Regular Object Types

Vladimir Gapeyev

Benjamin C. Pierce

University of Pennsylvania
Philadelphia, USA

XML

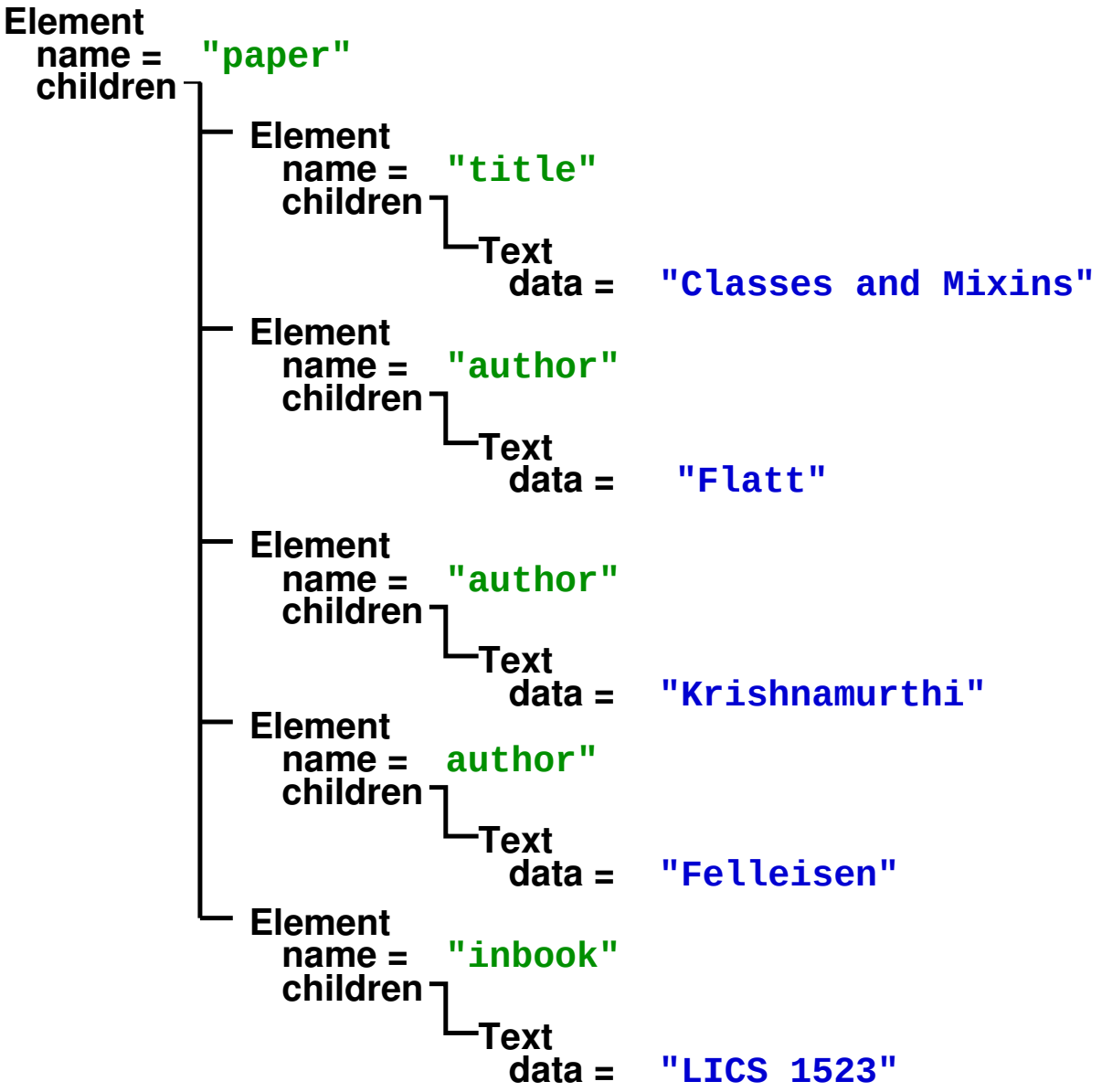
```
<readings>
  <paper>
    <title>Import is
           not Inheritance</title>
    <author>Szyperski</author>
    <inproc>ECOOP'92</inproc>
  </paper>
  <paper>
    <title>Classes and Mixins</title>
    <author>Flatt</author>
    <author>Krishnamurthi</author>
    <author>Felleisen</author>
    <inbook>LNCS 1523</inbook>
  </paper>
</readings>
```

