Cω

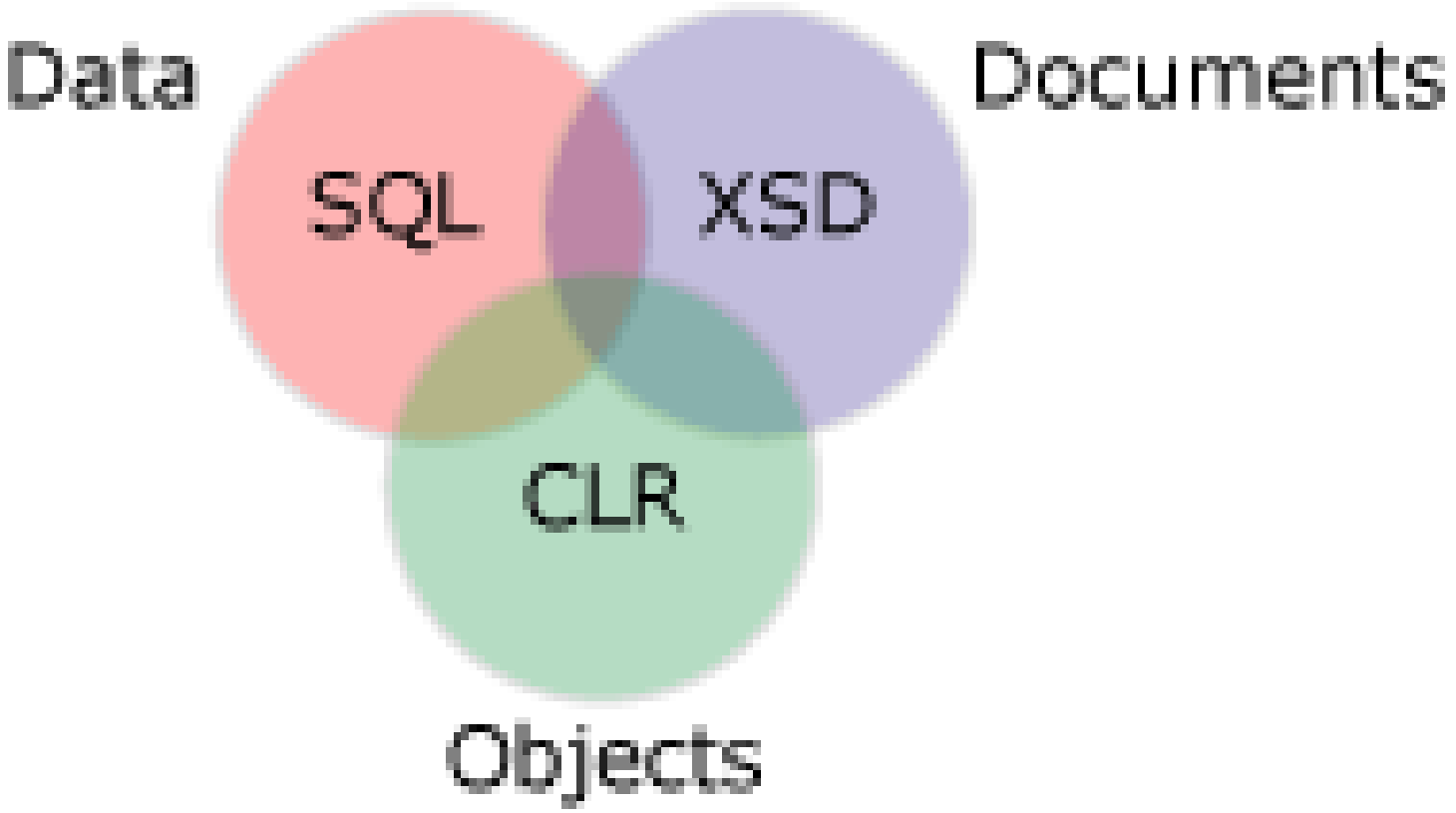
Perchè Cω?

- Target: applicazioni distribuite web-based
- Architettura “comune”:
  - Database (dati relazionali)
  - Main application – C#/Java
  - User interface (XML/HTML – dati generici semi strutturati)
- Concurrency scenario: eventi asincroni e scambio messaggi
  - Web services, …

Perchè Cω?

