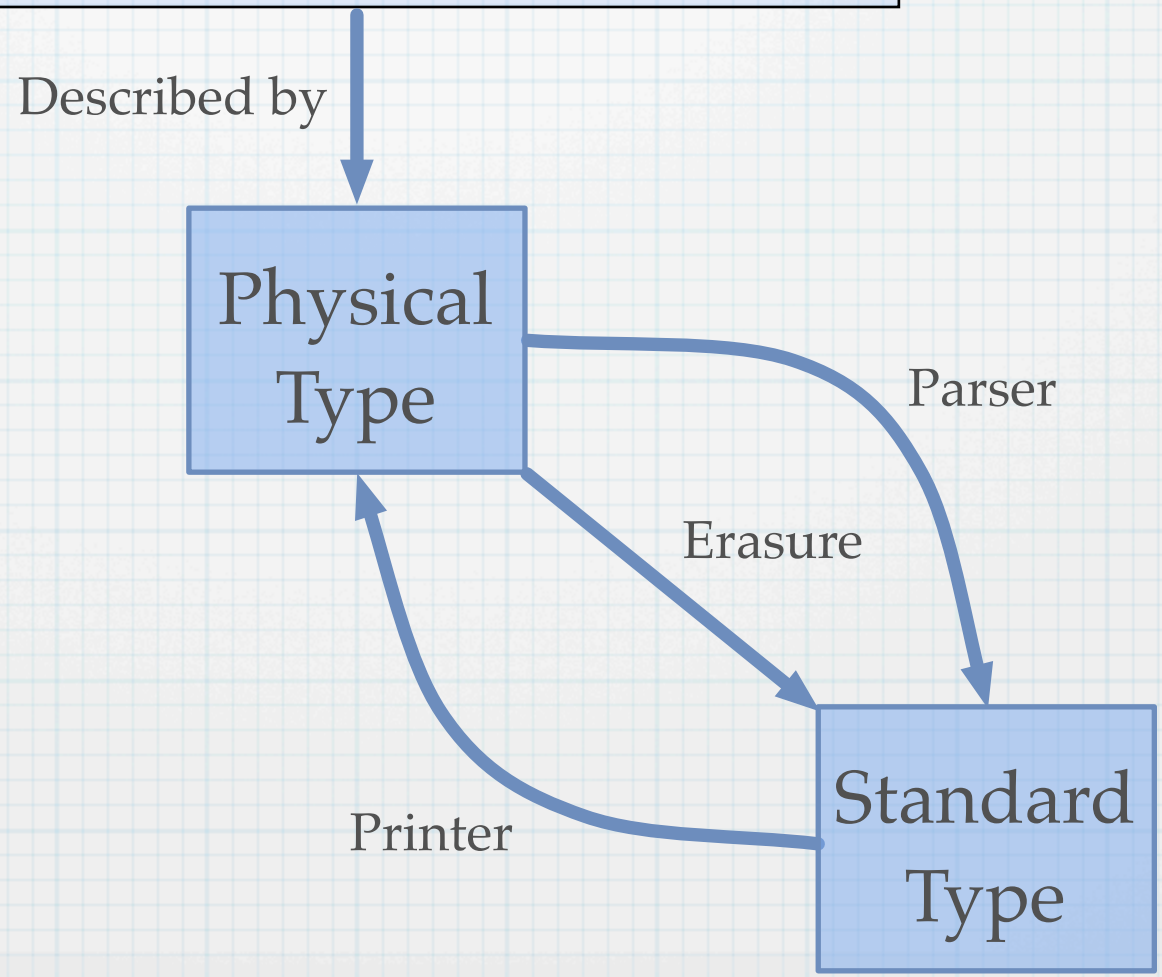
Typing Ad hoc Data

```
"TEAJBUS",197713,-1.948233,Quadrillion Btu,4
"TEAJBUS",197813,-0.336538,Quadrillion Btu,4
"TEAJBUS",197913,-1.649302,Quadrillion Btu,4
"TEAJBUS",198013,-1.0537,Quadrillion Btu,4
```



Typing Ad hoc Data

```
"TEAJBUS",197713,-1.948233,Quadrillion Btu,4
"TEAJBUS",197813,-0.336538,Quadrillion Btu,4
"TEAJBUS",197913,-1.649302,Quadrillion Btu,4
"TEAJBUS",198013,-1.0537,Quadrillion Btu,4
```



Roadmap

* Introduction
* Exploring how types describe physical data
* Differences
* Further connections with PL ideas
* Physical type inference
* Conclusion

Base Types

```
"TEAJBUS",197313,-0.456483,Quadrillion Btu,4
"TEAJBUS",197413,-0.482265,Quadrillion Btu,4
```

Base Types

```
"TEAJBUS",197313,-0.456483,Quadrillion Btu,4
"TEAJBUS",197413,-0.482265,Quadrillion Btu,4
```

String, Int, Float

Tuple Types

```
"TEAJBUS",197313,-0.456483,Quadrillion Btu,4
"TEAJBUS",197413,-0.482265,Quadrillion Btu,4
```

```
String * Int * Float * String * Int
```

Singleton Types

```
"TEAJBUS",197313,-0.456483,Quadrillion Btu,4
"TEAJBUS",197413,-0.482265,Quadrillion Btu,4
```

```
'\"' * String * '\"' * ','
     * Int            * ','
     * Float          * ','
     * String         * ','
     * Int
```

Singleton Types

```
"TEAJBUS",197313,-0.456483,Quadrillion Btu,4
"TEAJBUS",197413,-0.482265,Quadrillion Btu,4
```

```
'\"' * String * '\"' * ','
     * Int            * ','
     * Float          * ','
     * String         * ','
     * Int
```

We write ' , ' for the **singleton** type containing only the value ' , '.

Simple Dependent Types

```
"TEAJBUS",197313,-0.456483,Quadrillion Btu,4
"TEAJBUS",197413,-0.482265,Quadrillion Btu,4
```

```
'\"' * String('\"') * '\"' * ','
     * Int                  * ','
     * Float                * ','
     * String(',')          * ','
     * Int
```

Records

```
"TEAJBUS",197313,-0.456483,Quadrillion Btu,4
"TEAJBUS",197413,-0.482265,Quadrillion Btu,4
```

```
{                                   '\"'
  source: String('\"'),             "\","
  date:   Int,                      ','
  measurement: Float,               ','
  units:  String(',')               ','
  order:  Int
}
```

Unions

```
Southern California Regional Railroad Authority,"Los Angeles, CA",
U,45,46,46,47,49,51,U,45,46,46,47,49,51
Connecticut Department of Transportation ,"New Haven, CT",
U,U,U,U,U,U,8,U,U,U,U,U,U,8
Tri-County Commuter Rail Authority ,"Miami, FL",
U,U,U,U,U,U,18,U,U,U,U,U,U,18
```

Anonymous:

‘U’ + Int

Unions

```
Southern California Regional Railroad Authority,"Los Angeles, CA",
U,45,46,46,47,49,51,U,45,46,46,47,49,51
Connecticut Department of Transportation ,"New Haven, CT",
U,U,U,U,U,U,8,U,U,U,U,U,U,8
Tri-County Commuter Rail Authority ,"Miami, FL",
U,U,U,U,U,U,18,U,U,U,U,U,U,18
```

Anonymous:

```
'U' + Int
```

Named:

```
type OptInt = unavailable of 'U'
            |   available of Int
```

Arrays/Lists

```
Southern California Regional Railroad Authority,"Los Angeles, CA",
U,45,46,46,47,49,51,U,45,46,46,47,49,51
Connecticut Department of Transportation ,"New Haven, CT",
U,U,U,U,U,U,8,U,U,U,U,U,U,8
Tri-County Commuter Rail Authority ,"Miami, FL",
U,U,U,U,U,U,18,U,U,U,U,U,U,18
```

```
type OptInt = unavailable of 'U'
            |   available of Int

type counts = OptInt[14]
```

Arrays/Lists

```
Southern California Regional Railroad Authority,"Los Angeles, CA",
U,45,46,46,47,49,51,U,45,46,46,47,49,51
Connecticut Department of Transportation ,"New Haven, CT",
U,U,U,U,U,U,8,U,U,U,U,U,U,8
Tri-County Commuter Rail Authority ,"Miami, FL",
U,U,U,U,U,U,18,U,U,U,U,U,U,18
```

```
type OptInt = unavailable of 'U'
            |   available of Int

type counts = OptInt[14] sep(',')
```

Arrays/Lists

```
Southern California Regional Railroad Authority,"Los Angeles, CA",
U,45,46,46,47,49,51,U,45,46,46,47,49,51
Connecticut Department of Transportation ,"New Haven, CT",
U,U,U,U,U,U,8,U,U,U,U,U,U,8
Tri-County Commuter Rail Authority ,"Miami, FL",
U,U,U,U,U,U,18,U,U,U,U,U,U,18
```

```
type OptInt = unavailable of 'U'
            |   available of Int

type counts = OptInt[] sep(',')
                       term(eor)
```