XML and Regular Expression (RE) types

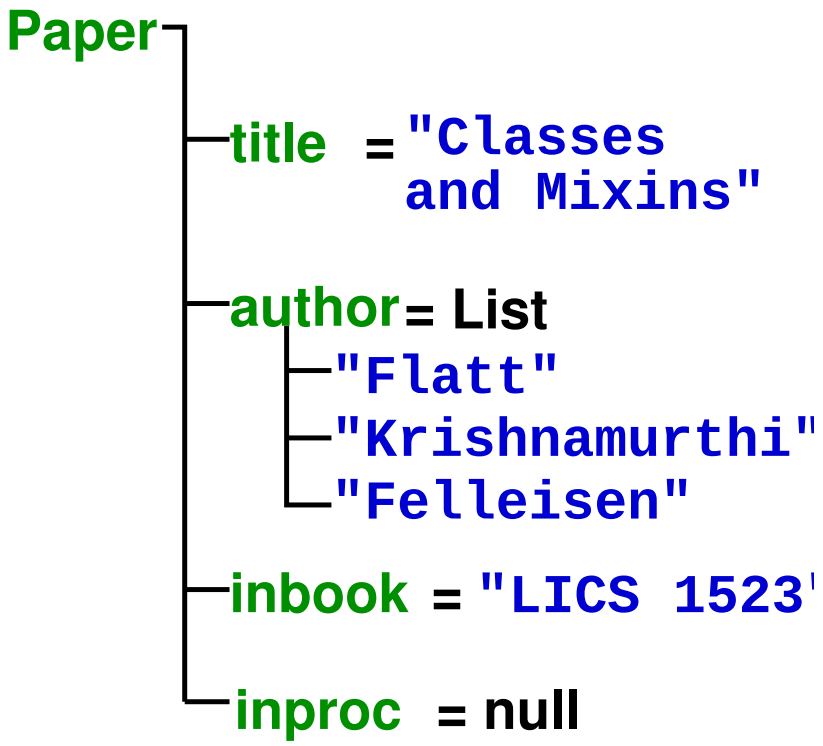
```
<readings>
  <paper>
    <title>Import is
           not Inheritance</title>
    <author>Szyperski</author>
    <inproc>ECOOP'92</inproc>
  </paper>
  <paper>
    <title>Classes and Mixins</title>
    <author>Flatt</author>
    <author>Krishnamurthi</author>
    <author>Felleisen</author>
    <inbook>LNCS 1523</inbook>
  </paper>
</readings>
```

```
Title  = <title>[pcdata]
Author = <author>[pcdata]

Source  =   <inproc>[pcdata]
          | <inbook>[pcdata]

Paper = <paper>[Title,
                Author+, Source]

Readings = <readings>[Paper*]
```

DOM: a generic AST for XML



XML data binding: schema-specific ASTs




```
public interface Paper extends
  Element, PaperType {}

public interface PaperType {
    String getTitle();
    void setTitle(String value);

    List getAuthor();

    String getInbook();
    void setInbook(String value);

    String getInproc();
    void setInproc(String value);
}
```

XDuce: RE types can be PL types

```
fun getConferenceTitles (ps as Paper*) : Title* =
  match ps with
  | <paper>[t as Title, Author*, <inproc>[pcdata]],
    rest as Paper* ->
      t, getConferenceTitles(rest)
  | Paper, rest as Paper* ->
      getConferenceTitles(rest)
  | () -> ()
```

XDuce: [Hosoya,Vouillon,Pierce – ICFP'00], [Hosoya,Pierce – POPL'01]

XTATIC **project:** XDUCE **to the masses!**

Vision:

Native support
for statically typed XML programming
in a mainstream OO language.

XTATIC **project:** XDUCE **to the masses!**

Vision:

Native support
for statically typed XML programming
in a mainstream OO language.

Technical challenge:

Combine an OO type system and an RE type system.

XTATIC **project:** XDUCE **to the masses!**

Vision:

Native support
for statically typed XML programming
in a mainstream OO language.

Technical challenge:

Combine an OO type system and an RE type system.

Practical focus:

XTATIC = C# + XDUCE

XTATIC **project:** XDUCE **to the masses!**

Vision:

Native support
for statically typed XML programming
in a mainstream OO language.

