SPACE-EFFICIENCY FOR ABSTRACT GRADUAL TYPING?

With Felipe Bañados Schwerter and Alison M. Clark
University of British Columbia

GRADUAL TYPING

Less Typed

More Typed

Sound Reasoning Principles

MIXED PROGRAMMING

```
def f(x:int) = x + 2
def h(g) = g(true)
h(f)
```

FROM MIXED TO GRADUAL

Mixed-Type Program

```
def f(x:int) = x + 2
def h(g) = g(true)
h(f)
```

desugar

Gradually-Typed Program

```
def f(x:int) = x + 2
def h(g:?) =g(true:?):?
h(f:?)
```

[Siek & Taha, 2006]

[Wadler & Findler, 2009]

FROM MIXED TO GRADUAL

Mixed-Type Program

```
def f(x:int) = x + 2
def h(g) = g(true)
h(f)
```

desugar

Gradually-Typed Program

```
def f(x:int) = x + 2
def h(g:?) =g(true:?):?
h(f:?)
```

Imprecise Types

[Siek & Taha, 2006]

[Wadler & Findler, 2009]

FROM GRADUAL TO CASTS

Gradually-Typed Program

```
def f(x:int) = x + 2
def h(g:?) =g(true:?):?
h(f:?)
```

elaborate

Cast Program

```
def f(x:int) = x + 2
def h(g:?) =(<?→?⇐?>g)(<?⇐bool>true)
h(<?⇐bool>f)
```

[Siek & Taha, 2006]

[Wadler & Findler, 2009]

FROM GRADUAL TO CASTS

Gradually-Typed Program

```
def f(x:int) = x + 2
def h(g:?) =g(true:?):?
h(f:?)
```

elaborate

Casts

Cast Program

```
def f(x:int) = x + 2
def h(g:?) =(<?→?⇐?>g)(<?⇐bool>true)
h(<?⇐bool>f)
```

[Siek & Taha, 2006]

[Wadler & Findler, 2009]

DEATH OF A TAIL CALL

$$even : \texttt{Dyn} \rightarrow \texttt{Dyn} \quad \stackrel{\text{def}}{=} \quad \lambda n{:}\texttt{Dyn}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{true}\ \ \textsf{else}\ odd\ (n - 1)$$

$$odd : \texttt{Int} \rightarrow \texttt{Bool} \quad \stackrel{\text{def}}{=} \quad \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ even\ (n - 1)$$

Herman et al. Space-Efficient Gradual Typing

DEATH OF A TAIL CALL

$$even : \texttt{Dyn} \to \texttt{Dyn} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Dyn}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{true}\ \ \textsf{else}\ odd\ (n-1)$$

$$odd : \texttt{Int} \to \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ even\ (n-1)$$

"Dyn" is pronounced "?"

Herman et al. Space-Efficient Gradual Typing

DEATH OF A TAIL CALL

$$even : \texttt{Dyn} \rightarrow \texttt{Dyn} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Dyn}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{true}\ \ \textsf{else}\ odd\ (n - 1)$$

$$odd : \texttt{Int} \rightarrow \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ even\ (n - 1)$$

Cast Insertion

$$odd_c : \texttt{Int} \rightarrow \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ \langle\texttt{Bool}\rangle(even\ (\langle\texttt{Dyn}\rangle\ (n-1)))$$

Herman et al. Space-Efficient Gradual Typing

DEATH OF A TAIL CALL

$$even : \texttt{Dyn} \to \texttt{Dyn} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Dyn}.\ \text{if } (n = 0) \text{ then } \texttt{true} \text{ else } odd\ (n-1)$$

$$odd : \texttt{Int} \to \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \text{if } (n = 0) \text{ then } \texttt{false} \text{ else } even\ (n-1)$$

looks like tail recursion!

Cast Insertion

$$odd_c : \texttt{Int} \to \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \text{if } (n = 0) \text{ then } \texttt{false} \text{ else } \langle\texttt{Bool}\rangle (even\ (\langle\texttt{Dyn}\rangle (n-1)))$$

Herman et al. Space-Efficient Gradual Typing

DEATH OF A TAIL CALL

$$even : \texttt{Dyn} \rightarrow \texttt{Dyn} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Dyn}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{true}\ \textsf{else}\ odd\ (n-1)$$

$$odd : \texttt{Int} \rightarrow \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ even\ (n-1)$$

looks like tail recursion!

Cast Insertion

$$odd_c : \texttt{Int} \rightarrow \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ \langle\texttt{Bool}\rangle(even\ (\langle\texttt{Dyn}\rangle\ (n-1)))$$

uh oh!

Herman et al. Space-Efficient Gradual Typing

WRAP VALUES OVER AND OVER!

$$
\begin{aligned}
\mathit{evenk}_c : \texttt{Int} \to \texttt{Bool} \quad &\stackrel{\text{def}}{=} \quad \lambda n\!:\!\texttt{Int}.\ \lambda k\!:\!(\texttt{Dyn} \to \texttt{Dyn}).\\
&\qquad \text{if } (n = 0)\\
&\qquad \text{then } \langle\texttt{Bool}\rangle\ (k\ (\langle\texttt{Dyn}\rangle\ \texttt{true}))\\
&\qquad \text{else } \mathit{oddk}_c\ (n-1)\ (\langle\texttt{Bool} \to \texttt{Bool}\rangle\ k)
\end{aligned}
$$

$$
\begin{aligned}
\mathit{oddk}_c : \texttt{Int} \to \texttt{Bool} \quad &\stackrel{\text{def}}{=} \quad \lambda n\!:\!\texttt{Int}.\ \lambda k\!:\!(\texttt{Bool} \to \texttt{Bool}).\\
&\qquad \text{if } (n = 0)\\
&\qquad \text{then } (k\ \texttt{false})\\
&\qquad \text{else } \mathit{evenk}_c\ (n-1)\ (\langle\texttt{Dyn} \to \texttt{Dyn}\rangle\ k)
\end{aligned}
$$

Herman et al. Space-Efficient Gradual Typing

WRAP VALUES OVER AND OVER!

$$
\begin{aligned}
evenk_c : \texttt{Int} \rightarrow \texttt{Bool} \quad &\stackrel{\text{def}}{=} \quad \lambda n{:}\texttt{Int}.\ \lambda k{:}(\texttt{Dyn} \rightarrow \texttt{Dyn}).\\
&\qquad \text{if } (n = 0)\\
&\qquad \text{then } \langle \texttt{Bool} \rangle\ (k\ (\langle \texttt{Dyn} \rangle\ \texttt{true}))\\
&\qquad \text{else } oddk_c\ (n - 1)\ (\langle \texttt{Bool} \rightarrow \texttt{Bool} \rangle\ k)
\end{aligned}
$$

$$
\begin{aligned}
oddk_c : \texttt{Int} \rightarrow \texttt{Bool} \quad &\stackrel{\text{def}}{=} \quad \lambda n{:}\texttt{Int}.\ \lambda k{:}(\texttt{Bool} \rightarrow \texttt{Bool}).\\
&\qquad \text{if } (n = 0)\\
&\qquad \text{then } (k\ \texttt{false})\\
&\qquad \text{else } evenk_c\ (n - 1)\ (\langle \texttt{Dyn} \rightarrow \texttt{Dyn} \rangle\ k)
\end{aligned}
$$

Wrap on each call!