Dependent Types

```
sdw-01.ab.ca – – [16/12/06] "GET /images/fish.gif HTTP/1.0" 200 8552
sdw-01.ab.ca – DBUser [16/12/06] "GET /images/bug.gif HTTP/1.0" 200 1357
64.233.161.99 – – [16/12/26] "GET /images/plex.gif HTTP/1.0" 304 -
69.30.123.195 – – [16/12/2006] "GET /images/adjoint.gif HTTP/1.0" 304 -
```

```
type responseCode = { x : Int | 99 < x < 600}
```

Dependent Types

```
sdw-01.ab.ca – – [16/12/06] "GET /images/fish.gif HTTP/1.0" 200 8552
sdw-01.ab.ca – DBUser [16/12/06] "GET /images/bug.gif HTTP/1.0" 200 1357
64.233.161.99 – – [16/12/26] "GET /images/plex.gif HTTP/1.0" 304 -
69.30.123.195 – – [16/12/2006] "GET /images/adjoint.gif HTTP/1.0" 304 -
```

```
type method = GET | POST | LINK | UNLINK | ...

fun  check(method, major, minor) = ...

type request =
   { method : method,          ' ',
     url    : String(' '), " HTTP/",
     major  : Int,           '.',
     minor  : Int
   } where check(method, major, minor)
```

Type Summary

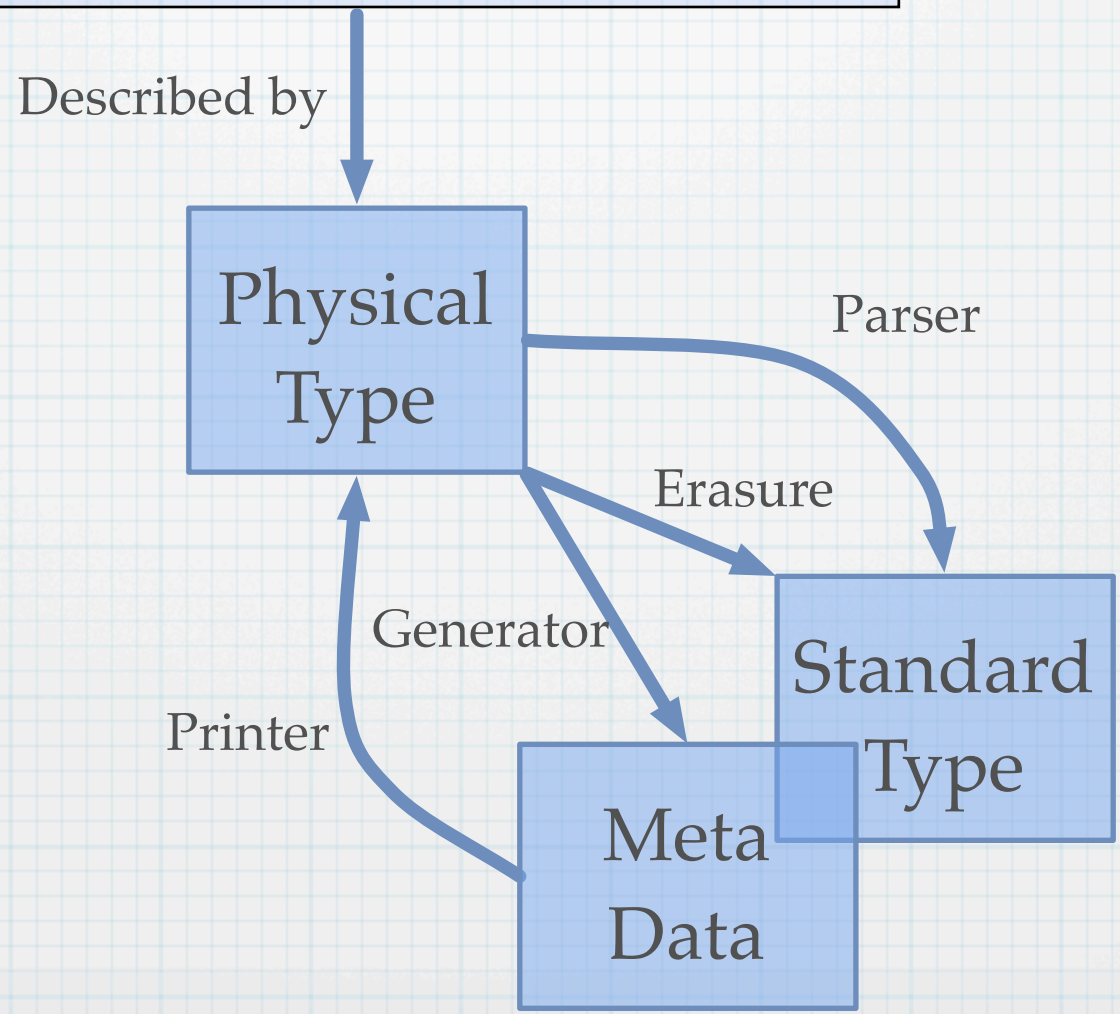
* Base types
* Tuples
* Singleton types
* Records
* Unions
* Lists/Arrays
* Dependent types
* Value abstraction
* Type abstraction
* Recursive types
* Pointers
* ???

Differences

* Data layout is not under the control of the type system.
  * Physical types need some extra information: separators, terminators.
  * Many physical types map to the same internal type: `String(' ')`, `String(':')`, `SBH_uint32`, `B_uint32` ...
* Dependent types much more important for physical types:
  * Missing value representations, value-level constraints, embedded array lengths, union tags.
* We should not assume data conforms 100% to description.

Meta Data

```
"TEAJBUS",197713,-1.948233,Quadrillion Btu,4
"TEAJBUS",197813,-0.336538,Quadrillion Btu,4
"TEAJBUS",197913,-1.649302,Quadrillion Btu,4
"TEAJBUS",198013,-1.0537,Quadrillion Btu,4
```

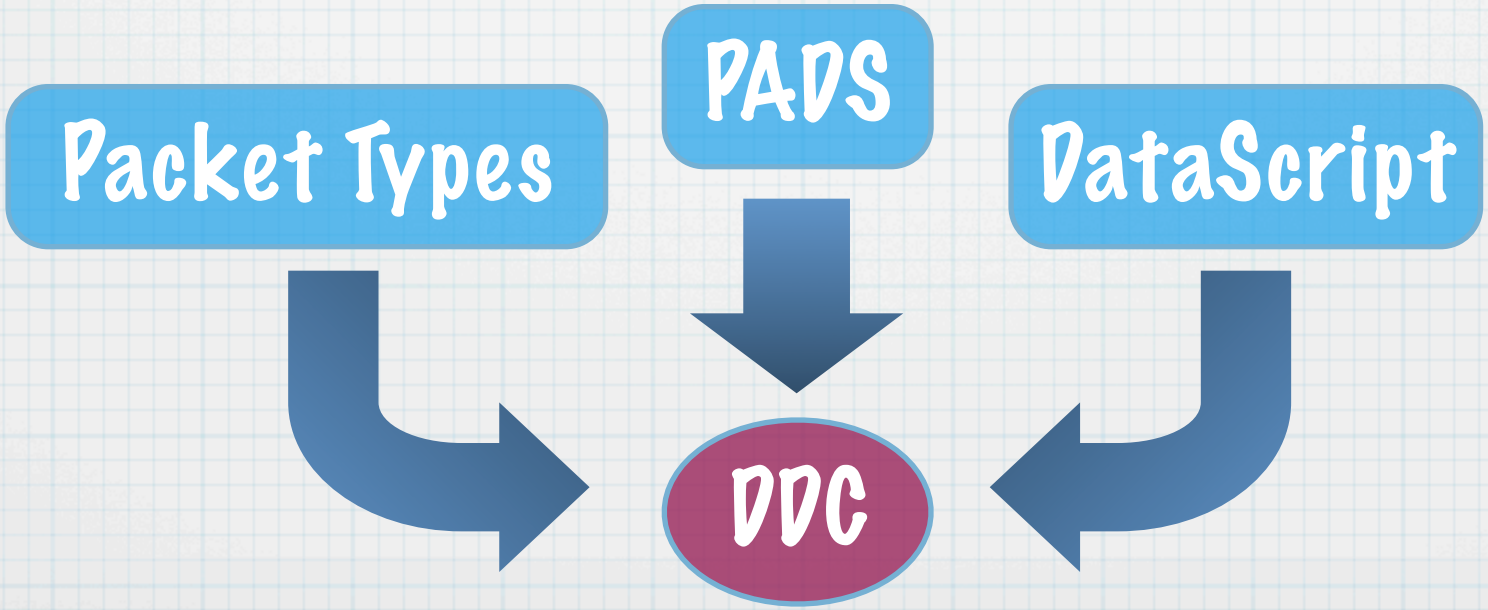


Data Description Languages

* Some examples in practice:
  * **PADS/C** [PLDI '05] and **PADS/ML** [POPL '07]
  * **PacketTypes** [SIGCOMM '98]
  * **DataScript** [GPCE '02]
  * **Erlang**'s bit types [ESOP '04]
  * **DFDL**