Technical challenge:

Combine an OO type system and an RE type system.

Practical focus:

$$\text{XTATIC} = \text{C}^{\#} + \text{XDUCE}$$

Theory:

$$\text{FX} = \text{FJ} + \text{XDUCE}$$

[FJ : Igarashi,Pierce,Wadler – OOPSLA'99]

XDuce's type system

Values: $\overline{t} ::= t_1, \ldots, t_n \qquad t ::= \langle (l) \rangle [\overline{t}]$

XDuce's type system

Values: $\overline{t} ::= t_1, \ldots, t_n$ $\quad t ::= \langle l \rangle [\overline{t}]$

$\langle L \rangle [T]$

T_1, T_2

$T_1 | T_2$

$[]$

$T*$

X

XDuce's type system

Values: $\overline{t} ::= t_1, \ldots, t_n \qquad t ::= \langle l \rangle [\overline{t}]$

Denotations:

$$
\begin{aligned}
[\![\langle L \rangle [T]]\!] &= \{\langle l \rangle [\overline{t}] \mid l \in [\![L]\!], \quad \overline{t} \in [\![T]\!]\} \\
[\![T_1, T_2]\!] &= \{\overline{t}_1, \overline{t}_2 \mid \overline{t}_k \in [\![T_k]\!]\} \\
[\![T_1 | T_2]\!] &= [\![T_1]\!] \cup [\![T_2]\!] \\
[\![[]]\!] &= \{[]\} \\
[\![T*]\!] &= \{\overline{t}_1, \ldots, \overline{t}_n \mid \overline{t}_k \in [\![T]\!]\} \\
[\![X]\!] &= [\![def(X)]\!]
\end{aligned}
$$

XDuce's type system

Values: $\overline{t} ::= t_1, \ldots, t_n \qquad t ::= \langle l \rangle [\overline{t}]$

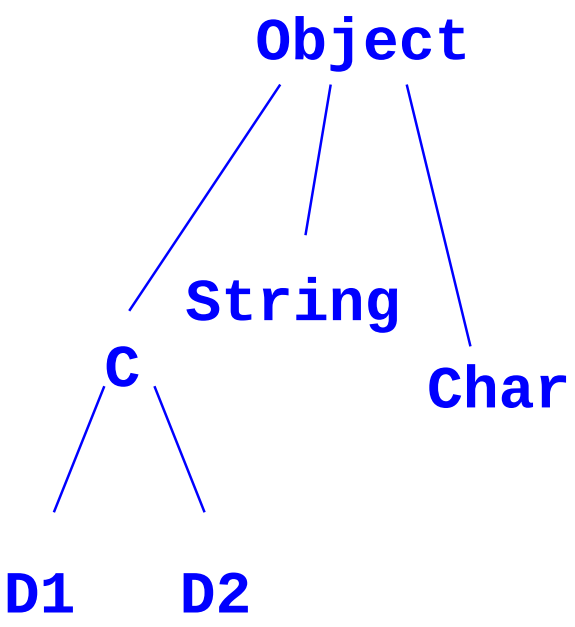
Denotations:

$$
\begin{aligned}
[\![\langle L \rangle [T]]\!] &= \{\langle l \rangle [\overline{t}] \mid l \in [\![L]\!], \quad \overline{t} \in [\![T]\!]\} \\
[\![T_1, T_2]\!] &= \{\overline{t}_1, \overline{t}_2 \mid \overline{t}_k \in [\![T_k]\!]\} \\
[\![T_1 | T_2]\!] &= [\![T_1]\!] \cup [\![T_2]\!] \\
[\![[]]\!] &= \{[]\} \\
[\![T*]\!] &= \{\overline{t}_1, \ldots, \overline{t}_n \mid \overline{t}_k \in [\![T]\!]\} \\
[\![X]\!] &= [\![def(X)]\!]
\end{aligned}
$$

Semantic subtyping:

$$T_1 <: T_2 \qquad \Longleftrightarrow \qquad [\![T_1]\!] \subseteq [\![T_2]\!]$$

