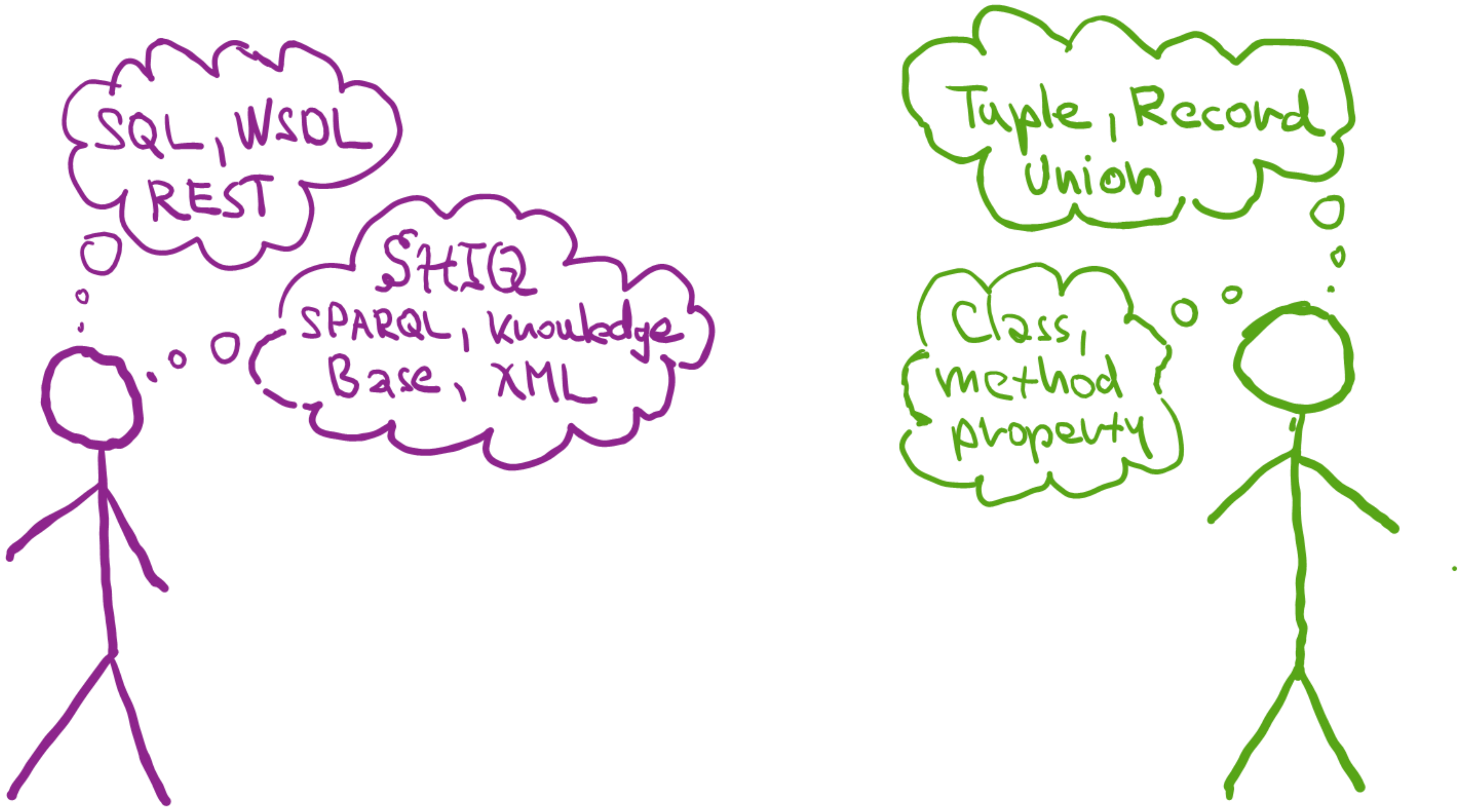
