Teaching old type systems new tricks with type providers

Tomas Petricek
University of Kent and The Alan Turing Institute
http://tomasp.net | tomas@tomasp.net | @tomaspetricek
DATA SCIENCE
A Visual History of Which Countries Have Dominated the Summer Olympics

By GREGOR AICH and LARRY BUCHANAN  UPDATED August 22, 2016
DEMO
Open, reproducible data visualizations
Tooling for data science
The gap between spreadsheets and programming
Tooling for data science

Making programming languages a bit easier
Tooling for data science

Learning from spreadsheet interaction model
Reading data
Unsafe dynamic access in a typed language

```csharp
var url = "http://dvd.netflix.com/Top100RSS";
var rss = XDocument.Load(topRssFeed);
var channel = rss.Element("rss").Element("channel");

foreach(var item in channel.Elements("item")) {
    Console.WriteLine(item.Element("text").Value);
}
```

Not found!
Reading data
Unsafe dynamic access in a typed language

```csharp
var url = "http://dvd.netflix.com/Top100RSS";
var rss = XDocument.Load(topRssFeed);
var channel = rss.Element("rss").Element("channel");

foreach(var item in channel.Elements("item")) {
    Console.WriteLine(item.Element("title").Value);
}
```
Reading data

Accessing data from external data sources

Languages do not understand data

There is rarely explicit schema

Manually define types to capture it

Easier in dynamic languages
Aggregating data
Athletes by number of gold medals from Rio 2016

```python
olympics = pd.read_csv("olympics.csv")
olympics[olympics["Games"] == "Rio (2016)"]
    .groupby("Athlete")
    .agg({"Gold": sum})
    .sort_values(by="Gold", ascending=False)
    .head(8)
```
Aggregating data

Language and data source features you need to know

- Python dictionaries \{"key": value\}
- Generalised indexers .[ condition ]
- Operation names sort_values
- Data column names "Athlete"
TYPE PROVIDERS
\[ \emptyset \vdash e : \tau \]
\[ \pi(\emptyset) \vdash e : \tau \]
DEMO
Reading data from an RSS feed
F# Data library

Type providers for structured data

- Structural shape inference
- Language integration via type providers
- Relative type safety
{title: string, author: {age: int}}

{author: {age: float}}

{ title: option<string>, author: {age: float} }
{ coordinates : {lng:num, lat:num} } + string
Shape inference

Pragmatic design choices for usability

- Prefers records for tooling
- Predictable and stable
- Open world assumption about sums
DEMO
Aggregating Olympic medalists
Dot-driven development

Encoding complex logic via simple member access

- Type providers for member generation
- Laziness for scaling to large hierarchies
- Fancy types for the masses!
Row types and phantom types

Row types to track names and types of fields

\[ \Gamma \vdash e : [f_1 : \tau_1, \ldots, f_n : \tau_n] \]
\[ \Gamma \vdash e.\ drop\ f_i : [f_1 : \tau_1, \ldots, f_{i-1} : \tau_{i-1}, f_{i+1} : \tau_{i+1}, \ldots, f_n : \tau_n] \]

Embed row types in provided nominal types

\[ \Gamma \vdash e : C_1 \]
\[ \Gamma \vdash e.\ drop\ f_i : C_2 \quad \text{where} \]

\[ \text{fields}(C_1) = \{ f_1 : \tau_1, \ldots, f_n : \tau_n \} \]
\[ \text{fields}(C_2) = \{ f_1 : \tau_1, \ldots, f_{i-1} : \tau_{i-1}, f_{i+1} : \tau_{i+1}, \ldots, f_n : \tau_n \} \]
Fancy types for the masses!

Powerful idea that works in other contexts

- Row types and phantom types
- Session types for communication
- Add your own fancy type here!
BEHIND THE SCENES
Relative type safety

Well typed programs do not go wrong.
(As long as the world is well-behaved.)
F# Data and safety

Given *representative samples and an input value*

\[ S(d) \sqsubseteq S(d_1, \ldots, d_n) \]

Any program written using a type provider reduces

\[ e_{user}[x \leftarrow \text{new } C(d)] \rightsquigarrow^* v \]
DEMO
Handling schema change and errors
F# Data and schema change
Provided type can change only in limited ways

\[ C[e] \rightarrow C[e. M] \]
\[ C[e] \rightarrow C[\text{match } e \text{ with } \ldots] \]
\[ C[e] \rightarrow C[\text{int}(e)] \]
Structure of a type provider

Context $L$ maps names to definitions and nested contexts

$$L(C) = \text{type } C(x : \tau) = \overline{m}, L'$$

Pivot provider takes schema and provides a class with context

$$\text{pivot}(F) = C, L$$
DEMO
Fancy types in action
Pivot type provider

Generate classes that drop individual columns

\[
\text{drop}(F) = C, \{ C \mapsto (l, L' \cup \bigcup L_f) \}
\]

\[
l = \text{type } C(x : \text{Query}) = \\
\quad \text{member } \langle \text{drop } f \rangle : C_f = C_f(\Pi_{\text{dom}(F')}(x)) \\
\quad \text{member } \text{then} : C' = C''(x)
\]

\[\forall f \in \text{dom}(F) \text{ where } C_f, L_f = \text{drop}(F') \]
and \[F' = \{ f' \mapsto \tau' \in F, f' \neq f \}\]
where \[C', L' = \text{pivot}(F')\]
JSON type provider

Generate class corresponding to a record shape

\[
\text{provide}(\{\nu_1 : \sigma_1, \ldots, \nu_n : \sigma_n\}) = \\
C, \{C \mapsto (l, L_1 \cup \ldots \cup L_n)\}
\]

\[
l = \text{type } C(x_1:\text{Data}) = \\
\text{member } \nu_1 : C_1 = \text{convField}(\nu, \nu_1, x_1, C_1) \quad (\ldots)
\]

\[
\text{member } \nu_n : C_n = \text{convField}(\nu, \nu_n, x_n, C_n)
\]

where \(C_i, L_i = \text{provide}(\sigma_i)\)
SUMMARY
Future work

Making programming with data easier

- Learning from spreadsheets
- Understanding programmer interactions
- Handling joins and data cleaning
- Read, analyse and visualize!
DEMO
Learning from spreadsheet
Thank you!
Teaching old type systems new tricks with type providers

Dot-driven Towards minimal calculus of interactions
Fancy types Encoding row types via type providers
Relative safety Necessity when working with data

Tomas Petricek
✉️ tomas@tomasp.net | @tomaspetricek | tomasp.net/academic
🌐 thegamma.net | fslab.org | gamma.turing.ac.uk
References

Don Syme, Keith Battocchi, Kenji Takeda, Donna Malayeri and Tomas Petricek. **Themes in Information-Rich Functional Programming for Internet-Scale Data Sources**. In proceedings of DDFP 2013

Tomas Petricek, Gustavo Guerra and Don Syme. **Types from data: Making structured data first-class citizens in F#**. PLDI 2016

Tomas Petricek. **Data exploration through dot-driven development**. In proceedings of ECOOP 2017