OO subtyping

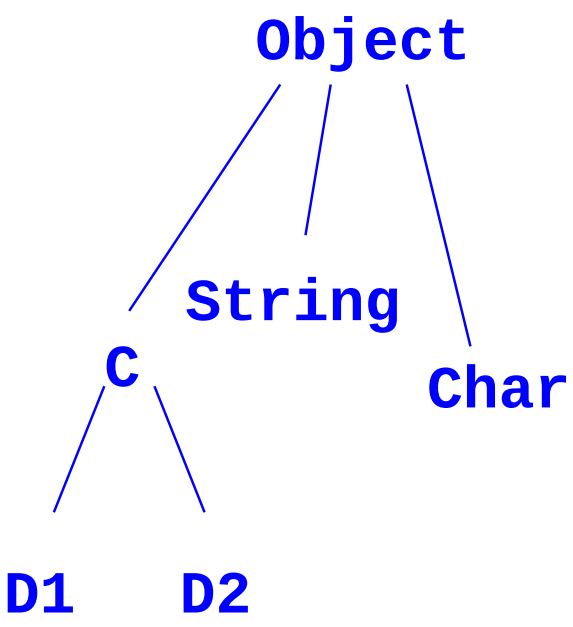


Subclassing: $\mathtt{D} \sqsubset: \mathtt{C}$

— reflexive and transitive closure of

$$\texttt{class } \mathtt{C}_1 \texttt{ extends } \mathtt{C}_2 \texttt{ \{...\}}$$

OO subtyping — semantically



Subclassing: $\mathtt{D} \sqsubset: \mathtt{C}$

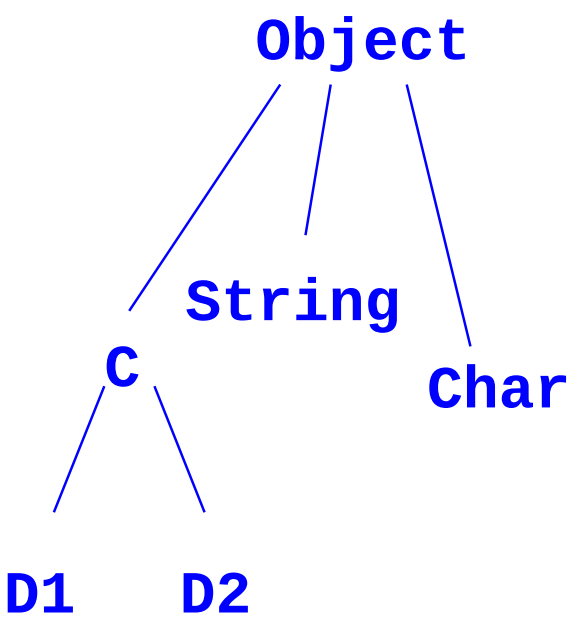
— reflexive and transitive closure of

$$\texttt{class } \mathtt{C}_1 \texttt{ extends } \mathtt{C}_2 \texttt{ \{...\}}$$

Define:

$$[\![\mathtt{C}]\!] \;=\; \{\texttt{new } \mathtt{D}(\overline{\mathtt{o}}) \mid \mathtt{D} \sqsubset: \mathtt{C},\ \mathit{fields}(\mathtt{D}) = \overline{\mathtt{F}}\ \overline{\mathtt{f}},\ \mathtt{o}_i \in [\![\mathtt{F}_i]\!]\}$$

OO subtyping — semantically



Subclassing: $\mathtt{D} \sqsubset: \mathtt{C}$

— reflexive and transitive closure of

$$\texttt{class } \mathtt{C}_1 \texttt{ extends } \mathtt{C}_2 \texttt{ \{...\}}$$

Define:

$$[\![\mathtt{C}]\!] \;=\; \{\texttt{new } \mathtt{D}(\overline{\mathtt{o}}) \mid \mathtt{D} \sqsubset: \mathtt{C},\ \mathit{fields}(\mathtt{D}) = \overline{\mathtt{F}}\ \overline{\mathtt{f}},\ \mathtt{o}_i \in [\![\mathtt{F}_i]\!]\}$$

Then:

$$\mathtt{C}_1 \sqsubset: \mathtt{C}_2 \iff [\![\mathtt{C}_1]\!] \subseteq [\![\mathtt{C}_2]\!]$$