- Garanzie compile-time più forti.
- Il compilatore ha più informazioni ed ha la “libertà” per scegliere le strategie differenti di esecuzione (per esempio effettuando le ottimizzazioni di domanda).
- Sintassi più naturale.
- Estensione di un linguaggio già esistente
- Nel caso della “concorrenza” viene adottato il modello teorico del Join-calculus (linguaggio funzionale).

Accesso ai dati in Cω



Minimal extensions to C#

- New types:
  - Streams (`T*`)
  - Choice (`choice{int;string;}`)
  - Tuples (`struct{int i;string s;}`)
  - Content classes
  - Nullables (`T?`)
  - `async` (subtype of `void`)
- Chords
- Generalized member access (cf. XPath)
- SQL-style `select` queries

Streams (T*)

```
virtual string* Foo()
{
    yield return "Hello world!";
}
```

NOTA: durante l'esecuzione di questo codice il compilatore non esegue un calcolo, ma restituisce una “chiusura” (i flussi sono esattamente delle “lazy lists” nello stile di Haskell.

I flussi sono sempre “appiattiti”, ovvero non è possibile avere flussi annidati.

Streams: Example

```
public static int* FromTo(int s, int e) {
    for (i = s; i <= e; i++) yield return i;
   }

int* OneToTen = FromTo(1,10);

foreach(int j in OneToTen) {
    Console.WriteLine(j);
};
```

Choice (`choice{int;string;}`)

choice{int;Button;} x = 3;

choice{int;Button;} y = new Button();

Questo tipo permette l'esecuzione di una scelta relativa all'attribuzione del tipo della variabile.

Tuples (`struct{int i;string s;}`)

struct{int i; Button;}

questa struttura contiene un membro di tipo int chiamato “i” ed uno di tipo Button senza nome (analogo delle tuple in ML o Haskell)

new(i=42,new Button()) - utilizzo della tupla

x[1].BackColor

per accedere ad un membro della tupla, ad esempio quello anonimo.

Content classes

Al fine di ottenere una maggiore integrazione con XML sono state introdotte le “Content classes”

```
class friend{
        struct{
                string name;
                int age;
        };
        void incAge(){...}
}
```

Pippo.age – ritornerà l'età di Pippo

Generalized member access

- Sequences

```
string* ss;
ss.ToUpper();
```

string*

- Anonymous struct types

```
struct{int i; string s; string s;} x;
x.i;
x.s;
```

int

string*

- Choice types

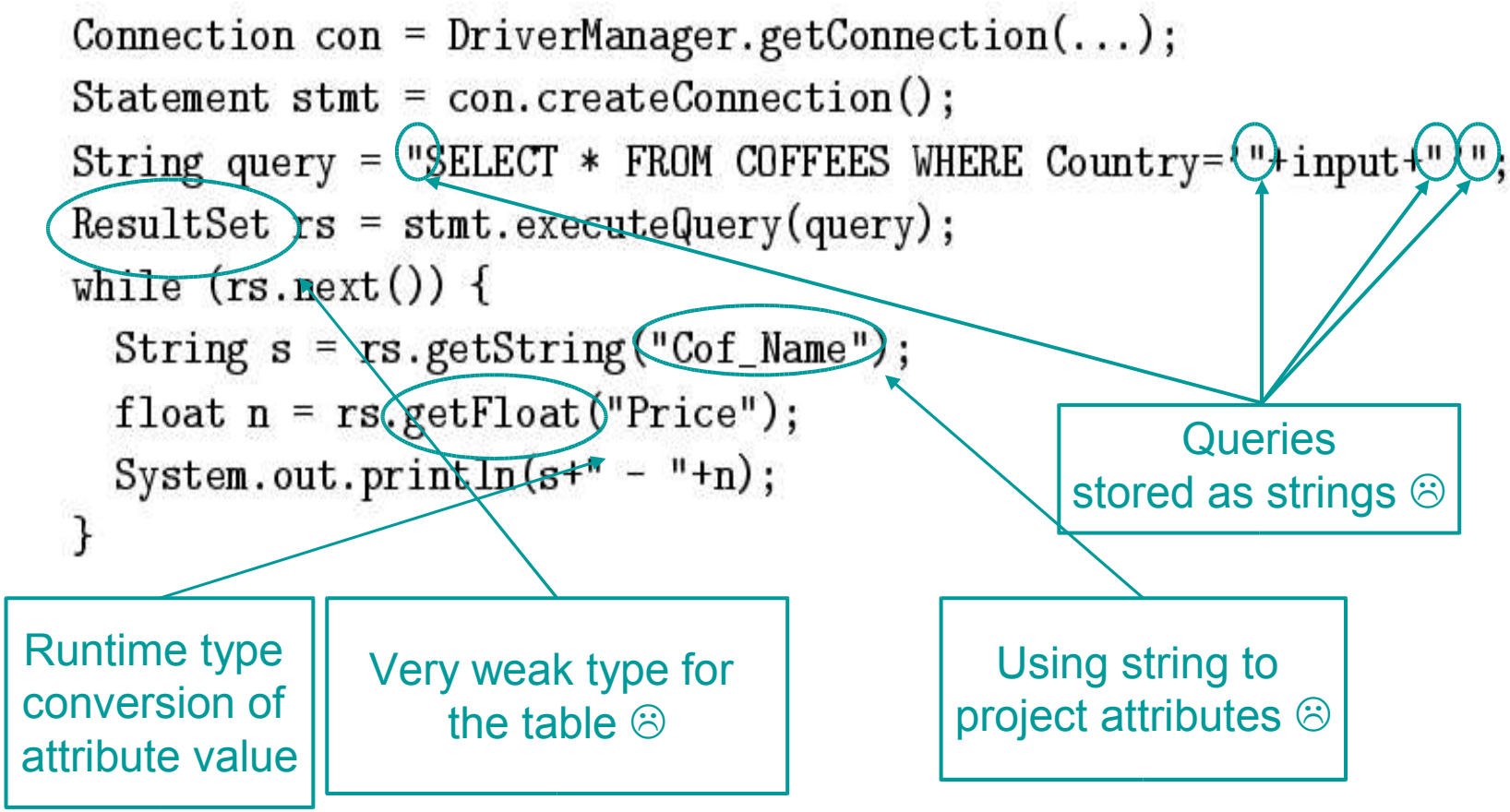
```
choice{string;Person;int;} y;
y.Length;
```