Herman et al. Space-Efficient Gradual Typing

SOLUTIONS

Space-Efficient Gradual Typing

David Herman · Aaron Tomb · Cormac Flanagan

Coercions

Threesomes, With and Without Blame *

Jeremy G. Siek
University of Colorado at Boulder
jeremy.siek@colorado.edu

Philip Wadler
University of Edinburgh
wadler@inf.ed.ac.uk

Threesomes

SOLUTIONS

Space-Efficient Gradual Typing

David Herman · Aaron Tomb · Cormac Flanagan

Coercions

Address Only *One* Gradual Type Discipline

Threesomes, With and Without Blame *

Jeremy G. Siek
University of Colorado at Boulder
jeremy.siek@colorado.edu

Philip Wadler
University of Edinburgh
wadler@inf.ed.ac.uk

Threesomes

MANY GRADUAL DISCIPLINES!

Less Typed		More Typed
Unityped	[Siek and Taha 06]	Simple
Unityped	[Siek and Taha 08]	Subtyping
Unityped	[Siek and Vachharajani 08]	Hindley/Milner
Simple	[Lehman et al. 17] [Dunfield et al. 17]	Refinements
Simple	[Bañados et al. 14]	Type&Effect
Simple	[Disney and Flanagan 11]	Security

static type system & type safety proof

interpretation of gradual types

Abstracting Gradual Typing

Ronald Garcia [*] Alison M. Clark [†]
Software Practices Lab
Department of Computer Science
University of British Columbia, Canada
{rxg,amclark1}@cs.ubc.ca

Éric Tanter [‡]
PLEIAD Laboratory
Computer Science Department (DCC)
University of Chile, Chile
etanter@dcc.uchile.cl

POPL 2016

gradual language

TYPE SYSTEM

DYNAMIC SEMANTICS

static type system & type safety proof

interpretation of gradual types

Abstracting Gradual Typing

Ronald Garcia [*] Alison M. Clark [†]
Software Practices Lab
Department of Computer Science
University of British Columbia, Canada
{rxg,amclark1}@cs.ubc.ca

Éric Tanter [‡]
PLEIAD Laboratory
Computer Science Department (DCC)
University of Chile, Chile
etanter@dcc.uchile.cl

POPL 2016

gradual language

TYPE SYSTEM	DYNAMIC SEMANTICS

TL;DR
(TOO LONG; DIDN'T READ)

SOME GOOD NEWS

$$
\begin{aligned}
\mathit{evenk}_c : \texttt{Int} \to \texttt{Bool} \quad &\stackrel{\text{def}}{=} \quad \lambda n{:}\texttt{Int}.\ \lambda k{:}(\texttt{Dyn} \to \texttt{Dyn}). \\
&\qquad\quad \text{if } (n = 0) \\
&\qquad\quad \text{then } \langle\texttt{Bool}\rangle\ (k\ (\langle\texttt{Dyn}\rangle\ \texttt{true})) \\
&\qquad\quad \text{else } \mathit{oddk}_c\ (n-1)\ (\langle\texttt{Bool} \to \texttt{Bool}\rangle\ k)
\end{aligned}
$$

$$
\begin{aligned}
\mathit{oddk}_c : \texttt{Int} \to \texttt{Bool} \quad &\stackrel{\text{def}}{=} \quad \lambda n{:}\texttt{Int}.\ \lambda k{:}(\texttt{Bool} \to \texttt{Bool}). \\
&\qquad\quad \text{if } (n = 0) \\
&\qquad\quad \text{then } (k\ \texttt{false}) \\
&\qquad\quad \text{else } \mathit{evenk}_c\ (n-1)\ (\langle\texttt{Dyn} \to \texttt{Dyn}\rangle\ k)
\end{aligned}
$$

Wrap on each call!

Herman et al. Space-Efficient Gradual Typing

SOME GOOD NEWS

$evenk_c$: `Int` $\rightarrow$ `Bool` ... `Dyn`).

(⟨`Dyn`⟩ `true`))

1) (⟨`Bool` $\rightarrow$ `Bool`⟩ k)

$oddk_c$: `Int` $\rightarrow$ `Bool` ... $\rightarrow$ `Bool`).

- 1) (⟨`Dyn` $\rightarrow$ `Dyn`⟩ k)

Wrap on each call!

Herman et al. Space-Efficient Gradual Typing

CONSERVATION OF GOODNESS

$$even : \texttt{Dyn} \rightarrow \texttt{Dyn} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Dyn}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{true}\ \ \textsf{else}\ odd\ (n - 1)$$

$$odd : \texttt{Int} \rightarrow \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ even\ (n - 1)$$

looks like a tail call!

Cast Insertion

$$odd_c : \texttt{Int} \rightarrow \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ \langle\texttt{Bool}\rangle\ (even\ (\langle\texttt{Dyn}\rangle\ (n-1)))$$

uh oh!

Herman et al. Space-Efficient Gradual Typing

CONSERVATION OF GOODNESS

$even : \texttt{Dyn} \longrightarrow \texttt{Dyn}$ $\overset{\text{def}}{}$ … else $odd\ (n-1)$

$odd : \texttt{Int} \longrightarrow \texttt{Bool}$ … else $even\ (n-1)$

looks like a tail call!

$odd_c : \texttt{Int} \rightarrow \texttt{Bool} \overset{\text{def}}{=} \lambda n{:}\texttt{Int}.$ if $(n = 0)$ then $\texttt{false}$ else $\langle\texttt{Bool}\rangle\ (even\ (\langle\texttt{Dyn}\rangle\ (n-1)))$

uh oh!