XTATIC = C# + XDUCE

Values:

```
t  ::=
   <(a)>[t]
```

Types:

```
T  ::=
   <(A)>[T]
   T1,T2
   T1|T2
   []
   T*
   X
```

XTATIC = C# + XDUCE

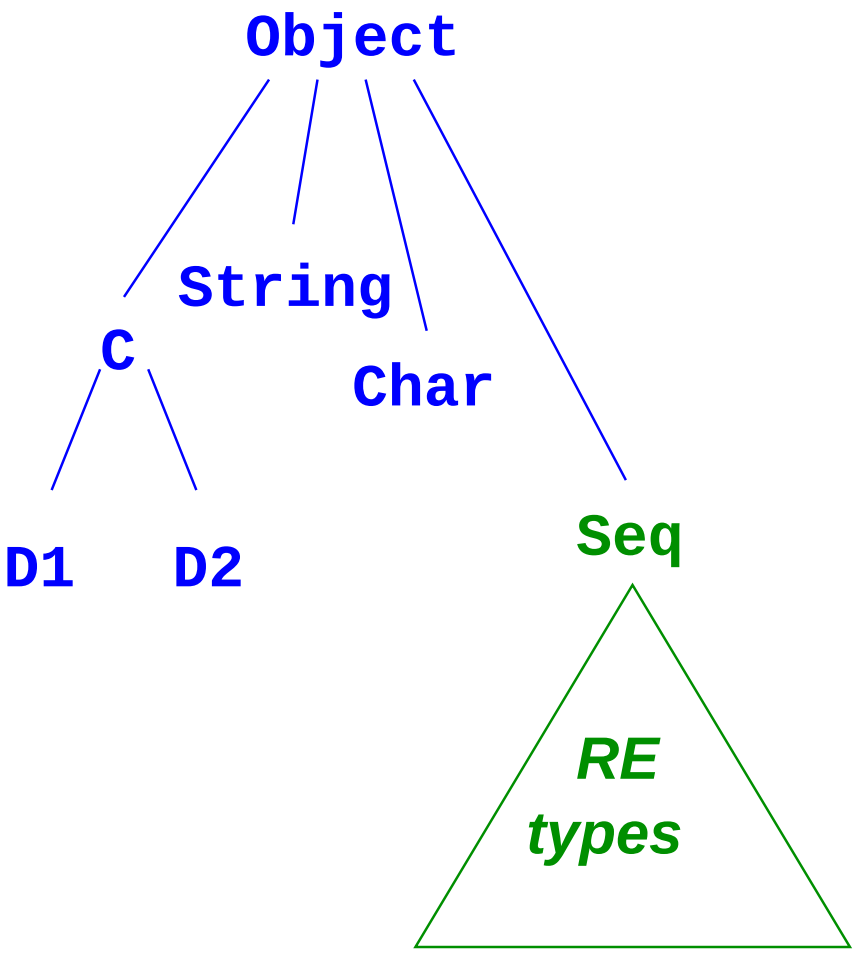
Values:

```
a  ::=
   new C(ā)
   [t̄]
t  ::=
   <(a)>[t̄]
```

Types:

```
A  ::=
   C
   [T]
T  ::=
   <(A)>[T]
   T1,T2
   T1|T2
   []
   T*
   X
```

XTATIC = C# + XDUCE



Values:

a	::=	
	new C($\overline{a}$)	
	[$\overline{t}$]	
t	::=	
	‹(a)› [$\overline{t}$]	

Types:

A	::=	
	C	
	[T]	
T	::=	
	‹(A)› [T]	
	T_1 , T_2	
	T_1 | T_2	
	[]	
	T*	
	X	

PIZZA: [Odersky,Wadler – POPL'97]

Type denotations and subtyping

$$[\![[\langle (A) \rangle [T]]]\!] \quad = \quad \{ \langle (a) \rangle [\overline{t}] \mid a \in [\![A]\!], \quad [\overline{t}] \in [\![[T]]\!] \}$$

$$\ldots$$

Type denotations and subtyping

$$[\![[\langle (\mathtt{A}) \rangle [\mathtt{T}]]]\!] = \{ \langle (\mathtt{a}) \rangle [\overline{\mathtt{t}}] \mid \mathtt{a} \in [\![\mathtt{A}]\!], \quad [\overline{\mathtt{t}}] \in [\![[\mathtt{T}]]\!] \}$$

$$\ldots$$

$$[\![\mathtt{Seq}]\!] = \{ [\overline{\mathtt{t}}] \mid \overline{\mathtt{t}} \in [\![[\mathtt{T}]]\!] \text{ for some } \mathtt{T} \}$$