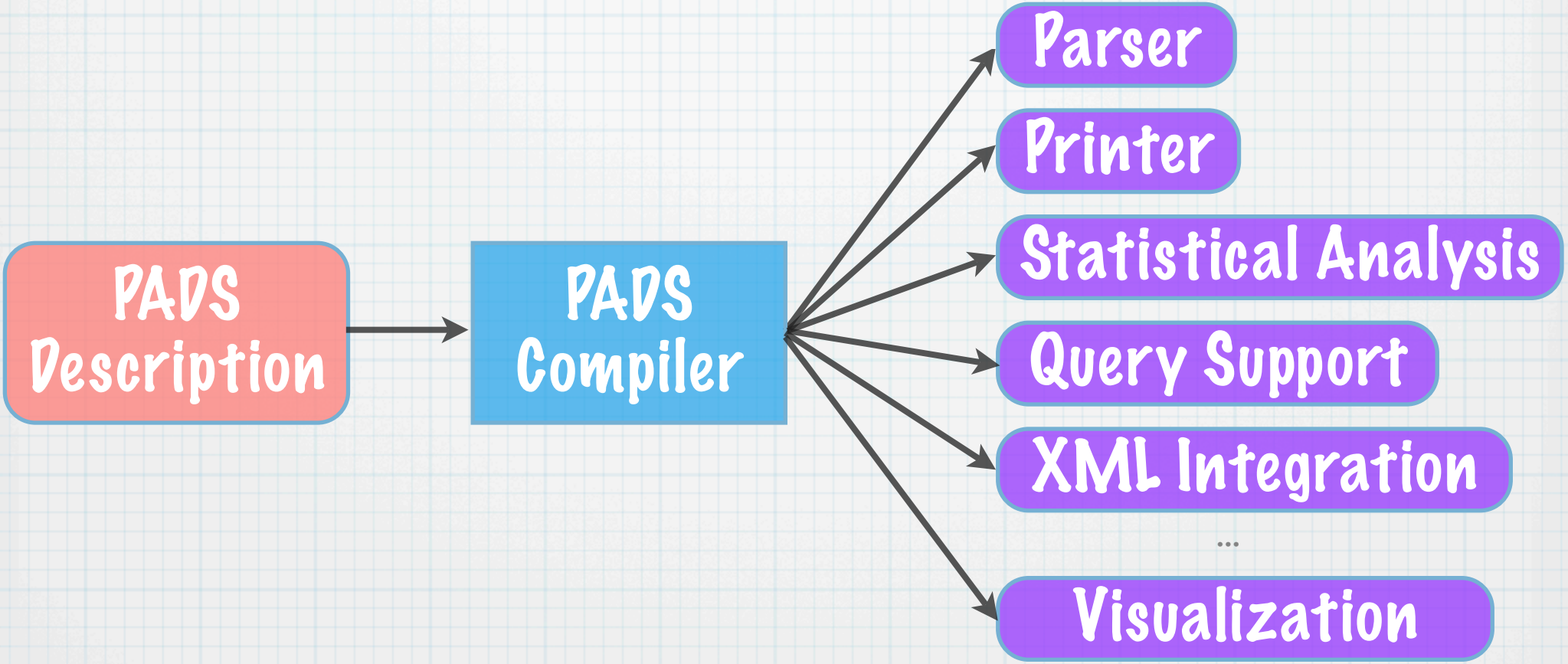
Formal Theory

* A core data description calculus (DDC) [POPL '06]
  * Based on dependent type theory
  * Simple, orthogonal, composable types
  * Types transduce external data source to internal representation.
* Encodings of high-level DDLs in low-level DDC



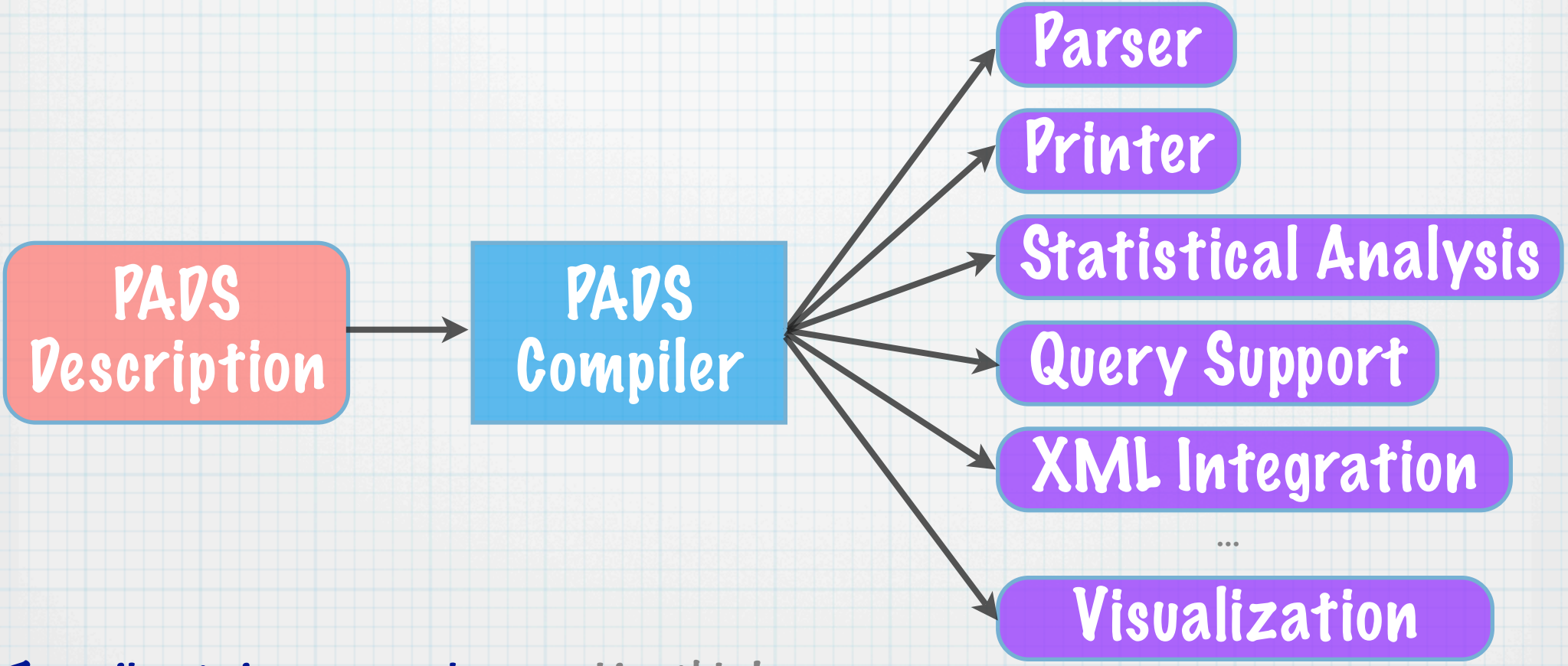
Leverage!

Given a data description, the computer **understands** the data, so we can generate **many** tools from **one** description:



Leverage!

Given a data description, the computer **understands** the data, so we can generate **many** tools from **one** description:



Type-directed programming provides this leverage.
For each base type, we have to specify the desired behavior.
The compiler then lifts the behavior to all structured types.

Statistical Analysis

* Accumulated profile of "leaves" in a data source:

```
<top>.length : uint32
good: 53544   bad: 3824    pcnt-bad: 6.666
min: 35  max: 248591  avg: 4090.234
top 10 values out of 1000 distinct values:
tracked 99.552% of values
 val:  3082 count:  1254  %-of-good:  2.342
 val:   170 count:  1148  %-of-good:  2.144
                . . .

. . . . . . . . . . . . . . . . . . . . . .
 SUMMING    count:  9655  %-of-good: 18.032
```

Not all lengths were legal!

* AT&T uses to get "bird's eye" view of 4000 daily feeds, to vet data, and to debug PADS descriptions.