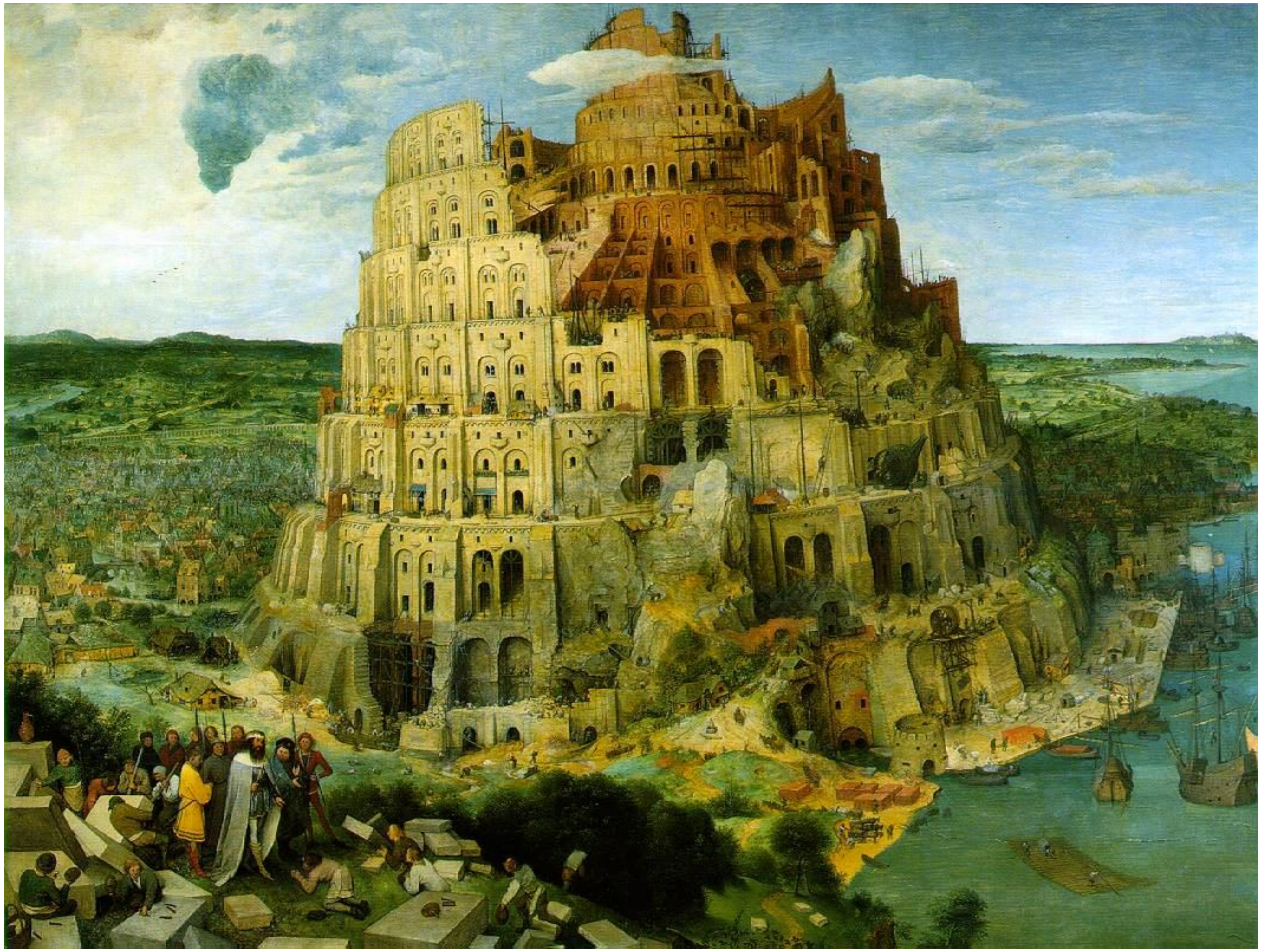
Information-rich programming in F#