Type denotations and subtyping

$$[\![\texttt{[<(A)>[T]]}]\!] = \{ \texttt{<(a)>[}\overline{\texttt{t}}\texttt{]} \mid \texttt{a} \in [\![\texttt{A}]\!], \quad \texttt{[}\overline{\texttt{t}}\texttt{]} \in [\![\texttt{[T]}]\!] \}$$

$$\cdots$$

$$[\![\texttt{Seq}]\!] = \{ \texttt{[}\overline{\texttt{t}}\texttt{]} \mid \overline{\texttt{t}} \in [\![\texttt{[T]}]\!] \text{ for some } \texttt{T} \}$$

$$[\![\texttt{C}]\!] = \{ \texttt{new D(}\overline{\texttt{a}}\texttt{)} \mid \texttt{D} \sqsubset\texttt{: C}, \ \mathit{fields}(\texttt{D}) = \overline{\texttt{F}}\ \overline{\texttt{f}}, \ \texttt{a}_\texttt{i} \in [\![\texttt{F}_\texttt{i}]\!] \}$$

Type denotations and subtyping

$$
\begin{array}{lcl}
[\![\texttt{[<(A)>[T]]}]\!] & = & \{ \texttt{<(a)>[}\overline{\texttt{t}}\texttt{]} \mid \texttt{a} \in [\![\texttt{A}]\!], \quad \texttt{[}\overline{\texttt{t}}\texttt{]} \in [\![\texttt{[T]}]\!] \} \\
 & \cdots & \\
[\![\texttt{Seq}]\!] & = & \{ \texttt{[}\overline{\texttt{t}}\texttt{]} \mid \overline{\texttt{t}} \in [\![\texttt{[T]}]\!] \text{ for some } \texttt{T} \} \\
[\![\texttt{C}]\!] & = & \{ \texttt{new D(}\overline{\texttt{a}}\texttt{)} \mid \texttt{D} \sqsubset \texttt{C},\ \mathit{fields}(\texttt{D}) = \overline{\texttt{F}}\ \overline{\texttt{f}},\ \texttt{a}_i \in [\![\texttt{F}_i]\!] \} \\
 & & (\cup [\![\texttt{Seq}]\!], \text{ if } \texttt{C} = \texttt{Object})
\end{array}
$$

Type denotations and subtyping

$$[\![[\langle (\mathtt{A}) \rangle [\mathtt{T}]]]\!] = \{ \langle (\mathtt{a}) \rangle [\overline{\mathtt{t}}] \mid \mathtt{a} \in [\![\mathtt{A}]\!], \quad [\overline{\mathtt{t}}] \in [\![[\mathtt{T}]]\!] \}$$

$$\ldots$$

$$[\![\mathtt{Seq}]\!] = \{ [\overline{\mathtt{t}}] \mid \overline{\mathtt{t}} \in [\![[\mathtt{T}]]\!] \text{ for some } \mathtt{T} \}$$

$$[\![\mathtt{C}]\!] = \{ \mathtt{new}\ \mathtt{D}(\overline{\mathtt{a}}) \mid \mathtt{D} \sqsubset: \mathtt{C},\ \mathit{fields}(\mathtt{D}) = \overline{\mathtt{F}}\ \overline{\mathtt{f}},\ \mathtt{a}_i \in [\![\mathtt{F}_i]\!] \}$$

$$(\cup [\![\mathtt{Seq}]\!], \text{ if } \mathtt{C} = \mathtt{Object})$$

Define XTATIC subtyping $<:$ as

$$\mathtt{A}_1 <: \mathtt{A}_2 \iff [\![\mathtt{A}_1]\!] \subseteq [\![\mathtt{A}_2]\!]$$

Corollary:

$$\mathtt{C}_1 <: \mathtt{C}_2 \iff [\![\mathtt{C}_1]\!] \subseteq [\![\mathtt{C}_2]\!] \iff \mathtt{C}_1 \sqsubset: \mathtt{C}_2$$

Featherweight XTATIC expressions

e	::=	
	`new C(`$\overline{\texttt{e}}$`)`	object creation
	`<(e)>[`$\overline{\texttt{e}}$`]`	tree
	`[`$\overline{\texttt{e}}$`]`	sequence
	`e.f`	field access
	`e.m(`$\overline{\texttt{e}}$`)`	method call
	`match(e){case [`$\overline{\texttt{P}}$`]: `$\overline{\texttt{e}}$`}`	pattern match
	`x`	value variable

Standard properties (for FX)

Theorem [Preservation]: Suppose Σ conforms to Γ.