Pretty Printer

* Customizable program to reformat data:

```
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
```

Normalize time zones
Normalize delimiters

Drop unnecessary values
Filter/repair errors

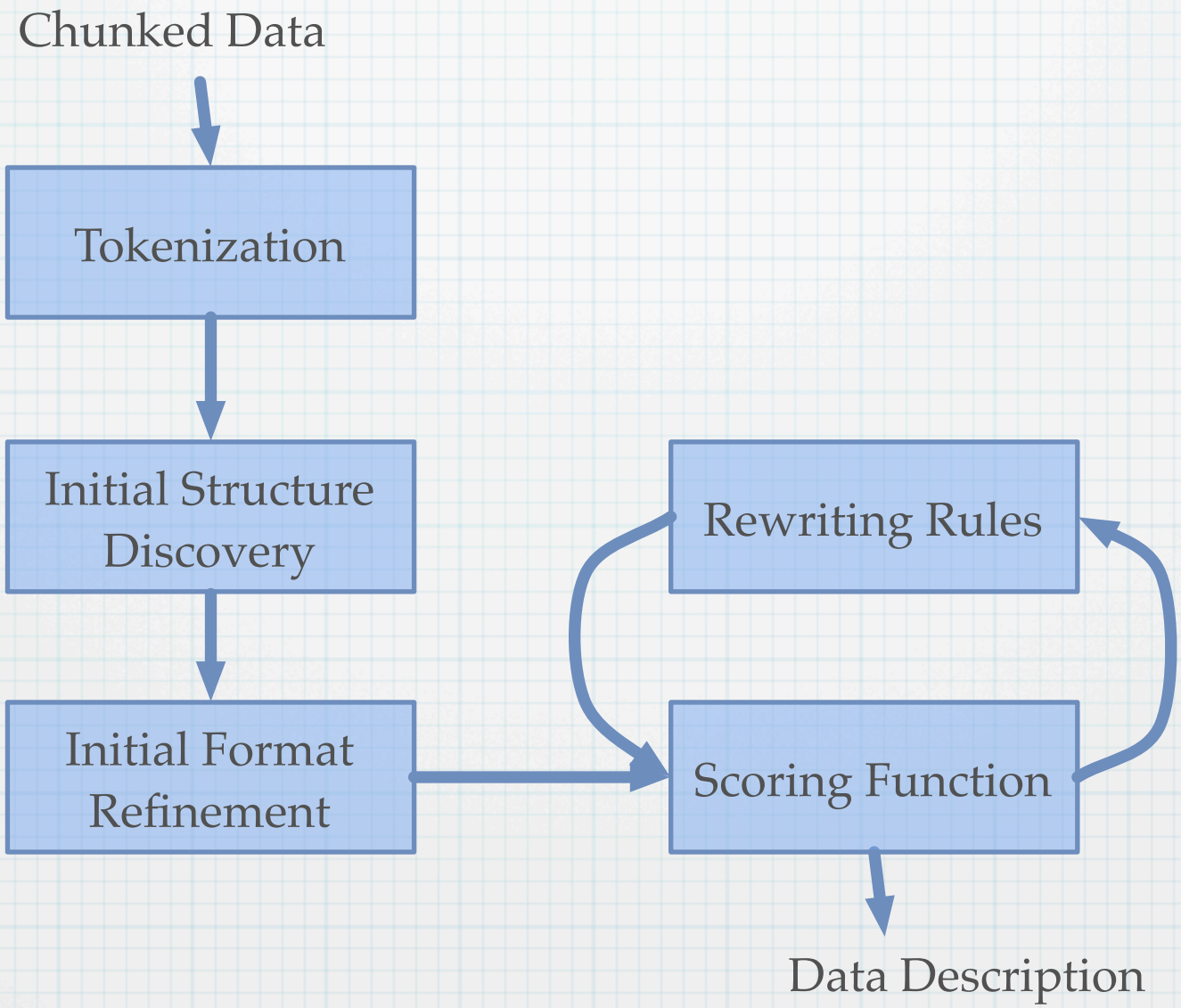
```
207.136.97.49|-|-|10/16/97:01:46:51|GET|/tk/p.txt|1|0|200|30
tj62.aol.com|-|-|10/16/97:21:32:22|POST|/scpt/dd@grp.org/confirm|1|0|200|941
```

* Users can override printing on a per type basis.
* Used at AT&T to normalize monitoring data before loading into a relational database.

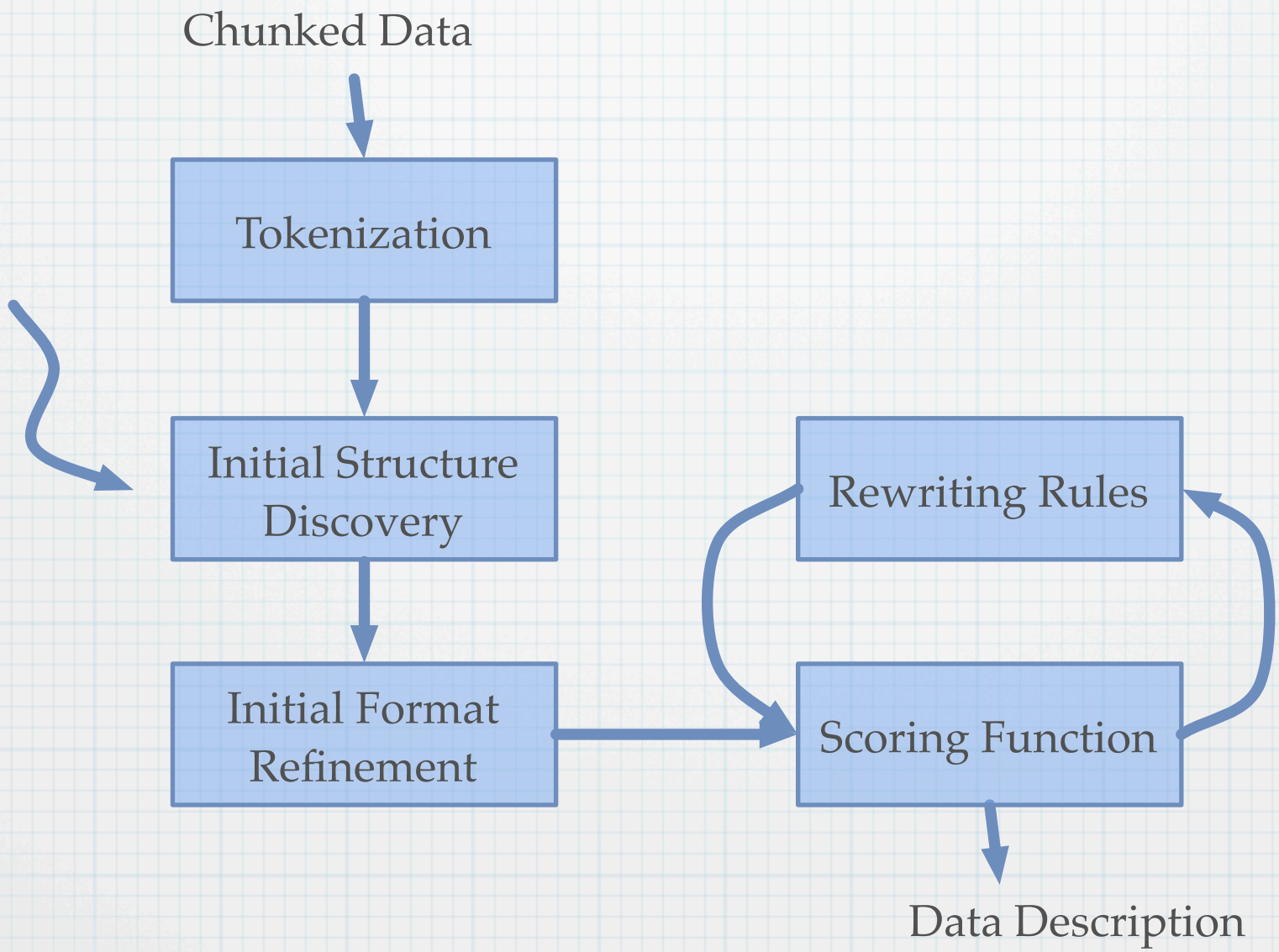
Other Relevant PL Ideas

* Analyses to determine "well-formedness" of descriptions:
  * Do union branches overlap?
  * When do printing and parsing compose [Brabrand, et al, DBPL '05]?
  * What is the on-disk size?
* Type-directed programming
  * Support user-defined tools and transformations
* Structural subtyping?
  * Generate conversion from one format to another
* Type equality?
  * Semantic basis for rewriting descriptions (simpler, shredded,...)
* Type inference?