Tomas Petricek, University of Cambridge
Don Syme and the **F# team**, Microsoft

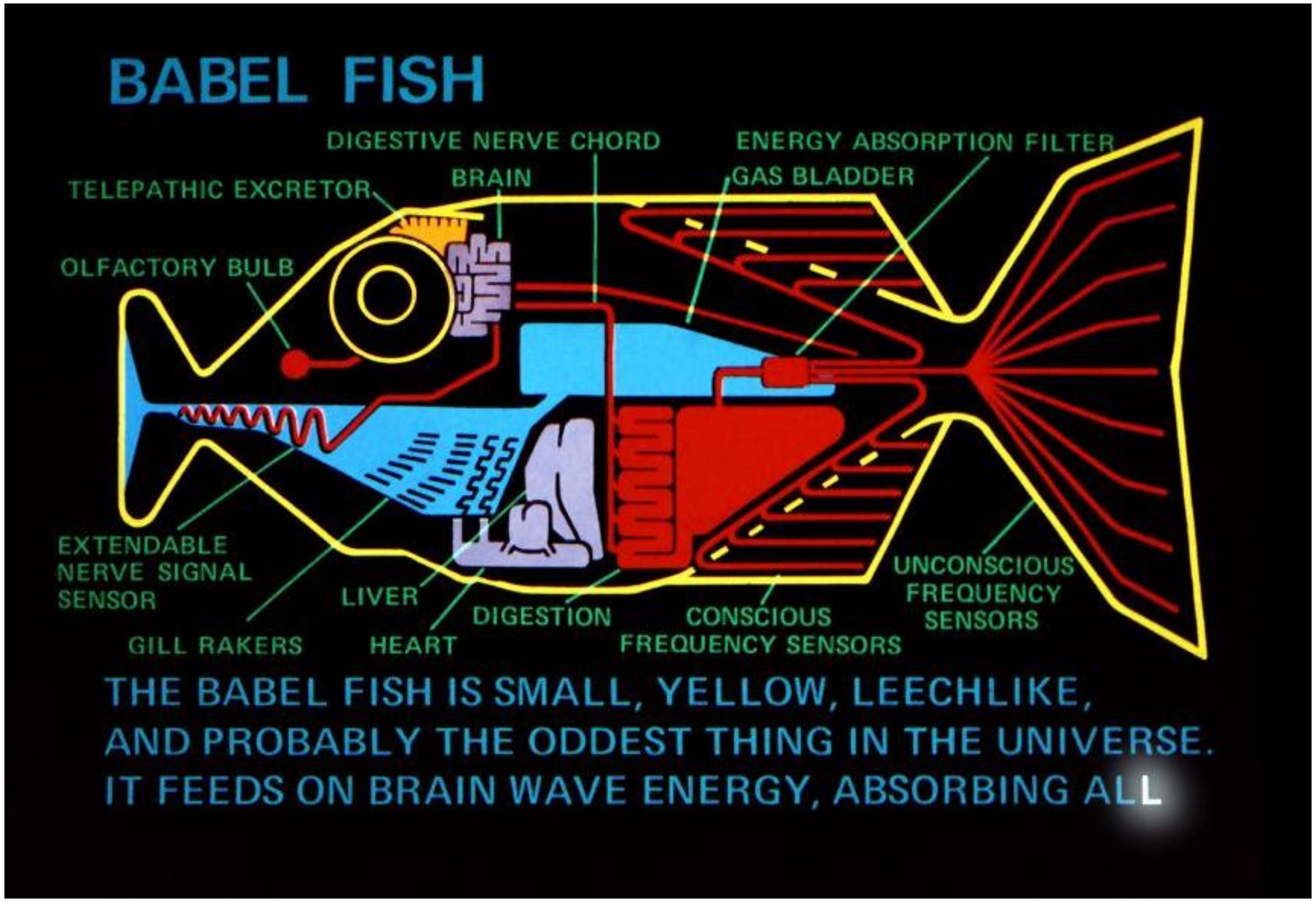
The confusion of languages



The confusion of languages



What is a type provider?