If $\Gamma \vdash \mathtt{e} \in \mathtt{A}$

and $\Sigma \vdash \mathtt{e} \Downarrow \mathtt{a}$

then $\mathtt{a} \in [\![\mathtt{A}]\!]$.

Theorem [Soundness]:

If $\bullet \vdash \mathtt{e} \in \mathtt{A}$

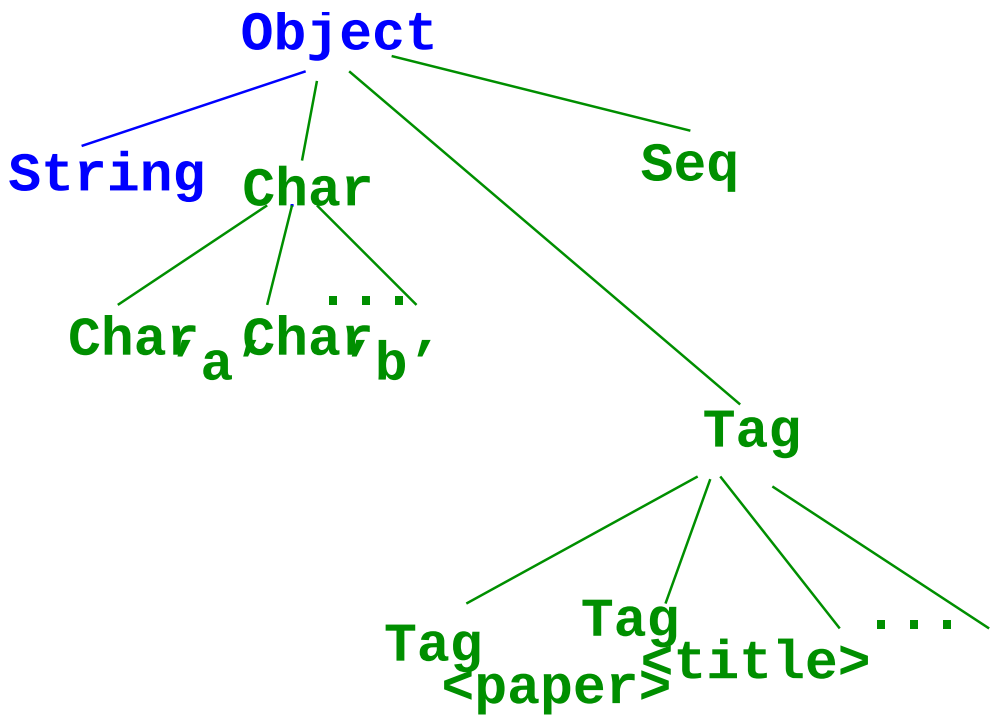
then evaluation of $\mathtt{e}$ does not go wrong.

XML representation: types

a	::=	t	::=
	new C($\overline{a}$)		<(a)>[$\overline{t}$]
	[$\overline{t}$]		
A	::=	T	::=
	C		<(A)>[T]
	[T]		T_1, T_2
			...