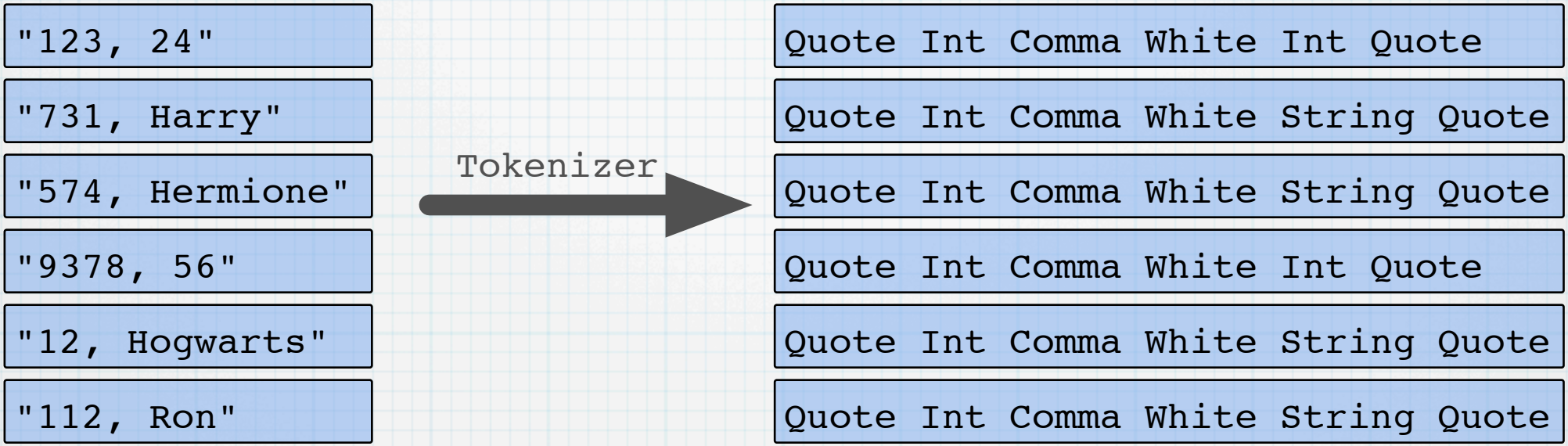
Physical Type Inference



Physical Type Inference



Tokenization

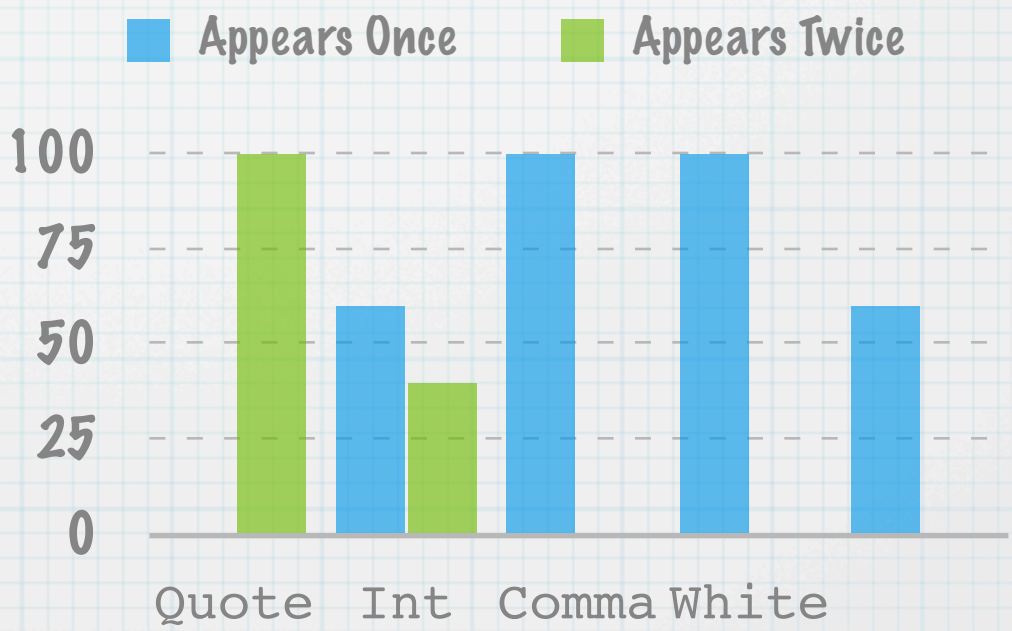


- Tokens expressed as regular expressions.
- Basic tokens
  - Integer, white space, punctuation, strings
- Distinctive tokens
  - IP addresses, dates, times, MAC addresses, ...

Histograms

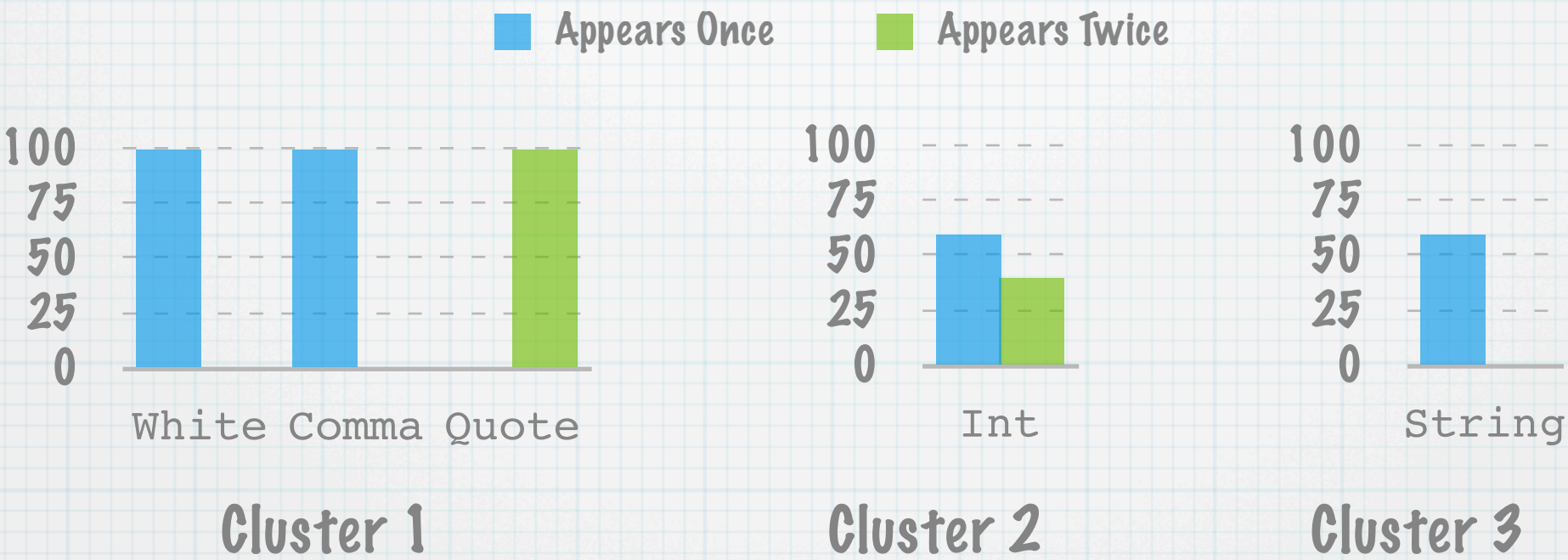
```
Quote Int Comma White Int Quote
Quote Int Comma White String Quote
Quote Int Comma White String Quote
Quote Int Comma White Int Quote
Quote Int Comma White String Quote
Quote Int Comma White String Quote
```

Frequency Analysis



Clustering

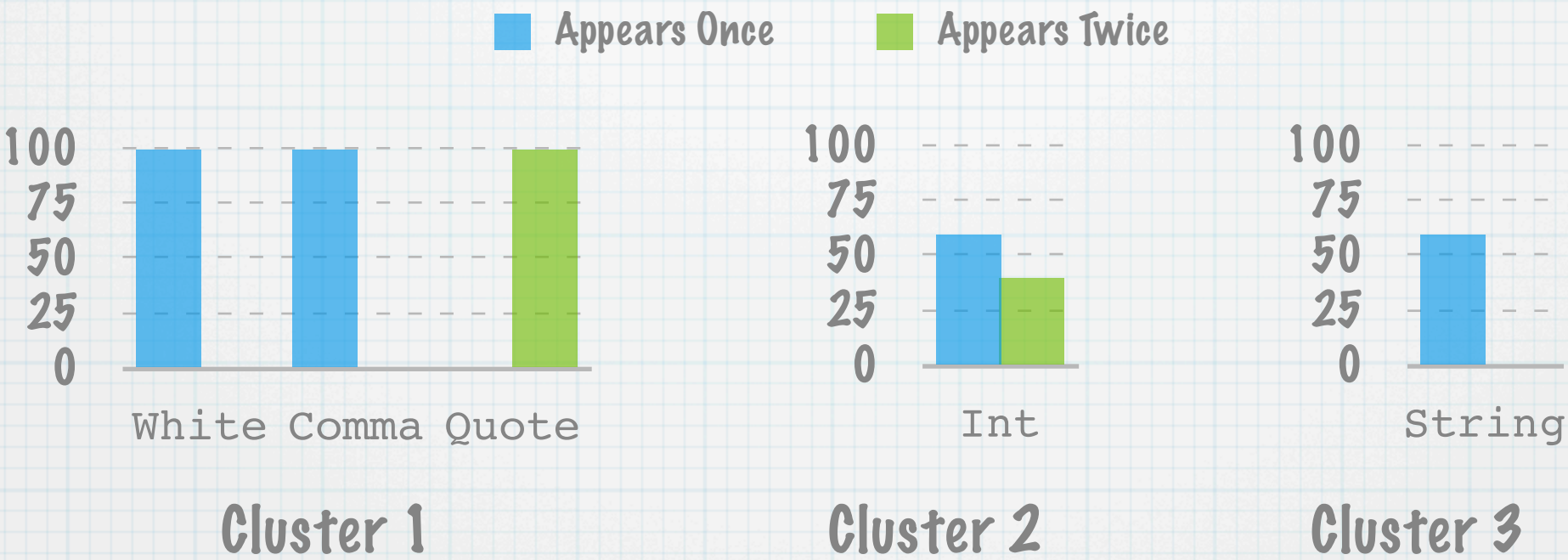
Group clusters with similar frequency distributions



Two frequency distributions are similar if they have the same shape (within some error tolerance) when the columns are sorted by height.