Herman et al. Space-Efficient Gradual Typing

AGT-BASED DISCIPLINES

Less Typed ⟷ More Typed

Unityped	[Garcia et al. 16]	Y	Simple
Unityped	[Garcia et al. 16]	N	Subtyping
Simple	[Lehman et al. 17]	?	Refinements
Simple	[Toro et al. 18]	Y	Security
Unityped	[Toro et al. 19]	Y	Parametricity

AGT-BASED DISCIPLINES

Less Typed ⟷ More Typed

Unityped	[Garcia et al. 16]	Y	Simple
Unityped	[Garcia et al. 16]	N	Subtyping
Simple	[Lehman et al. 17]	?	Refinements
Simple	[Toro et al. 18]	Y	Security
Unityped	[Toro et al. 19]	Y	Parametricity

CASE STUDY: RECORD SUBTYPING

GTFL$_{\lesssim}$

$$S \in \text{GType} \quad ::= \quad ? \mid \text{Int} \mid \text{Bool} \mid S \rightarrow S \mid [\overline{l : S}] \mid [\overline{l : S}, ?]$$

GTFL$_{\lesssim}$

records with
width/depth subtyping

$$S \in \text{GType} \quad ::= \quad ? \mid \text{Int} \mid \text{Bool} \mid S \to S \mid [\overline{l : S}] \mid [\overline{l : S}, ?]$$

$\mathbf{GTFL}_{\lesssim}$

records with width/depth subtyping

$$S \in \text{GType} \quad ::= \quad ? \mid \text{Int} \mid \text{Bool} \mid S \rightarrow S \mid [\overline{l : S}] \mid [\overline{l : S}, ?]$$

“unknown type”

GTFL$_{\lesssim}$

records with
width/depth subtyping

$$S \in \text{GType} \quad ::= \quad ? \mid \text{Int} \mid \text{Bool} \mid S \to S \mid [\overline{l : S}] \mid [\overline{l : S}, ?]$$

“unknown
type”

“unknown
row”

EXAMPLE 1

$$
\begin{aligned}
&\text{let sum (hasM : Bool) (x : [f : Int, ?]) =} \\
&\qquad\qquad \text{if hasM then x.f + x.m else x.f + x.q} \\
&\text{in (sum true [f = 6, m = 2]) + (sum false [f = 6, q = 2])}
\end{aligned}
$$

EXAMPLE 1

must have an integral “f” field

```
let sum (hasM : Bool) (x : [f : Int, ?]) =
        if hasM then x.f + x.m else x.f + x.q
in (sum true [f = 6, m = 2]) + (sum false [f = 6, q = 2])
```

statically checked

EXAMPLE 1

may have additional fields

```
let sum (hasM : Bool) (x : [f : Int, ?]) =
        if hasM then x.f + x.m else x.f + x.q
in (sum true [f = 6, m = 2]) + (sum false [f = 6, q = 2])
```

dynamically checked

EXAMPLE 1

may have additional fields

$$
\begin{aligned}
&\text{let sum (hasM : Bool) (x : [f : Int, ?]) =}\\
&\qquad\text{if hasM then x.f + x.m else x.f + x.q}\\
&\text{in (sum true [f = 6, m = 2]) + (sum false [f = 6, q = 2])}
\end{aligned}
$$

dynamically checked

"Type-constrained downcasting"

EXAMPLE 2

let x : $[a : \text{Int}, b : \text{Bool}] = [a = 5, b = \text{false}]$ in
let y : $[a : \text{Int}, ?] =$ x in
let z : $[a : \text{Int}, b : \text{Bool}] =$ y in
z.b

EXAMPLE 2

let x : $[a : \text{Int}, b : \text{Bool}] = [a = 5, b = \text{false}]$ in
let y : $[a : \text{Int}, ?] =$ x in
let z : $[a : \text{Int}, b : \text{Bool}] =$ y in
z.b

runtime
"type check"

EXAMPLE 2

let x : $[a : \text{Int}, b : \text{Bool}] = [a = 5, b = \text{false}]$ in
let y : $[a : \text{Int}, ?] =$ x in
let z : $[a : \text{Int}, b : \text{Bool}] =$ y in
z.b

runtime
“type check”

Type Checks
Runs Successfully

EXAMPLE 2

let x : $[a$: Int, ~~b : Bool~~$] = [a = 5, b =$ false$]$ in
let y : $[a$: Int, $?] =$ x in
let z : $[a$: Int, b : Bool$] =$ y in
z.b

runtime
"type check"

Type Checks
Runs Successfully

EXAMPLE 3

$[a : \mathsf{Int}, b : \mathsf{Bool}] <: [a : \mathsf{Int}]$

$\mathsf{let}\ \mathsf{x} : [a : \mathsf{Int}] = [a = 5, b = \mathsf{false}]\ \mathsf{in}$
$\mathsf{let}\ \mathsf{y} : [a : \mathsf{Int}, ?] = \mathsf{x}\ \mathsf{in}$
$\mathsf{let}\ \mathsf{z} : [a : \mathsf{Int}, b : \mathsf{Bool}] = \mathsf{y}\ \mathsf{in}$
$\mathsf{z}.b$

runtime
“type check”

EXAMPLE 3

$[a : \mathsf{Int}, b : \mathsf{Bool}] <: [a : \mathsf{Int}]$

let x : $[a : \mathsf{Int}] = [a = 5, b = \mathsf{false}]$ in
let y : $[a : \mathsf{Int}, ?]$ = x in
let z : $[a : \mathsf{Int}, b : \mathsf{Bool}]$ = y in
z.b

runtime
"type check"

Type Checks
Runtime Error!

TYPES: STATIC *AND GRADUAL*

records with
width/depth subtyping

$$S \in \mathrm{GType} \quad ::= \quad ? \mid \mathrm{Int} \mid \mathrm{Bool} \mid S \rightarrow S \mid [\overline{l : S}] \mid [\overline{l : S}, ?]$$

“unknown
type”

“unknown
row”

TYPES: STATIC *AND GRADUAL*

records with
width/depth subtyping

$$
\begin{array}{rcl}
T \in \text{Type} & ::= & \text{Int} \mid \text{Bool} \mid T \rightarrow T \mid [\overline{l : T}] \\
S \in \text{GType} & ::= & ? \mid \text{Int} \mid \text{Bool} \mid S \rightarrow S \mid [\overline{l : S}] \mid [\overline{l : S}, ?]
\end{array}
$$

$$\textsc{Type} \subseteq \textsc{GType}$$

“unknown type”

“unknown row”