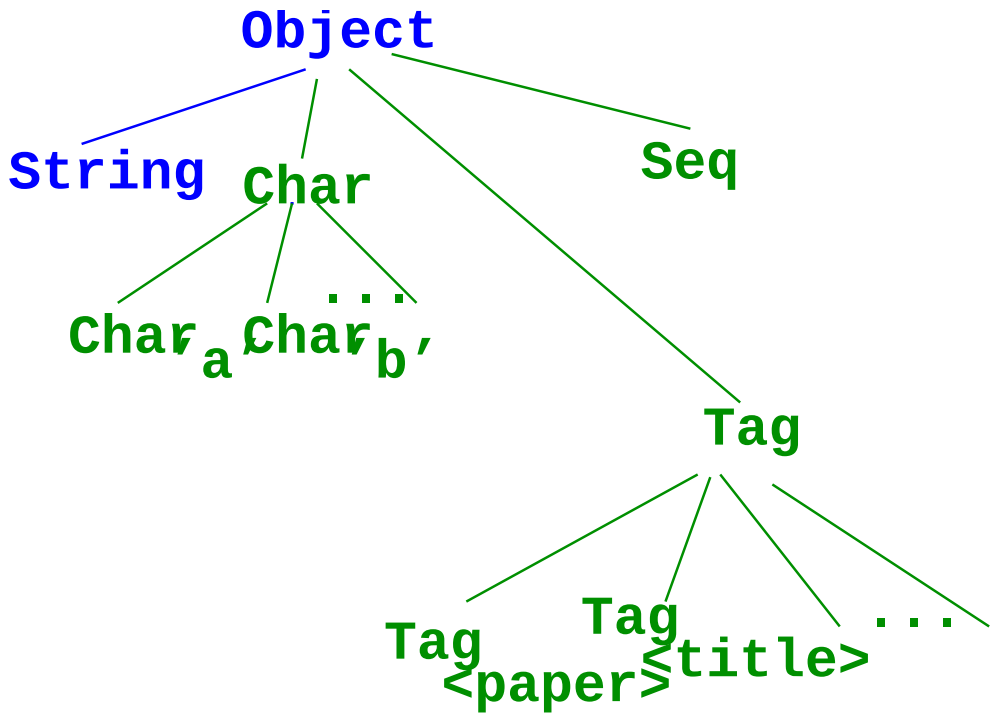
XML representation: types

$$
\begin{array}{lll}
a & ::= & t ::= \\
 & \texttt{new C}(\overline{a}) & \texttt{<}(a)\texttt{>}[\overline{t}] \\
 & [\overline{t}] & \\
A & ::= & T ::= \\
 & C & \texttt{<}(A)\texttt{>}[T] \\
 & [T] & T_1, T_2 \\
 & & \ldots
\end{array}
$$



XML representation: types

$$
\begin{array}{lll}
a & ::= & t ::= \\
\quad \texttt{new C}(\overline{a}) & & \quad \langle (a) \rangle [\overline{t}] \\
\quad [\overline{t}] & & \\
A & ::= & T ::= \\
\quad C & & \quad \langle (A) \rangle [T] \\
\quad [T] & & \quad T_1, T_2 \\
 & & \quad \ldots
\end{array}
$$




```
pcdata = [ pcchar* ]
pcchar = [ <(Char)>[] ]
xml = [ [pcchar | element]* ]
element = [ <(Tag)>[xml] ]
```

XML representation: values

XML:

```
<paper>
   <title>Mixins</title>
   <author>Flatt</author>
</paper>
```

XML representation: values

XML:

```
<paper>
   <title>Mixins</title>
   <author>Flatt</author>
</paper>
```

Abstract syntax:

```
[ ⟨(new Tag<paper>)⟩ [
    ⟨(new Tag<title>)⟩ [
      ⟨(new Char'M')⟩[], ⟨(new Char'i')⟩[], ...]
    ⟨(new Tag<autor>)⟩ [
      ⟨(new Char'F')⟩[], ⟨(new Char'l')⟩[], ...]
  ] ]
```

XML representation: values

XML:

```
<paper>
   <title>Mixins</title>
   <author>Flatt</author>
</paper>
```

Abstract syntax:

```
[ ‹(new Tag<paper>)› [
    ‹(new Tag<title>)› [
      ‹(new Char'M')›[], ‹(new Char'i')›[], ...]
    ‹(new Tag<autor>)› [
      ‹(new Char'F')›[], ‹(new Char'l')›[], ...]
  ] ]
```

Concrete syntax:

```
[ <paper>[
    <title>[‘Mixins‘]
    <author>[‘Flatt‘] ] ]
```

An example in XTATIC

```
[Title*] getEcoopTitles ([Paper*] ps) {
  [Title*] result = [];
  while (!ps.Equals([])) {
    match (ps) {
      case [ <paper>[Title t, Author*,
                    <inproc>[‘ECOOP‘, pcdata] ],
             Paper* rest  ]:
        result = [result, t];
        ps = rest;
      case [ Paper, Paper* rest ]:
        ps = rest;
      case []:  ;
    }
  return result;
  }
}
```

Summary

XTATIC extends a class-based OO type system with RE types:

- a pure OO program is a valid XTATIC program
- XDUCE-style pattern matching
- objects and trees can “interweave”
- XTATIC subtyping combines both OO and RE subtyping

Related work

XML data binding:

- JAXB, Castor, Relaxer
- HaXML, XMλ

OO type system extensions:

- Pizza, GJ
- ODMG

Semistructural data:

- OEM, Ozone

Static guarantees for XML processing:

- XDuce, CDuce
- XQuery
- JWig