Clustering

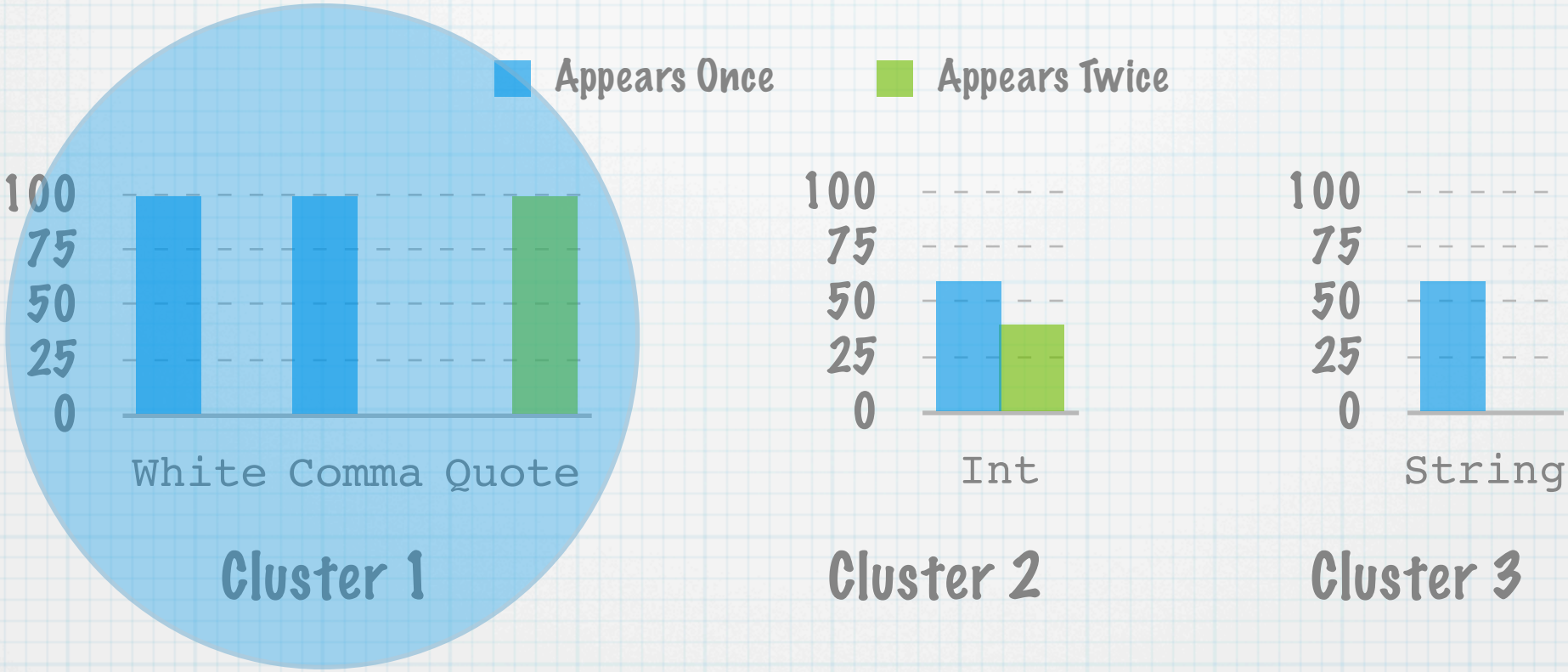
Group clusters with similar frequency distributions



Rank clusters by metric that rewards high coverage and narrower distributions. Chose cluster with highest score.

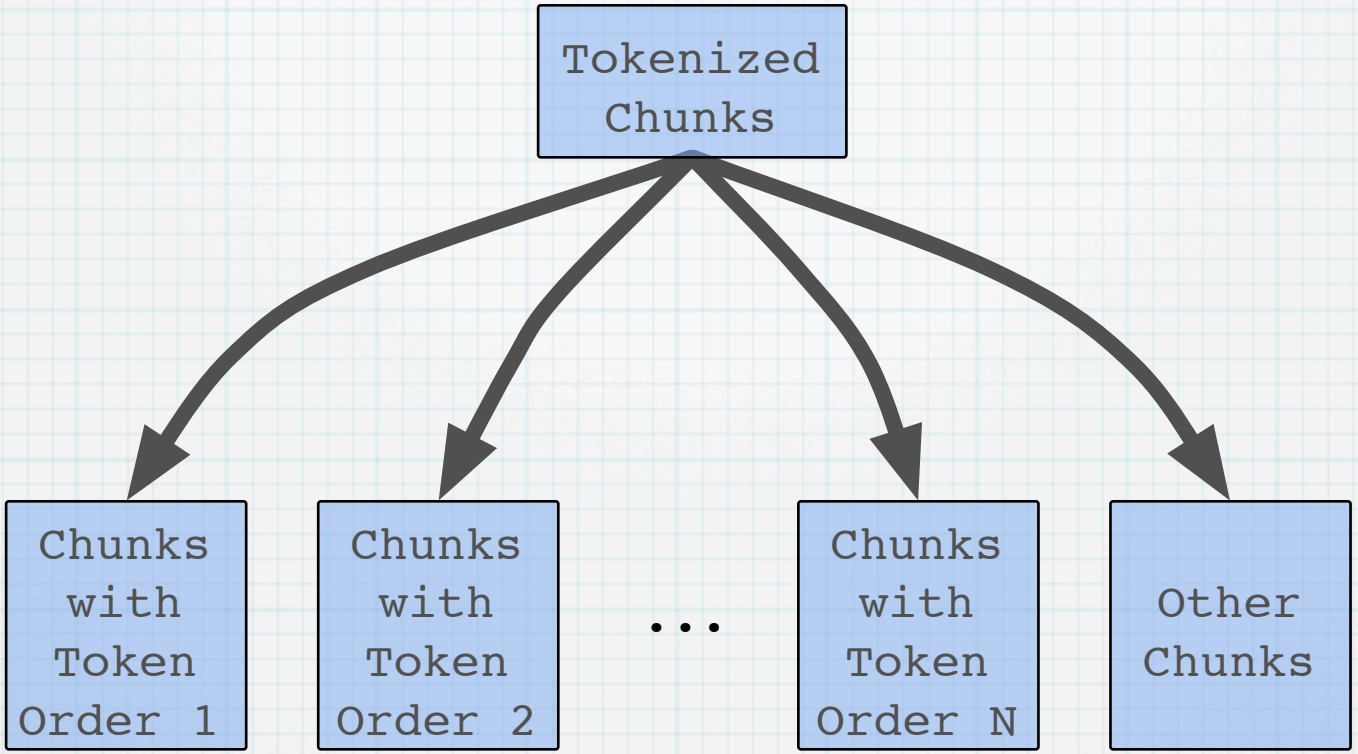
Clustering

Group clusters with similar frequency distributions



Rank clusters by metric that rewards high coverage and narrower distributions. Chose cluster with highest score.

Partition Chunks



$$\tau_1 + \tau_2 \ldots + \ldots \tau_N + \tau_{other}$$

In our example, all the tokens appear in the same order in all chunks, so the union is degenerate.

Find Subcontexts

Quote Int **Comma** **White** Int **Quote**

Quote Int **Comma** **White** String **Quote**

Quote Int **Comma** **White** String **Quote**

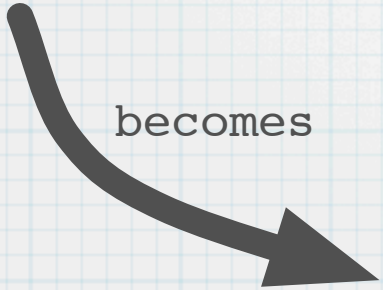
Quote Int **Comma** **White** Int **Quote**

Quote Int **Comma** **White** String **Quote**

Quote Int **Comma** **White** String **Quote**

Tokens in selected cluster:

