Generic tools for *ad hoc* data: PADX, a case study

Michael Greenberg, *University of Pennsylvania*
Mary Fernández and Yitzhak Mandelbaum, *AT&T Research*

*plus:*

Kathleen Fisher, *AT&T Research*,
J. Nathan Foster, *University of Pennsylvania*
ad hoc data formats:
• configuration files
• AT&T proprietary data formats
• log files

ad hoc data formats aren't going to disappear

how can we use them?
PADS/ML: type-based data descriptions

```
ip vrf 1023
  description ANTI-PESTO S.W.A.T. TEAM
  export map To_NY_VPN
  route-target 100:3
  maximum routes 150 80
```

```
pctype ip_vrf_command =
  Description of "description " * pstring('|') * '|
| Export of "export map " * pstring('
')
| Route_target of "route-target " * pint * ':': * pint
| Max_routes of "max routes " * pint * ' ' * pint

pctype ip_vrf = {
  header : "ip vrf " * pint * '
';
  commands : ip_vrf_command plist('
')
}
```
using data descriptions

PADS/ML description

PADS/ML compiler

- parser
- printer
- analysis
- error scrubbing
- XML export
- XML import
- XML schema
ip vrf 1023
description ANTI-PESTO S.W.A.T. TEAM|
export map To_NY_VPN
route-target 100:3
maximum routes 150 80

ptype ip_vrf_command =
  Description of "description " * ...
  | Export of "export map " * ...
  | Route_target of "route-target " * ...
  | Max_routes of "max routes " * ...

ptype ip_vrf = {
  header: "ip vrf " * pint * '\n';
  commands: ip_vrf_command plist(\'\n\')
}

PADS/ML representation

{   header: 1023,
    commands: [Description "ANTI-PESTO S.W.A.T. TEAM";
       Export "To_NY_VPN";
       Route_target (100, 3);
       Max_routes (150, 80)] }
XML representation

```xml
<ip_vrf>
  <header><val>1023</val></header>
  <commands>
    <elt><Description><val>ANTI-PESTO S.W.A.T. TEAM</val></Description></elt>
    <elt><Export><val>To_NY_VPN</val></Export></elt>
    <elt><Route_target>
      <elt1><val>100</val></elt1>
      <elt2><val>3</val></elt2>
    </Route_target></elt>
    <elt><Max_routes>
      <elt1><val>150</val></elt1>
      <elt2><val>80</val></elt2>
    </Max_routes></elt>
  </commands>
</ip_vrf>
```
Galax and XQuery

```
<route_limits>{.//Max_routes}</route_limits>

for $ip_vrf in ...
return <ip_vrf>
  <header><val>{$ip_vrf/header/val}</val></header>
  <commands>{
    for $cmd in $ip_vrf/commands/elt
    where $cmd/Max_routes/elt1/val > 110
    return $cmd
  }</commands>
</ip_vrf>
```
one way or round-trip?

```
ip vrf 1023
  description ANTI-PESTO S.W.A.T. TEAM
  export map To_NY_VPN
  route-target 100:3
  maximum routes 150 80
```

```
<ip_vrf>
  ...
</ip_vrf>
```
one way or round-trip?

```
type ip_vrf_command = ...
type ip_vrf = ...
ip vrf 1023
description ANTI-PESTO S.W.A.T. TEAM|
export map To_NY_VPN
route-target 100:3
maximum routes 150 80
```
PADX tool framework

- PADS/ML compiler
- Parser
- XML export
- XML import
- XML schema
- Printer
- Galax
- XQuery

PADS/ML description
generic tool framework

PADS/ML description

PADS/ML compiler

generic tool framework

parser

printer

analysis

error scrubbing

XML export

XML import

XML schema
generic tool framework

different tool types
  how can one interface work for
all PADS types and all PADS tools?
generic tool framework

each tool has a type in terms of:
  • some PADS/ML data representation
  • some PADS/ML PD
  • something else

examples:
  • (rep, pd) -> Galax.DM_node
  • Galax.DM_node -> (rep, pd)
  • (rep, pd) -> (rep, pd)
generic tool framework

each type is defined in terms of:

- base types, e.g., `int`, `float`
- product types, e.g., `ip_vrf` (arrays, structs in C)
- sum types, e.g., `ip_vrf_command` (unions, enums in C)
- lists and streams
- constraints, e.g., `[ age:pint | age > 21 ]`

tools are written as a type-directed case analysis

see [Yang 1998] and [Hinze 2004]
SO:

now we can write:

- $(\text{rep, pd}) \rightarrow \text{Galax.DM\_node}$
- $\text{Galax.DM\_node} \rightarrow (\text{rep, pd})$

without:

- constraining ourselves to a specific PADS/ML description
- changing the compiler
- expert knowledge of PADS/ML internals
future work

PADX: viewing and querying *ad hoc* data with XQuery
  • streaming and laziness
  • generalization of data sources
  • real-world use!

generic tools for PADS/ML
  • varying levels of granularity
  • dependent types and constraints
    
    \{ size: pint; content: pstring_FW(size) \}
  
  • theorems
round-trip?

```xml
<ip_vrf>
    <header><val>1023</val></header>
    <commands>
        <elt><Description><val>ANTI-...</val></Description></elt>
        <elt><Export><val>To_NY_VPN</val></Export></elt>
        <elt><Route_target>
            <elt1><val>100</val></elt1>
            <elt2><val>3</val></elt2>
        </Route_target></elt>
        <elt><Max_routes>
            <elt1><val>150</val></elt1>
            <elt2><val>80</val></elt2>
        </Max_routes></elt>
    </commands>
</ip_vrf>

<ip_vrf_command>
    <Max_routes>
        <elt1><val>150</val></elt1>
        <elt2><val>80</val></elt2>
    </Max_routes>
</ip_vrf_command>
```
round-trip type detection: XMLSchema

```plaintext
ptype ip_vrf_command =
    Description of "description " * pstring('|') * '|
| Export of "export map " * pstring('
')
| Route_target of "route-target " * pint * ':' * pint
| Max_routes of "max routes " * pint * ' ' * pint

ptype ip_vrf = {
    header : "ip vrf " * pint * '
';
    commands : ip_vrf_command plist('
')
}

<xs:schema ...>
    <xs:element name="ip_vrf_command" type=... />
    <xs:element name="ip_vrf" type=... />
    ...
</xs:schema>
```
run-time type representations

each tool has a type \((\text{rep}, \text{pd}, \alpha)\) tool

the truth, though Ocaml can't bear to hear it: the tool-type \((\text{rep}, \text{pd}, \alpha)\) tool really is:

\[
T \to (\text{rep}, \text{pd}, \alpha) \text{ tool}
\]

\(T\) is the type of PADS/ML types

“type representations” let us realize this