TYPES: STATIC *AND GRADUAL*

records with
width/depth subtyping

$$
\begin{aligned}
T \in \text{Type} \quad &::= \quad \text{Int} \mid \text{Bool} \mid T \to T \mid [\overline{l : T}] \\
S \in \text{GType} \quad &::= \quad ? \mid \text{Int} \mid \text{Bool} \mid S \to S \mid [\overline{l : S}] \mid [\overline{l : S}, ?]
\end{aligned}
$$

$$\textsc{Type} \subseteq \textsc{GType}$$

“unknown
type”

every static type is also
a gradual type

“unknown
row”

PRECISION

$$\frac{}{S \sqsubseteq ?}$$

$$\frac{S \in \{\,\mathsf{Int}, \mathsf{Bool}\,\}}{S \sqsubseteq S}$$

$$\frac{S_{11} \sqsubseteq S_{21} \qquad S_{12} \sqsubseteq S_{22}}{S_{11} \rightarrow S_{12} \sqsubseteq S_{21} \rightarrow S_{22}}$$

$$\frac{\overline{S_{1i} \sqsubseteq S_{2i}}}{[\overline{l_i : S_{1i}}] \sqsubseteq [\overline{l_i : S_{2i}}]}$$

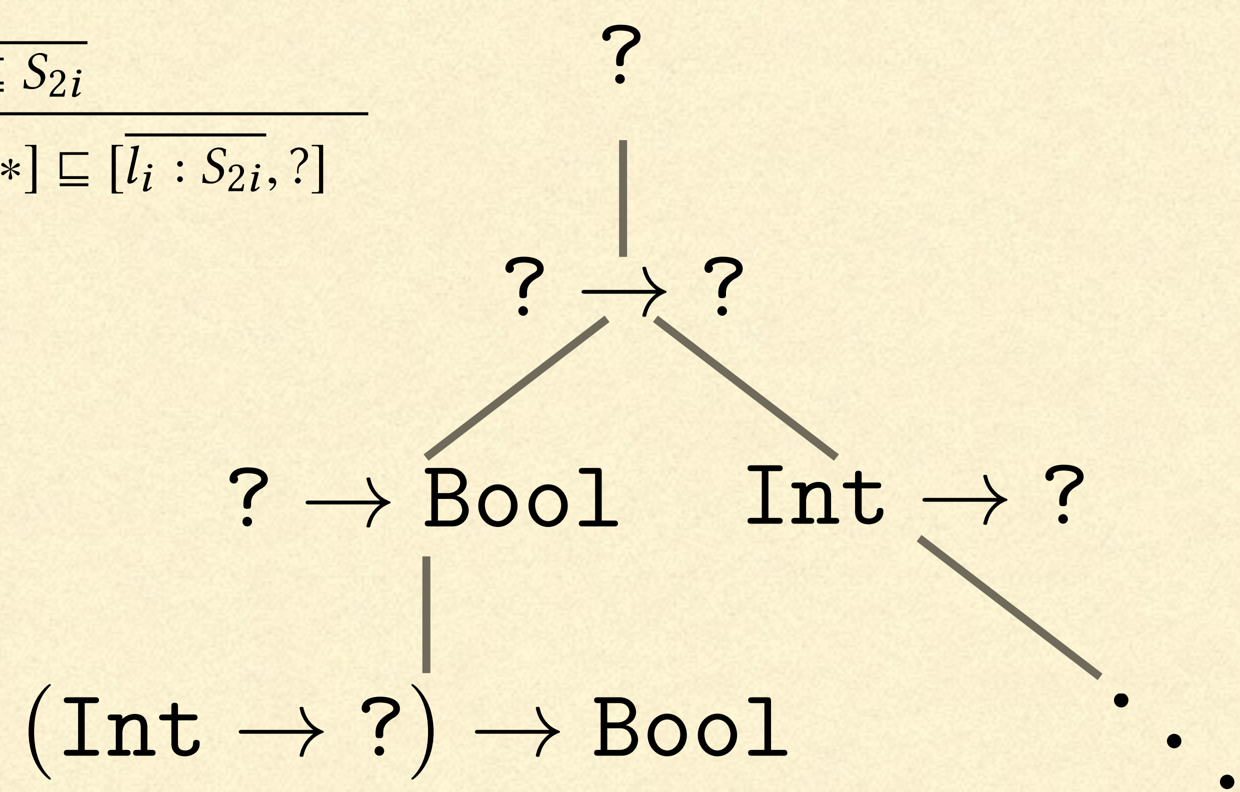
$$\frac{\overline{S_{1i} \sqsubseteq S_{2i}}}{[\overline{l_i : S_{1i}}, \overline{l_j : S_{1j}}, *] \sqsubseteq [\overline{l_i : S_{2i}}, ?]}$$

PRECISION

$$\frac{}{S \sqsubseteq ?} \qquad \frac{S \in \{\,\mathsf{Int}, \mathsf{Bool}\,\}}{S \sqsubseteq S} \qquad \frac{S_{11} \sqsubseteq S_{21} \qquad S_{12} \sqsubseteq S_{22}}{S_{11} \to S_{12} \sqsubseteq S_{21} \to S_{22}} \qquad \frac{\overline{S_{1i} \sqsubseteq S_{2i}}}{[\overline{l_i : S_{1i}}] \sqsubseteq [\overline{l_i : S_{2i}}]}$$

$$\frac{\overline{S_{1i} \sqsubseteq S_{2i}}}{[\overline{l_i : S_{1i}}, \overline{l_j : S_{1j}}, *] \sqsubseteq [\overline{l_i : S_{2i}}, ?]}$$