Optional type

choice{int;Float;}?

Data access in Cω

Current API data access



A sneak preview…

```
<!ELEMENT book
   (title,
    (author*|editor*),
    publisher,
    price)
>
```

```
public class book {
   struct{
     string title;
     choice{
      struct{editor editor;}*;
      struct{author author;}*;
     }
     string publisher;
     decimal price;
   }
}
```

A sneak preview…

```
book b = <book>
           <title>Cw in a nutshell</title>
           <author><first>Nick</first>
                   <last>Benton</last>
           </author>
           <author><first>Gavin</first>
                   <last>Bierman</last>
           </author>
           <publisher>OReally</publisher>
           <price>40.99</price>
         </book>;


b.author.first.{ Console.WriteLine(it); };
```

XML in your code!!

Path expressions (like XPath) in your code!!

A sneak preview…


```
alts1 = select Title,Artist
        from CDs
        where Style == CDStyle.Alt;


alts1.{ConsoleWriteLine("Title={0},Artist={1}",
                        it.Title,it.Artist);};
```

Path expressions

- GMA allows us to write OQL-like path expressions:

```
static int* FromTo(int s, int e) {
    for (i = s; i <= e; i++) yield return i;
   }
FromTo(0,100).{return it.ToString("x");}
             .ToUpper()
             .{Console.WriteLine(it);};
```

XQuery use case 3

For each book in the bibliography, list the title and authors, grouped inside a “result” element

XQUERY

```
for $b in $bs/book
return
 <result>
   {$b/title}
   {$b/author}
 <result>
```

Cω

```
foreach (b in bs.book)
{
  yield return <result>
                 {b.title}
                 {b.author}
               </result>;
}
```

A simple buffer

(for use by producer/consumer threads)