Quote(2) Comma White



Quote *

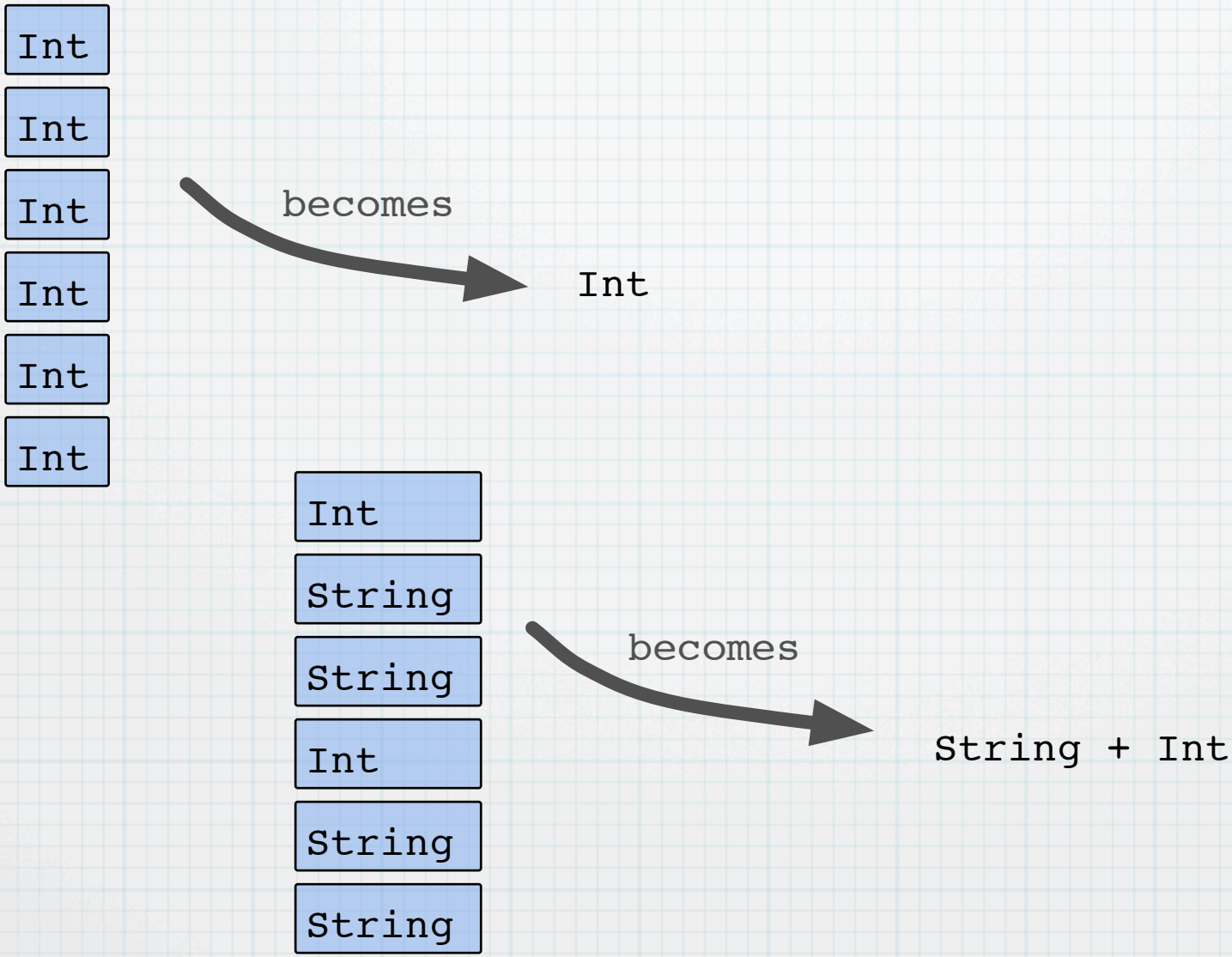
Int
Int
Int
Int
Int
Int

* Comma * White *

Int
String
String
Int
String
String

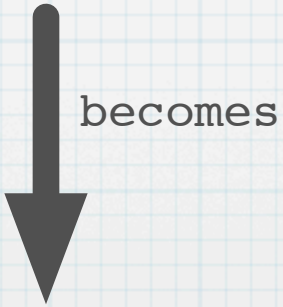
* Quote

Then Recurse...



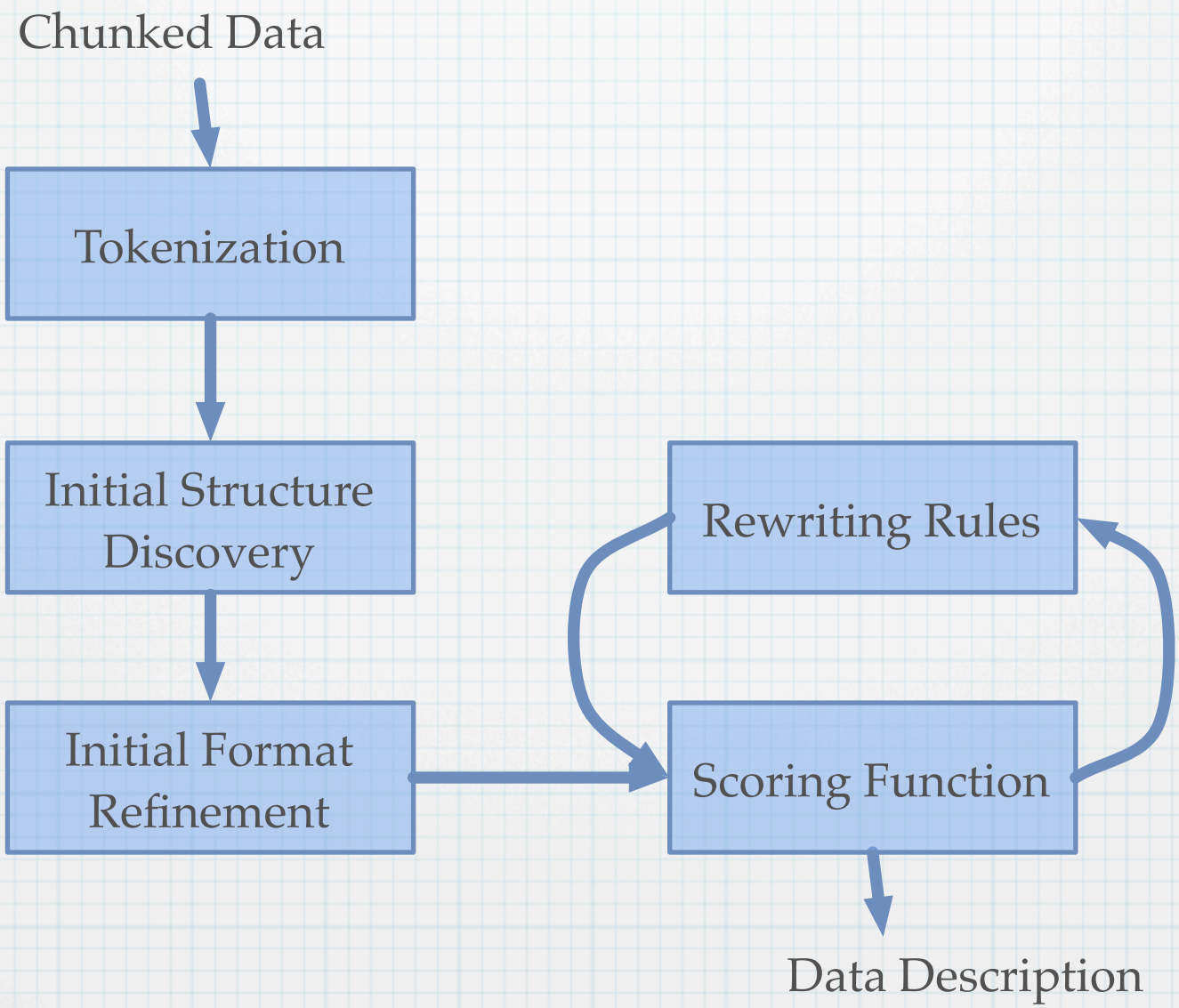
Inferred Type

```
"123, 24"
"731, Harry"
"574, Hermione"
"9378, 56"
"12, Hogwarts"
"112, Ron"
```




```
Quote * Int * Comma * White * (String + Int) * Quote
```

Physical Type Inference



Inferring PADS Descriptions

http://www.galois.com/LearnPADS/cgi-bin/learning-demo.cgi?userid=id.1221700843

Google

Inferring PADS Descriptions

PADS Learning Demo

Resources

- PADS Home
- Learning Demo Home
- Data Formats
- Machine Description
- Papers

Bug Reports

We are building a system to infer PADS descriptions of ad hoc data formats and to generate tools to manipulate such data automatically. We currently build (1) a tool for converting ad hoc data into a canonical form of XML with a corresponding XSchema, (2) a tool for converting ad hoc data into a more regular form that may be suitable for loading into a relational system such as a database or an Excel spreadsheet, and (3) a statistical analysis tool we call an *accumulator*.

To try a demo, select one of the ad hoc formats below. Pressing submit will cause the learning software to process the selected format, returning the example data and the inferred description on the resulting page. From there, you will be able to run any of the generated tools. The 'Roll your own' selection lets you enter your own data.

Computing the description may take a minute or so, depending upon the speed of the machine hosting the demo and the complexity of the data.

Data sources

ai.3000
asl.log
boot.log
crashreporter.log
crashreporter.log.modified
ls-l.txt
netstat-an
page_log
quarterlypersonalincome
railroad.txt
scrollkeeper.log
windowserver_last.log
yum.txt
1967Transactions.short
MER_T01_01.csv
Roll your own

Submit

This work is the product of a collaboration between AT&T, Princeton University, and Galois.