BABEL FISH
DIGESTIVE NERVE CHORD
ENERGY ABSORPTION FILTER
GAS BLADDER
BRAIN
TELEPATHIC EXCRETOR
OLFACTORY BULB
EXTENDABLE NERVE SIGNAL SENSOR
LIVER
DIGESTION
CONSCIOUS FREQUENCY SENSORS
UNCONSCIOUS FREQUENCY SENSORS
GILL RAKERS
HEART
THE BABEL FISH IS SMALL, YELLOW, LEECHLIKE,
AND PROBABLY THE ODDEST THING IN THE UNIVERSE.
IT FEEDS ON BRAIN WAVE ENERGY, ABSORBING ALL

DEMO: Accessing World Bank

Comparing **university enrollment rate** in **Czech Republic** and **OECD** countries

Problems with data access

Data is not types with members

Use dynamic languages?

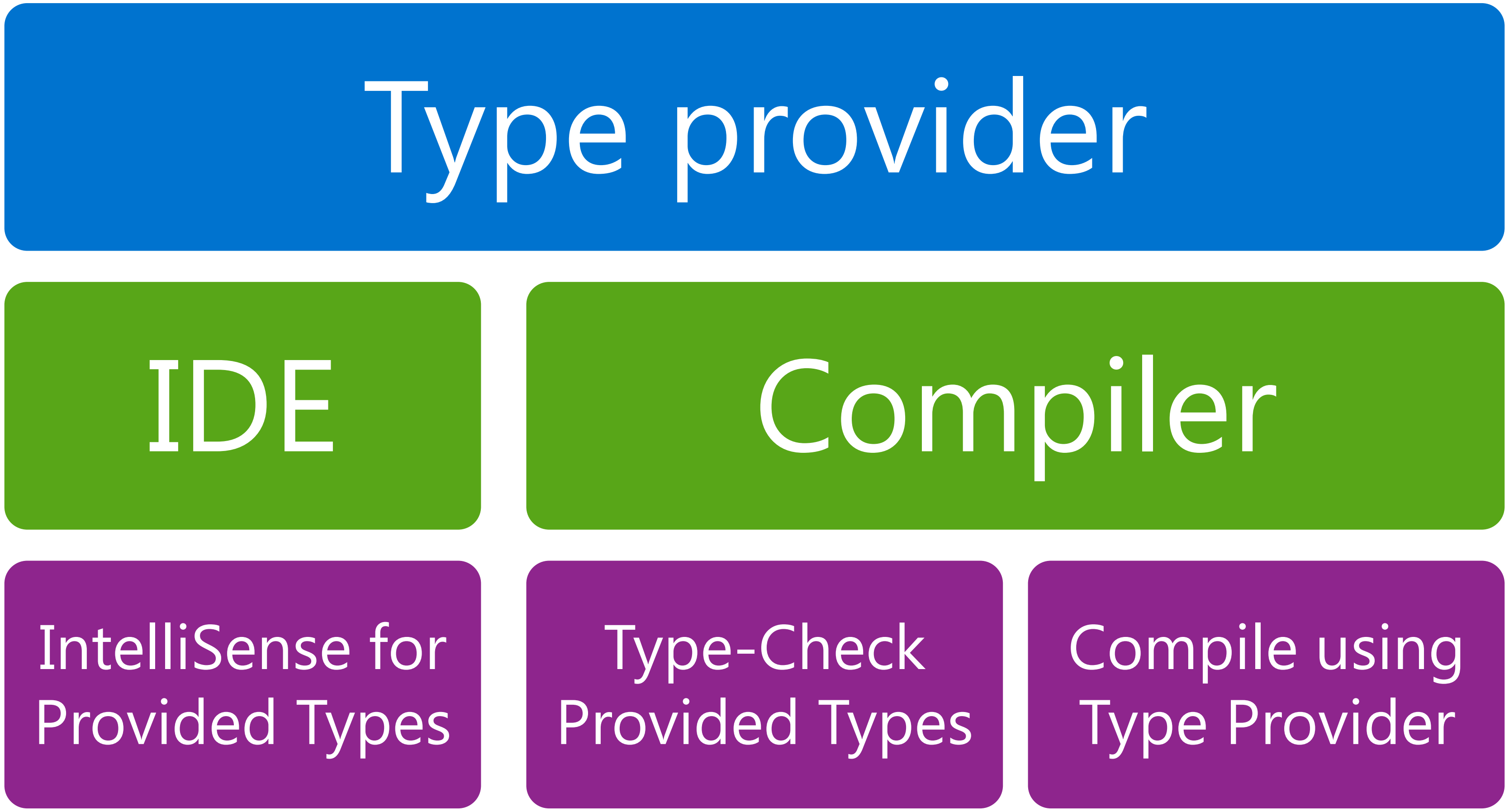
Need to know **names of properties**

Use code generation and static languages?

Enormous scale of data sources

Types need to be generated "on demand"

Components of a type provider


Type provider
IDE
Compiler
IntelliSense for Provided Types
Type-Check Provided Types
Compile using Type Provider

Research problems

Mapping data sources to types

    What is a type? What is a value?

Types are provided **on demand**

    Cannot generate all indicator types at once!

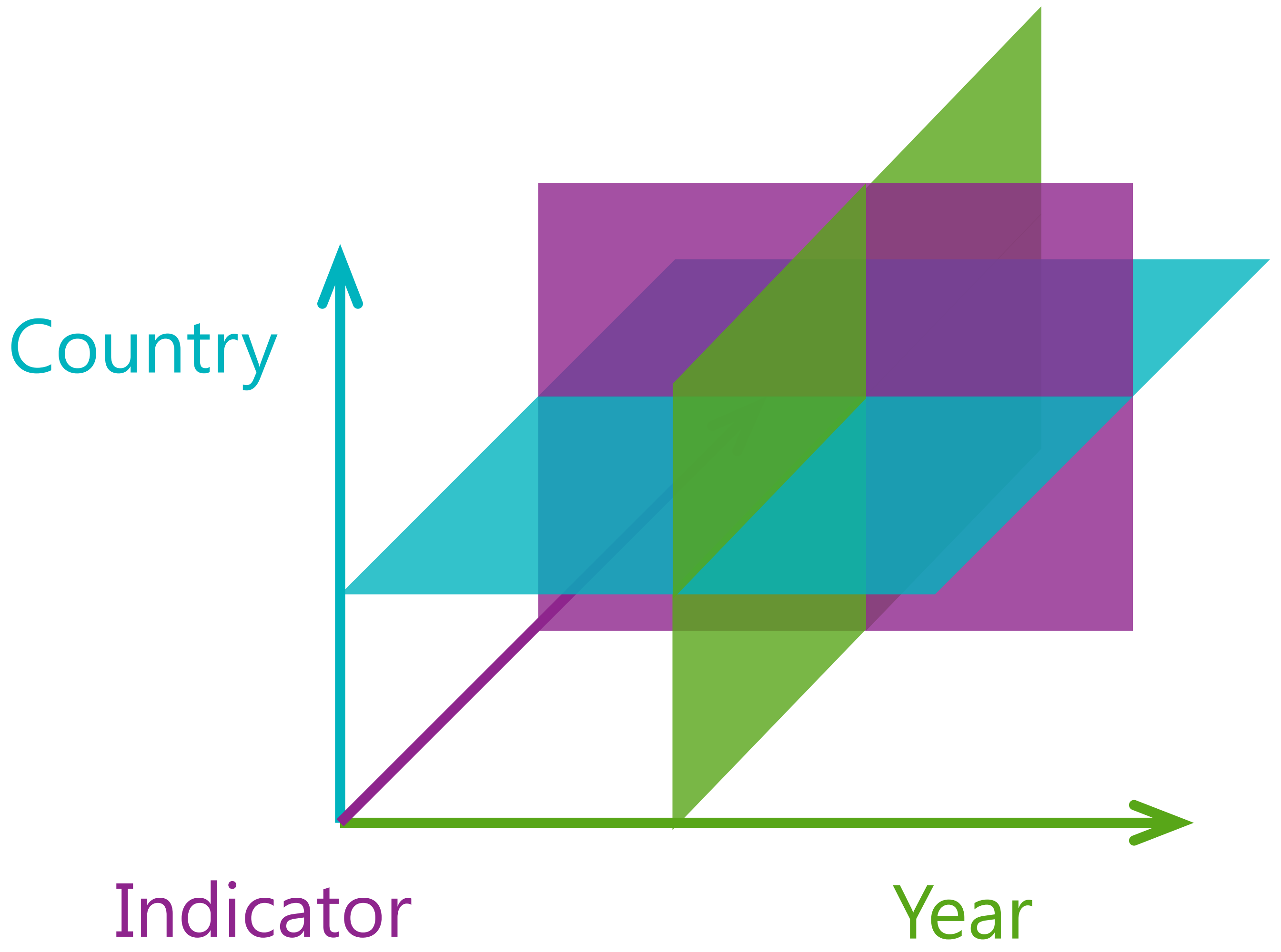
Representing **data source properties** as types