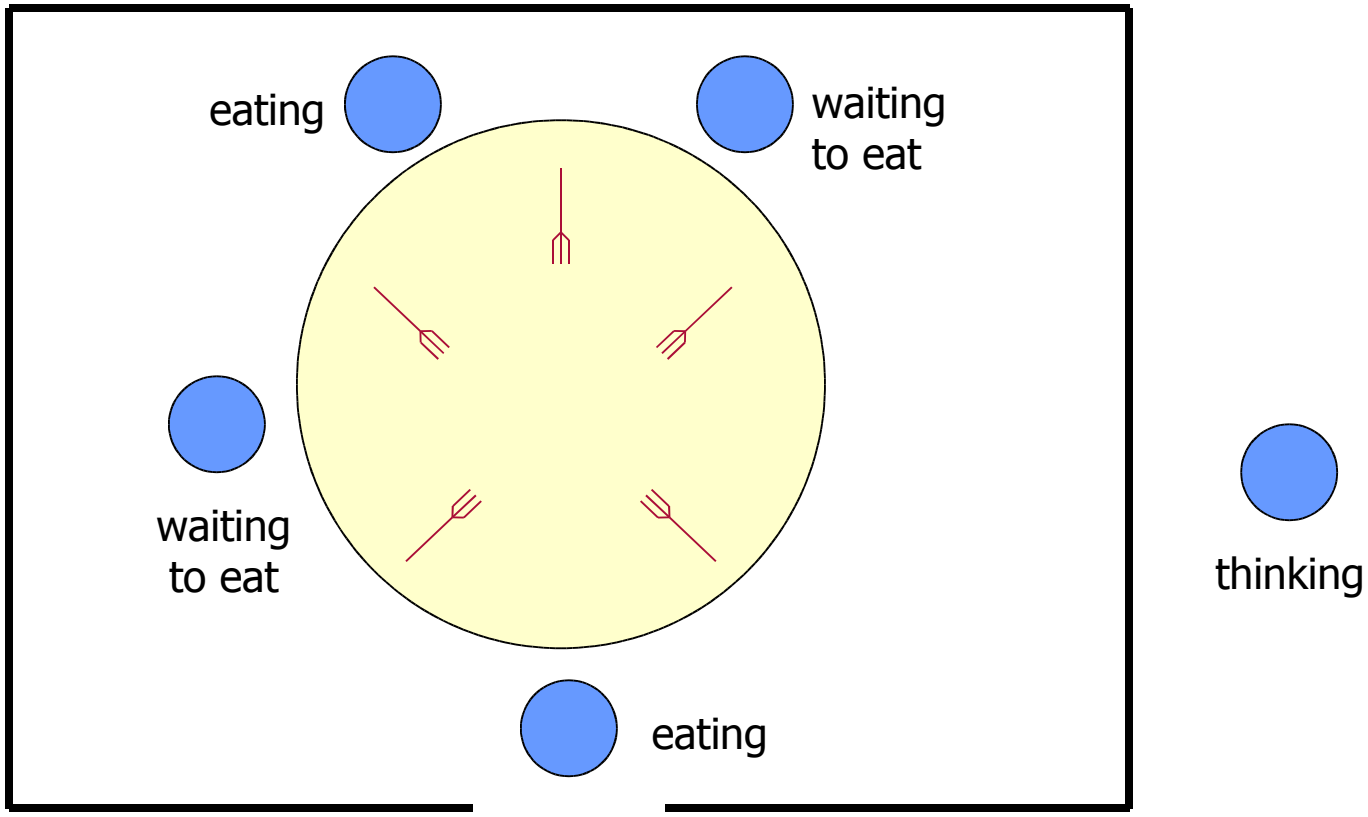
```
class Buffer {
  async put(string s);
  string get() & put(string s) {
      return s;
  }
}
```

A simple buffer

```
class Buffer {
  async put(string s);
  string get() & put(string s) {
      return s;
  }
}
```

- Calls to `put()` return immediately, but are internally queued if there's no waiting `get()`
- Calls to `get()` block until/unless there's a matching `put()`
- When there's a match the body runs, returning the argument of the `put()` to the caller of `get()`.
- Exactly which pairs of calls are matched up is unspecified.

Dining Philosophers



Code extract

```
class Room {
    async hasspaces(int n);
    async isfull();

    public Room (int size) { hasspaces(size); }

    public void enter() & hasspaces(int n) {
      if (n > 1)  hasspaces(n-1);
      else        isfull();
    }

    public void leave()
      & hasspaces(int n) {
          hasspaces(n+1);
      }
      & isfull() {
         hasspaces(1);
      }
}
```

Relationship with C#

- Note: Cω is **not** C# n.0
  - We're a **research** project
  - We're not sponsored by the C# or VS team

"The C# team is excited about Cω and other C#-based research projects that MSR-Cambridge is working on. We have no current plans to extend the C# language in this direction, but will continue to observe the progress of Cω and other MSR-Cambridge projects."

Scott Wiltamuth, C# Product Unit Manager

Compiler release

- Available (v.1.0.2) from http://research.microsoft.com/Comega

Domande??