? type is most imprecise



PRECISION

$$\frac{}{S \sqsubseteq ?}$$

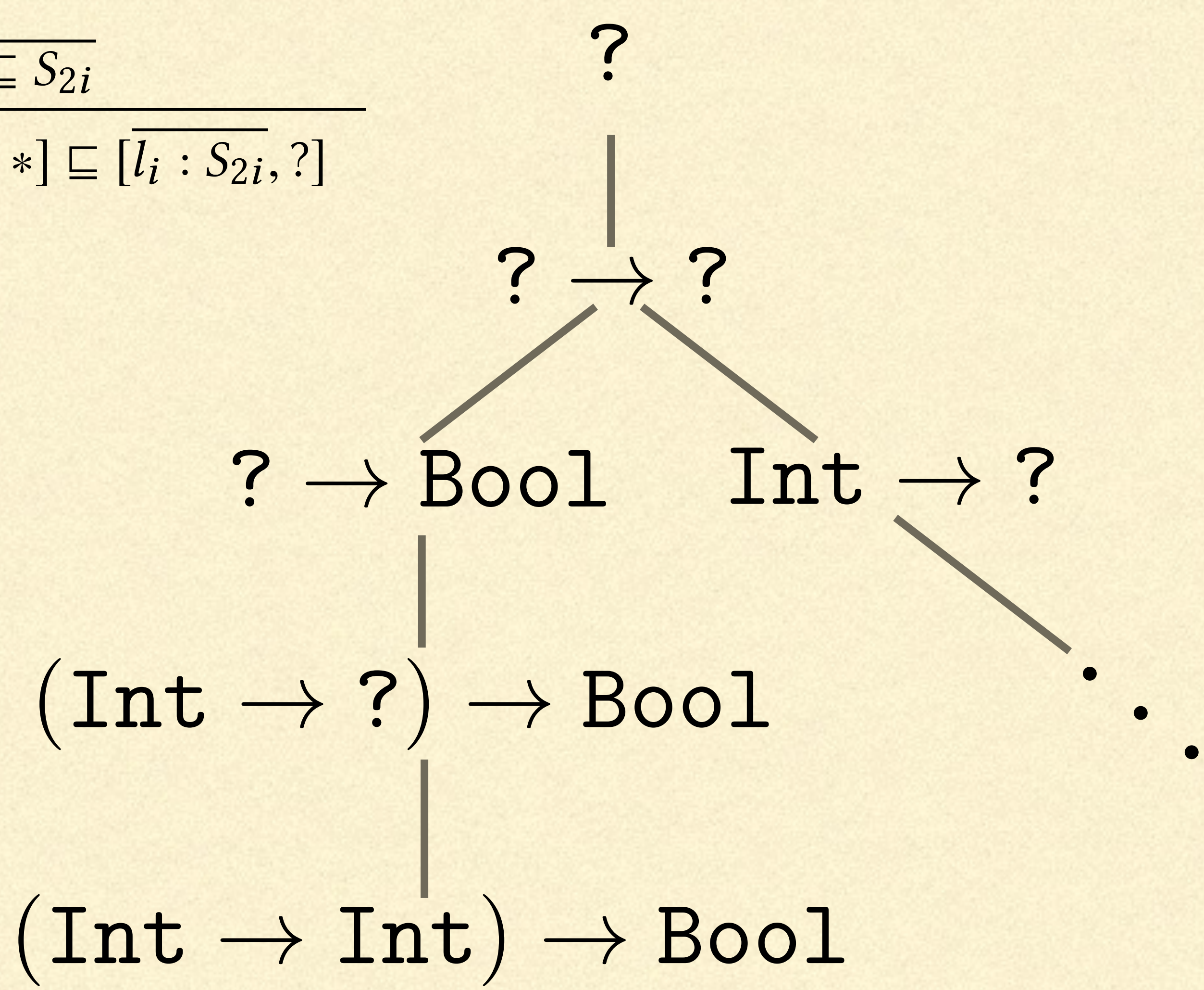
$$\frac{S \in \{\, \text{Int}, \text{Bool} \,\}}{S \sqsubseteq S}$$

$$\frac{S_{11} \sqsubseteq S_{21} \qquad S_{12} \sqsubseteq S_{22}}{S_{11} \to S_{12} \sqsubseteq S_{21} \to S_{22}}$$

$$\frac{\overline{S_{1i} \sqsubseteq S_{2i}}}{[\overline{l_i : S_{1i}}] \sqsubseteq [\overline{l_i : S_{2i}}]}$$

$$\frac{\overline{S_{1i} \sqsubseteq S_{2i}}}{[\overline{l_i : S_{1i}}, \overline{l_j : S_{1j}}, *] \sqsubseteq [\overline{l_i : S_{2i}}, ?]}$$

? type is most imprecise



Static Types are Least

PRECISION

$$\frac{}{S \sqsubseteq ?}$$

$$\frac{S \in \{\,\text{Int}, \text{Bool}\,\}}{S \sqsubseteq S}$$

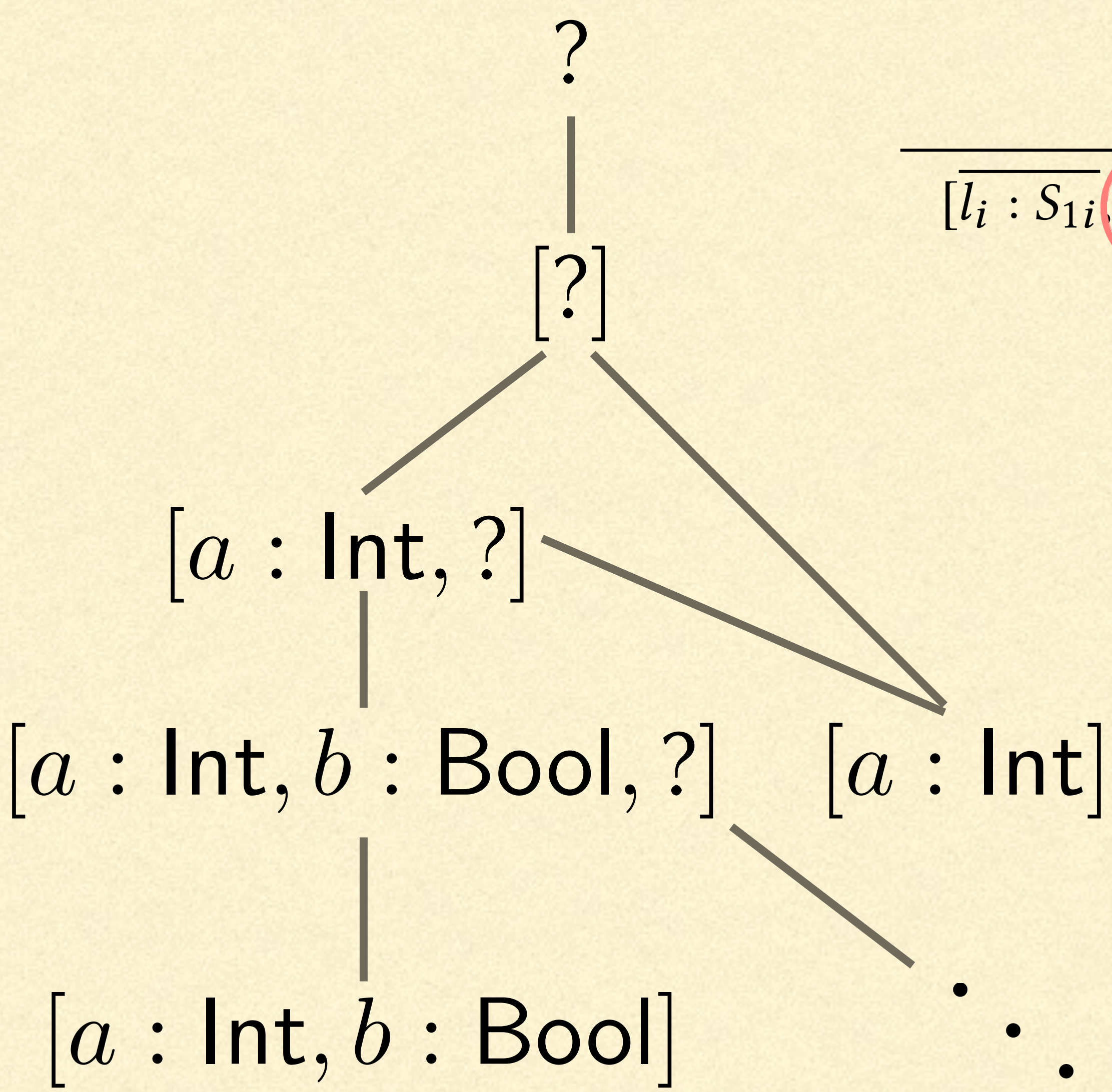
$$\frac{S_{11} \sqsubseteq S_{21} \quad S_{12} \sqsubseteq S_{22}}{S_{11} \to S_{12} \sqsubseteq S_{21} \to S_{22}}$$