It was partially supported by DARPA and the NSF.

Results for crashreporter.log

http://www.galois.com/LearnPADS/cgi-bin/data-results.cgi?datasource=crashreporter.log&Submit=Si

Google

Results for crashreporter.log

PADS Learning Demo

Available Outputs

- PADS Description
- XML
- Reformatted output
- Accumulator

Other Resources

- PADS Home
- Learning Demo Home
- Data Formats
- Machine Description
- Papers

Bug Reports

Input Data: crashreporter.log

```
Wed Jun 21 10:24:44 2006 crashdump[1524]: crashdump started
Wed Jun 21 10:24:46 2006 crashdump[1524]: Started writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Preview.
Wed Jun 21 10:24:47 2006 crashdump[1524]: Finished writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Previev
Fri Jun 23 17:51:55 2006 crashdump[1892]: crashdump started
Fri Jun 23 17:51:58 2006 crashdump[1892]: Started writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Microsoft
Fri Jun 23 17:52:02 2006 crashdump[1892]: Finished writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Microsof
Sat Jun 24 06:38:46 2006 crashdump[2165]: crashdump started
Sat Jun 24 06:38:46 2006 crashdump[2165]: Unable to determine task_t for pid: 95 name: Exited process
Sat Jun 24 06:38:46 2006 crashdump[2164]: crashdump started
Sat Jun 24 06:38:46 2006 crashdump[2164]: Unable to determine task_t for pid: 95 name: Exited process
```

Generated PADS description

```
    Pcase mach_msg31:   Struct_221  v_struct_221_2;
 }
};
Precord Pstruct Struct_222 {
    PPdate  v_date_1;
    ' ';
    PPtime  v_time_6;
    ' ';
    Puint16  v_intconst11 : v_intconst11 == 2006;
    ' ';
    Enum_15  v_enum_15;
    '[';
    Puint16  v_intrange_21;
    ']: ';
    Enum_31  v_enum_31;
    Switch_116 (:v_enum_31:) v_switch_116;
};
Psource Parray entries_t {
    Struct_222[];
};
```

We are compiling the generated library in the background. The compilation is finished when the page stops loading.

The links in the navigation pane let you run the generated tools. It may take several minutes to compile the generated library.

Results for crashreporter.log

http://www.galois.com/LearnPADS/cgi-bin/data-results.cgi?datasource=crashreporter.log&Submit=Su

Google

Results for crashreporter.log

PADS Learning Demo

Input data: crashreporter.log

```
Wed Jun 21 10:24:44 2006 crashdump[1524]: crashdump started
Wed Jun 21 10:24:46 2006 crashdump[1524]: Started writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Preview.
Wed Jun 21 10:24:47 2006 crashdump[1524]: Finished writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Previev
Fri Jun 23 17:51:55 2006 crashdump[1892]: crashdump started
Fri Jun 23 17:51:58 2006 crashdump[1892]: Started writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Microsoft
Fri Jun 23 17:52:02 2006 crashdump[1892]: Finished writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Microsof
Sat Jun 24 06:38:46 2006 crashdump[2165]: crashdump started
Sat Jun 24 06:38:46 2006 crashdump[2165]: Unable to determine task_t for pid: 95 name: Exited process
Sat Jun 24 06:38:46 2006 crashdump[2164]: crashdump started
Sat Jun 24 06:38:46 2006 crashdump[2164]: Unable to determine task_t for pid: 95 name: Exited process
```

XML representation of the data


```
<Struct_222>
 <v_date_1><val>Wed Jun 21</val></v_date_1>
 <v_time_6><val>10:24:44</val></v_time_6>
 <v_intconst11><val>2006</val></v_intconst11>
 <v_enum_15><val>crashdump</val></v_enum_15>
 <v_intrange_21><val>1524</val></v_intrange_21>
 <v_enum_31><val>crashdump</val></v_enum_31>
 <v_switch_116>
  <v_struct_221_1>
   <v_opt_127>
   </v_opt_127>
   <v_enum_139><val>started</val></v_enum_139>
   <v_opt_144>
   </v_opt_144>
  </v_struct_221_1>
 </v_switch_116>
</Struct_222>
<Struct_222>
 <v_date_1><val>Wed Jun 21</val></v_date_1>
 <v_time_6><val>10:24:46</val></v_time_6>
 <v_intconst11><val>2006</val></v_intconst11>
 <v_enum_15><val>crashdump</val></v_enum_15>
 <v_intrange_21><val>1524</val></v_intrange_21>
 <v_enum_31><val>Started</val></v_enum_31>
 <v_switch_116>
  <v_struct_115_2>
   <v_enum_35><val>writing</val></v_enum_35>
```


Available Outputs

- PADS Description
- XML
- Reformatted output
- Accumulator

Other Resources

- PADS Home
- Learning Demo Home
- Data Formats
- Machine Description
- Papers

Bug Reports

Results for crashreporter.log

http://www.galois.com/LearnPADS/cgi-bin/data-results.cgi?datasource=crashreporter.log&Submit=Su

Google

Results for crashreporter.log

PADS Learning Demo

Available Outputs

- PADS Description
- XML
- Reformatted output
- Accumulator

Other Resources

- PADS Home
- Learning Demo Home
- Data Formats
- Machine Description
- Papers

Bug Reports

Input data: crashreporter.log

```
Wed Jun 21 10:24:44 2006 crashdump[1524]: crashdump started
Wed Jun 21 10:24:46 2006 crashdump[1524]: Started writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Preview.
Wed Jun 21 10:24:47 2006 crashdump[1524]: Finished writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Previev
Fri Jun 23 17:51:55 2006 crashdump[1892]: crashdump started
Fri Jun 23 17:51:58 2006 crashdump[1892]: Started writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Microsoft
Fri Jun 23 17:52:02 2006 crashdump[1892]: Finished writing crash report to: /Users/kfisher/Library/Logs/CrashReporter/Microsof
Sat Jun 24 06:38:46 2006 crashdump[2165]: crashdump started
Sat Jun 24 06:38:46 2006 crashdump[2165]: Unable to determine task_t for pid: 95 name: Exited process
Sat Jun 24 06:38:46 2006 crashdump[2164]: crashdump started
Sat Jun 24 06:38:46 2006 crashdump[2164]: Unable to determine task_t for pid: 95 name: Exited process
```

Accumulator report

```
Data file = ../data/crashreporter.log

warning: P_open: Installing default IO discipline : newline-terminated records

*****************************************************************************************
<top> : struct Struct_222
*****************************************************************************************
good vals:      441   bad vals:       0   pcnt-bad:   0.000

[Describing each field of <top>]

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
<top>.v_date_1 : typedef PPdate [--> Pstring_ME]
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
String lengths: 441 string length values, 100 pcnt good, 100 pcnt identical:       10

 Characterizing string values:
  => distribution of top 10 strings out of 45 distinct strings:
     val: "Mon Aug  7" count:      90  pcnt-of-good-vals:  20.408
     val: "Sun Oct 15" count:      33  pcnt-of-good-vals:   7.483
     val: "Fri Sep  8" count:      27  pcnt-of-good-vals:   6.122
     val: "Tue Aug  1" count:      27  pcnt-of-good-vals:   6.122
     val: "Wed Aug 23" count:      24  pcnt-of-good-vals:   5.442
     val: "Fri Sep 29" count:      18  pcnt-of-good-vals:   4.082
     val: "Mon Jul 31" count:      18  pcnt-of-good-vals:   4.082
     val: "Tue Aug 15" count:      18  pcnt-of-good-vals:   4.082
```

Related Work

* Grammar Induction
  * Extracting Structure from Web Pages [Arasu & Hector-Molena, SigMod, 2003].
  * Language Identification in the Limit [Gold, Information and Control, 1968].
  * Grammatical Inference for Information Extraction and Visualization on the Web [Hong, PhD Thesis, Imperial College, 2003].
  * Current Trends in Grammatical Inference [Higuera, LNCS, 2001].
* Functional dependencies
  * Tane: An Efficient Algorithm for Discovering Functional and Approximate Dependencies [Huhtal et al, Computer Journal, 1999].
* Information Theory
  * Information Theory, Inference, and Learning Algorithms [Mackay, Cambridge University Press, 2003].
  * Advances in Minimum Description Length [Grünwald, MIT Press, 2004].

Conclusions

* Ad hoc data is pervasive and difficult to deal with.
* Data description languages can help!
* Programming language ideas are highly relevant:
  * **Types** to describe physical data.
  * **Type-directed programming** to generate tools automatically.
  * **Program analysis** to discover properties of descriptions.
  * **Formal semantics** to specify the meaning of descriptions.
  * **Type inference** to learn descriptions from raw data.

Thank You!

* David Walker
  Princeton
* Kenny Zhu
  Princeton
* Yitzhak Mandelbaum
  AT&T
* Robert Gruber
  Google
* David Burke
  Galois
* Peter White
  Galois
* and many others...

Try it!

www.padsproj.org