    Physical units, provenance, temporal properties

Adapting to **schema change**

    Type soundness is relative w.r.t. data source changes

Mapping data source to types



Research problems

Mapping data sources to types

- What is a type? What is a value?

Types are provided **on demand**

- Cannot generate all indicator types at once!

Representing **data source properties** as types

- Physical units, provenance, temporal properties

Adapting to **schema change**

- Type soundness is relative w.r.t. data source changes

Gamma, The Forgotten

Standard **typing judgments**

$$\Gamma \vdash e : \tau$$

Gamma on steroids

$$\Gamma = \ldots,\ \mathrm{WB.DataContext}$$

$$\mathrm{WB.DataContext} = \mathrm{Countries} : \mathrm{delay}(\ldots)$$

Country list

Reducing **delayed context**

$$\Gamma \vdash e : \tau \Rightarrow \Gamma'$$

Research problems

Mapping data sources to types

- What is a type? What is a value?

Types are provided **on demand**

- Cannot generate all indicator types at once!

Representing **data source properties** as types

- Physical units, provenance, temporal properties

Adapting to **schema change**

- Type soundness is relative w.r.t. data source changes

DEMO: XML Type Provider

Working with **XML data** and adapting to **schema change**

Related Work

Compile-time **meta-programming**

Types generated eagerly, not on demand

Dependently typed languages

Type-level computation in the IO monad??

Multi-stage computations

Focus on performance vs. data access

For more information

Upcoming technical report

Don Syme, et al. Strongly-Typed Language Support for an Information-Rich World

Workshop on related topics

Data Driven Functional Programming Workshop, **Co-located with POPL 2013**

Summary

Mismatch between **data** and **types**

- Type providers bridge the gap
- Development-time, compile-time & run-time

Interesting future questions

- Relative type safety and schema change
- Capturing meta-data with types

Research problems

Mapping data sources to types

- What is a type? What is a value?

Types are provided **on demand**

- Cannot generate all indicator types at once!