$$\frac{\overline{S_{1i} \sqsubseteq S_{2i}}}{[\overline{l_i : S_{1i}}] \sqsubseteq [\overline{l_i : S_{2i}}]}$$

$$\frac{\overline{S_{1i} \sqsubseteq S_{2i}}}{[\overline{l_i : S_{1i}}, \overline{l_j : S_{1j}}, *] \sqsubseteq [\overline{l_i : S_{2i}}, ?]}$$

“*” stands for “?” or “”

? row adds imprecision about fields



CONSISTENT LIFTING

$$S_1 \lesssim S_2$$ Consistent Subtyping

if and only if $\sqsubseteq \quad \sqsubseteq$

$$T_1 <: T_2$$ Static Subtyping

For some T1 and T2

EXAMPLES

$$[a : \mathsf{Int}, b : \mathsf{Bool}] \lesssim [a : \mathsf{Int}]$$

$$\begin{array}{ccc} S_1 & \lesssim & S_2 \\ \sqsupseteq & & \sqsupseteq \\ T_1 & <: & T_2 \end{array}$$

EXAMPLES

$$[a : \mathsf{Int}, b : \mathsf{Bool}] \lesssim [a : \mathsf{Int}]$$

$$[a : \mathsf{Int}] \not\lesssim [a : \mathsf{Int}, b : \mathsf{Bool}]$$

$$\begin{array}{ccc} S_1 & \lesssim & S_2 \\ \sqsupseteq & & \sqsupseteq \\ T_1 & <: & T_2 \end{array}$$

EXAMPLES

$$[a : \mathsf{Int}, b : \mathsf{Bool}] \lesssim [a : \mathsf{Int}]$$

$$[a : \mathsf{Int}] \not\lesssim [a : \mathsf{Int}, b : \mathsf{Bool}]$$

$$[a : \mathsf{Int}, ?] \lesssim [a : \mathsf{Int}, b : \mathsf{Bool}]$$

$$\begin{array}{ccc} S_1 & \lesssim & S_2 \\ \sqsupseteq & & \sqsupseteq \\ T_1 & <: & T_2 \end{array}$$

EXAMPLES

$$[a : \mathsf{Int}, b : \mathsf{Bool}] \lesssim [a : \mathsf{Int}]$$

$$[a : \mathsf{Int}] \not\lesssim [a : \mathsf{Int}, b : \mathsf{Bool}]$$

$$[a : \mathsf{Int}, ?] \lesssim [a : \mathsf{Int}, b : \mathsf{Bool}]$$

$$[a : \mathsf{Int}, b : \mathsf{Bool}] \lesssim [a : \mathsf{Int}, ?]$$

$$\begin{array}{ccc} S_1 & \lesssim & S_2 \\ \sqsupseteq & & \sqsupseteq \\ T_1 & <: & T_2 \end{array}$$

EXAMPLES

$$[a : \mathsf{Int}, b : \mathsf{Bool}] \lesssim [a : \mathsf{Int}]$$

$$[a : \mathsf{Int}] \not\lesssim [a : \mathsf{Int}, b : \mathsf{Bool}]$$

$$[a : \mathsf{Int}, ?] \lesssim [a : \mathsf{Int}, b : \mathsf{Bool}]$$

$$[a : \mathsf{Int}, b : \mathsf{Bool}] \lesssim [a : \mathsf{Int}, ?]$$

$$[a : \mathsf{Int}] \not\lesssim [a : \mathsf{Int}, b : \mathsf{Bool}, ?]$$

$$\begin{array}{ccc} S_1 & \lesssim & S_2 \\ \sqsupseteq & & \sqsupseteq \\ T_1 & <: & T_2 \end{array}$$

EXAMPLES

$$S_1 \lesssim S_2$$
$$\sqsupseteq \qquad \sqsupseteq$$
$$T_1 <: T_2$$

$$[a : \mathsf{Int}, b : \mathsf{Bool}] \lesssim [a : \mathsf{Int}]$$

$$[a : \mathsf{Int}] \not\lesssim [a : \mathsf{Int}, b : \mathsf{Bool}]$$

$$[a : \mathsf{Int}, ?] \lesssim [a : \mathsf{Int}, b : \mathsf{Bool}]$$

$$[a : \mathsf{Int}, b : \mathsf{Bool}] \lesssim [a : \mathsf{Int}, ?]$$

$$[a : \mathsf{Int}] \not\lesssim [a : \mathsf{Int}, b : \mathsf{Bool}, ?]$$

$$[a : \mathsf{Int}, b : \mathsf{Bool}, ?] \lesssim [a : \mathsf{Int}, ?]$$

RUNTIME TYPE ENFORCEMENT

SEMANTICS OF GRADUAL TYPES

e.g.,

$$\gamma : \text{GT}\textsc{ype} \to \mathcal{P}^{+}(\text{T}\textsc{ype})$$

$$\gamma(S) = \{\, T \in \textbf{T}\textsc{ype} \mid T \sqsubseteq S \,\}$$

Concretization Function

Some critical properties:

$$\gamma(T) = \{\, T \,\}$$

$$S_1 \sqsubseteq S_2 \iff \gamma(S_1) \subseteq \gamma(S_2)$$

SEMANTICS OF GRADUAL TYPES

e.g.,

$$\alpha : \mathcal{P}^{+}(\textsc{Type}) \rightarrow \textsc{GType}$$

$$\alpha(C) = \sqcap \{ S \in \textsc{GType} \mid C \subseteq \gamma(S) \}$$

Abstraction Function

Some useful properties:

$\alpha(\{ T \}) = T$ (even better: $\alpha(\gamma(S)) = S$)

$$C_1 \subseteq C_2 \implies \alpha(C_1) \sqsubseteq \alpha(C_2)$$

EVIDENCE (I.E., "CASTS")

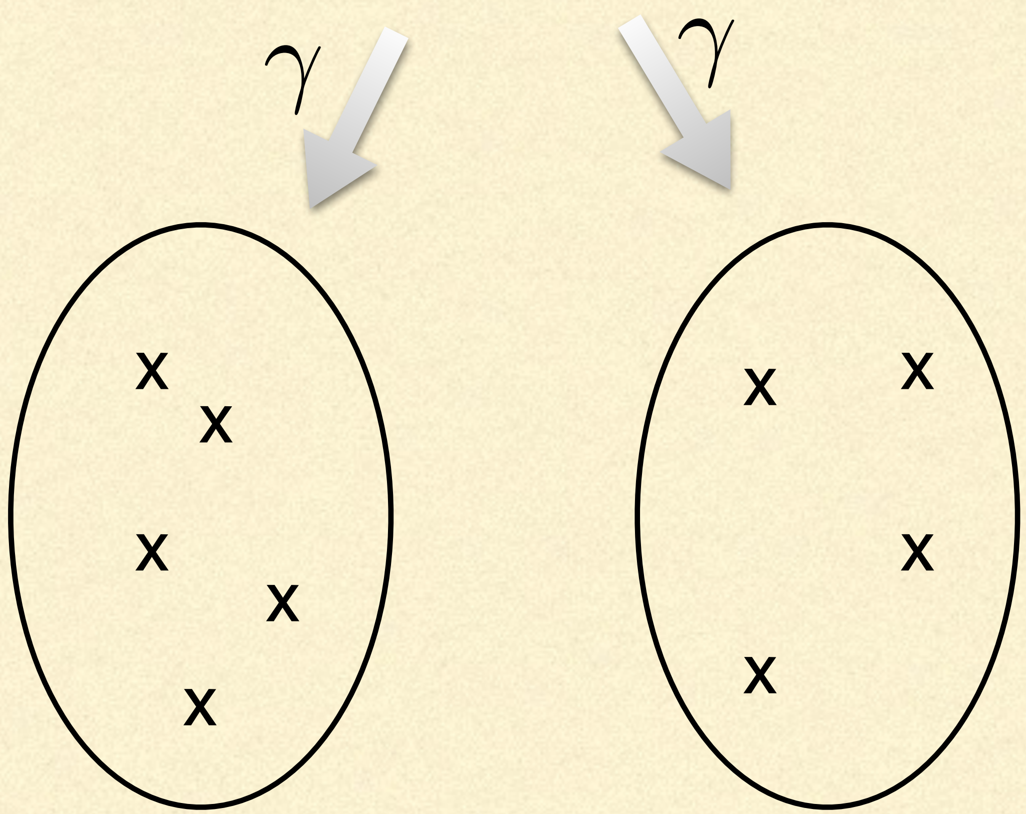
$$S_1 \lesssim S_2$$

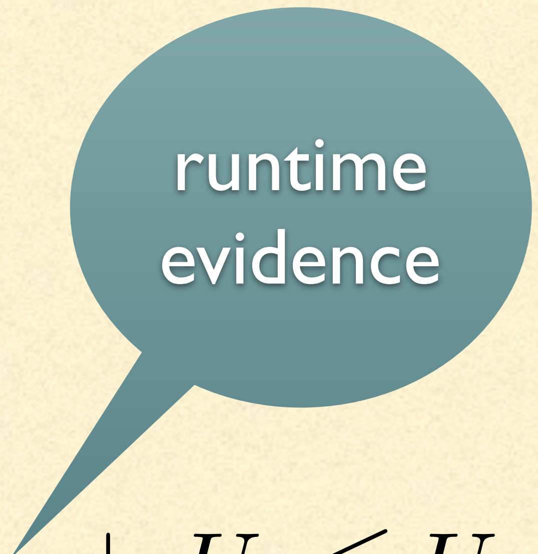
runtime evidence

$$\varepsilon_{12} \vdash U_1 \lesssim U_2$$

EVIDENCE (I.E., "CASTS")