Representing **data source properties** as types

- Physical units, provenance, temporal properties

Adapting to **schema change**

- Type soundness is relative w.r.t. data source changes

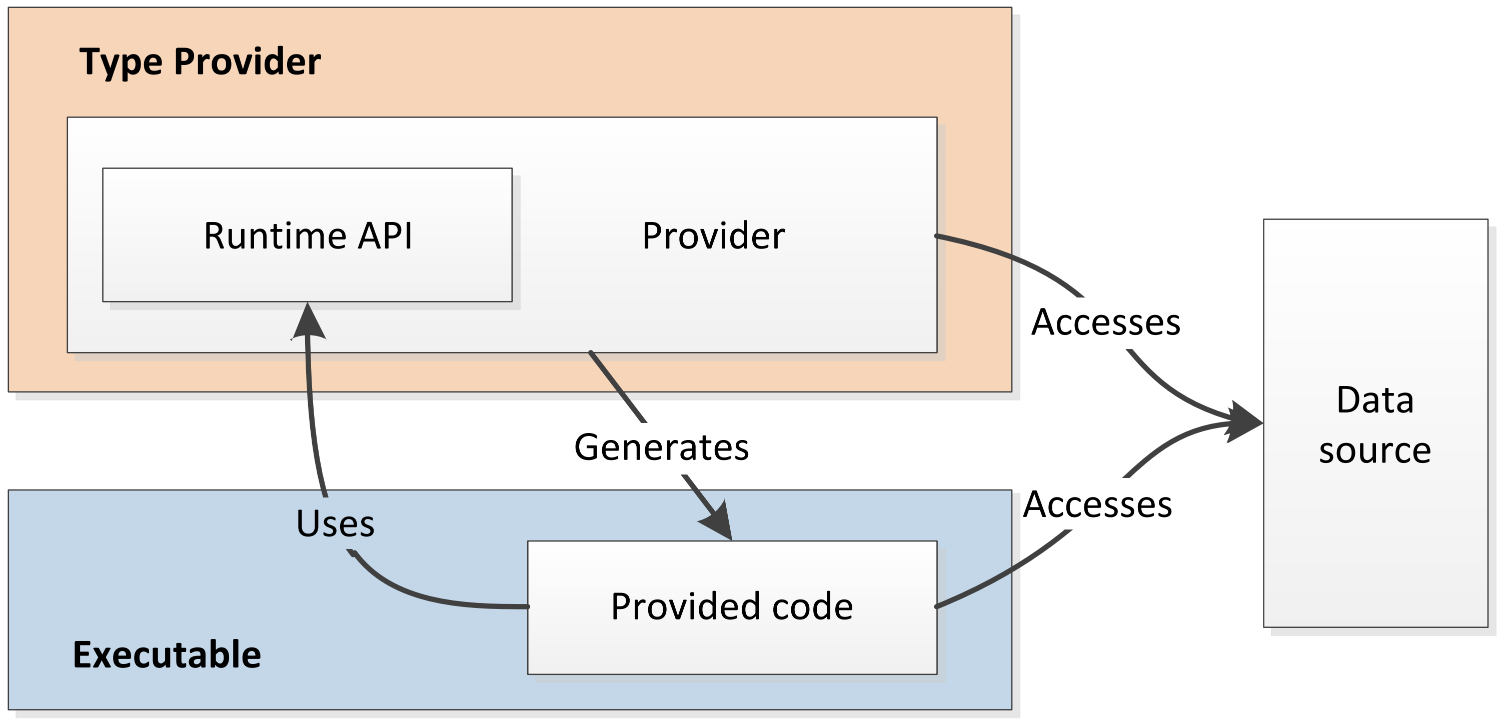
DEMO: FreeBase Type Provider

Working with **chemistry data**
and **units of measure**

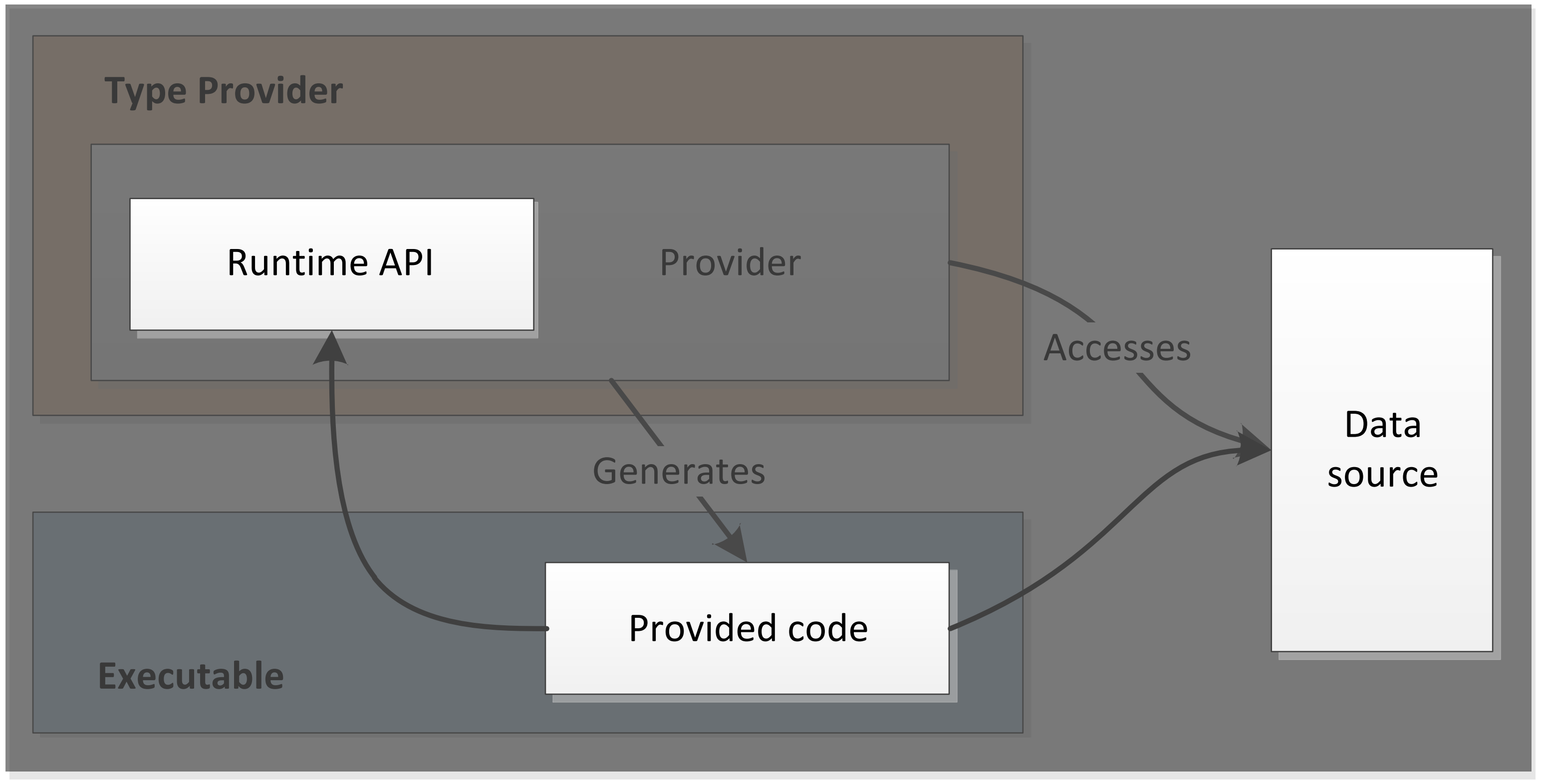
Structure of a Simple Provider

```
[<TypeProvider>]
type SampleTypeProvider(config: TypeProviderConfig) =
  inherit TypeProviderForNamespaces()

  // Define new type Samples.GeneratedType
  let thisAssembly = Assembly.GetExecutingAssembly()
  let providedType = ProvidedTypeDefinition( ... )
  do
    // Add property 'Hello' that just returns a string
    ProvidedProperty
      ( "Hello", typeof<string>, IsStatic = true,
        GetterCode = fun args -> <@@ Runtime.lookup "Hello" @@>)
    |> providedType.AddMember

    // Register the type with the compiler
    this.AddNamespace(namespaceName, [ providedType ])
```

Compile-Time vs. Run-time



Compile-Time vs. Run-time



Queries in F#

Can be turned to **quotations**

```
query { for movie in netflix.Titles do
        where (movie.Name.Contains(search))
        select movie }
```

Extensible query language

```
query { for index in Numbers do
        reverse
        takeWhile index > 10 }
```