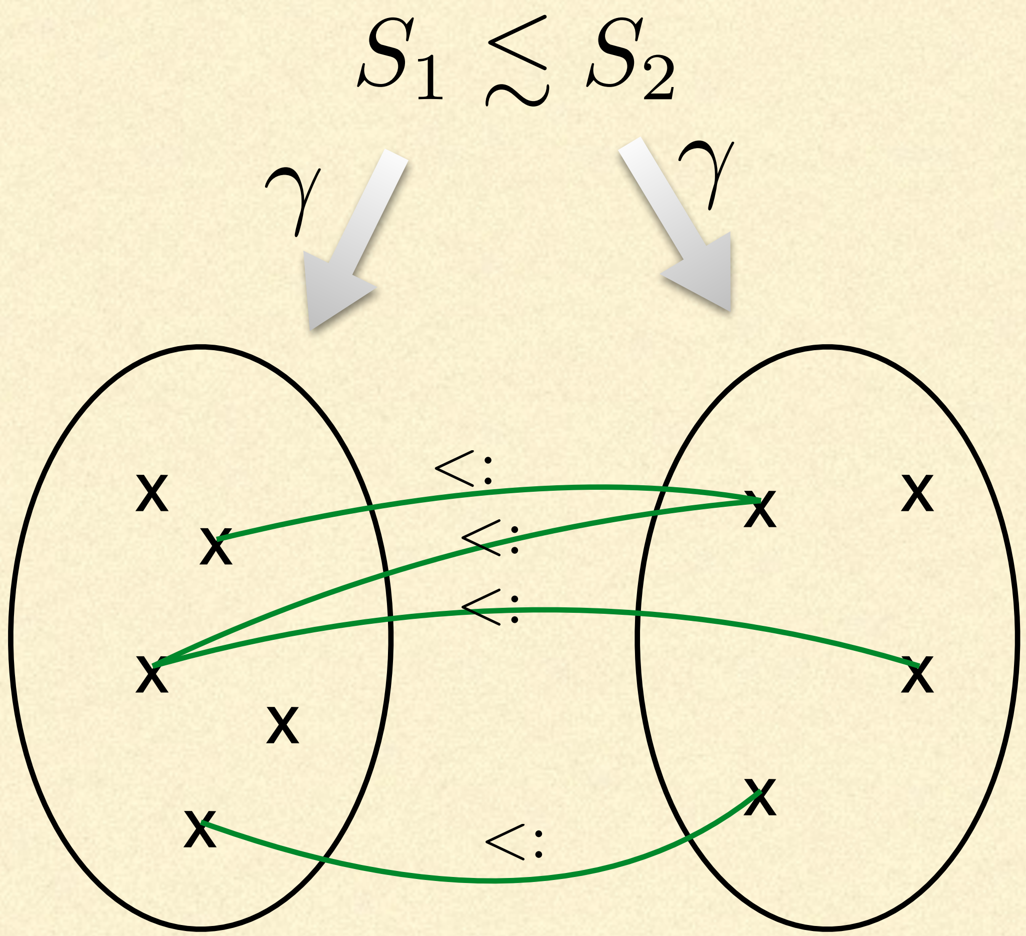
$$S_1 \lesssim S_2$$

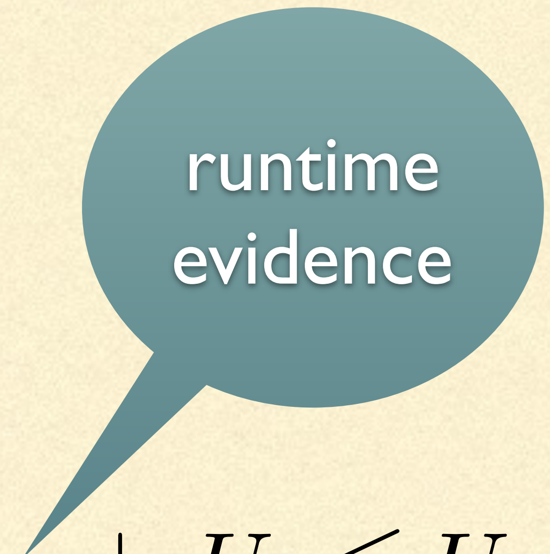




$$\varepsilon_{12} \vdash U_1 \lesssim U_2$$

EVIDENCE (I.E., "CASTS")

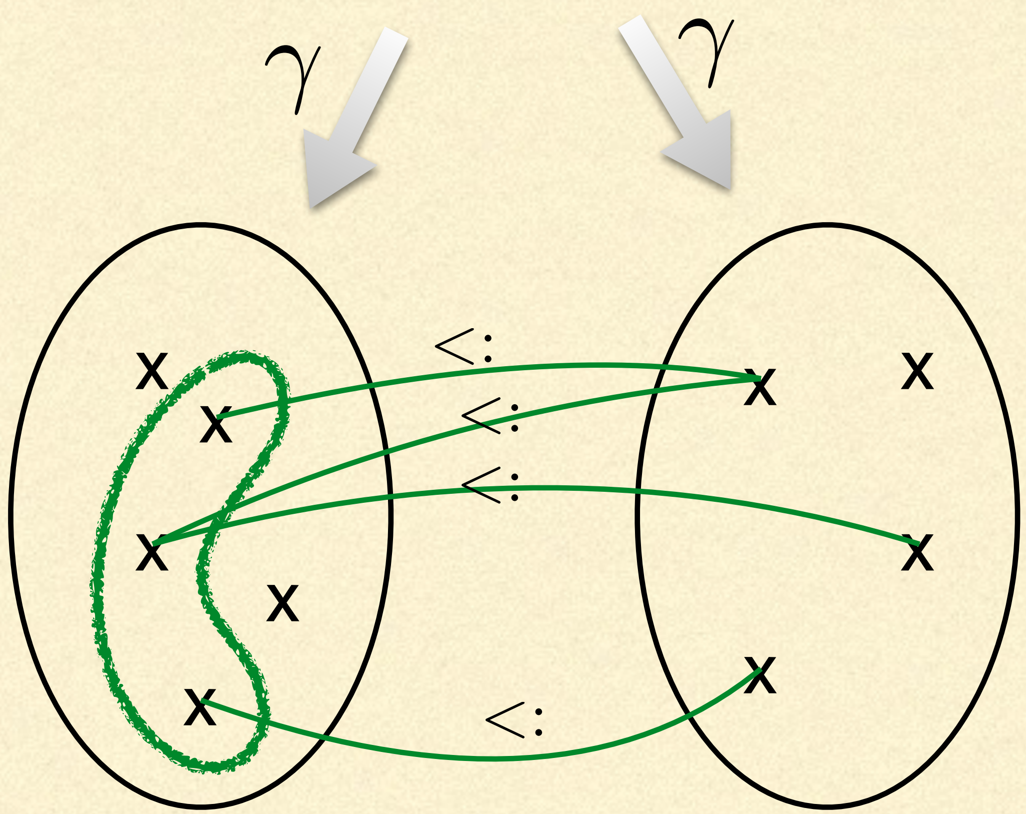


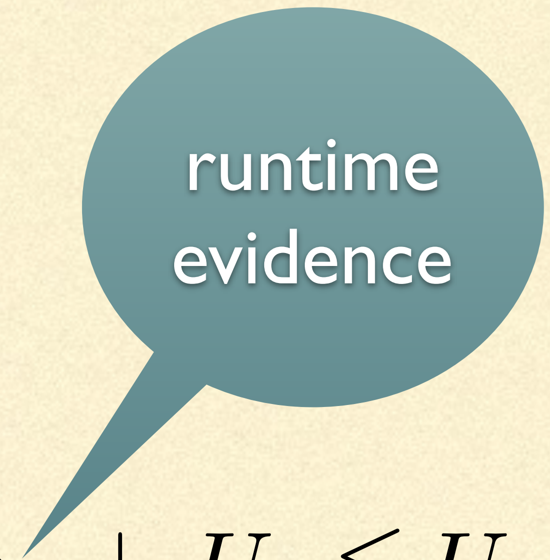


$$\varepsilon_{12} \vdash U_1 \lesssim U_2$$

EVIDENCE (I.E., “CASTS”)

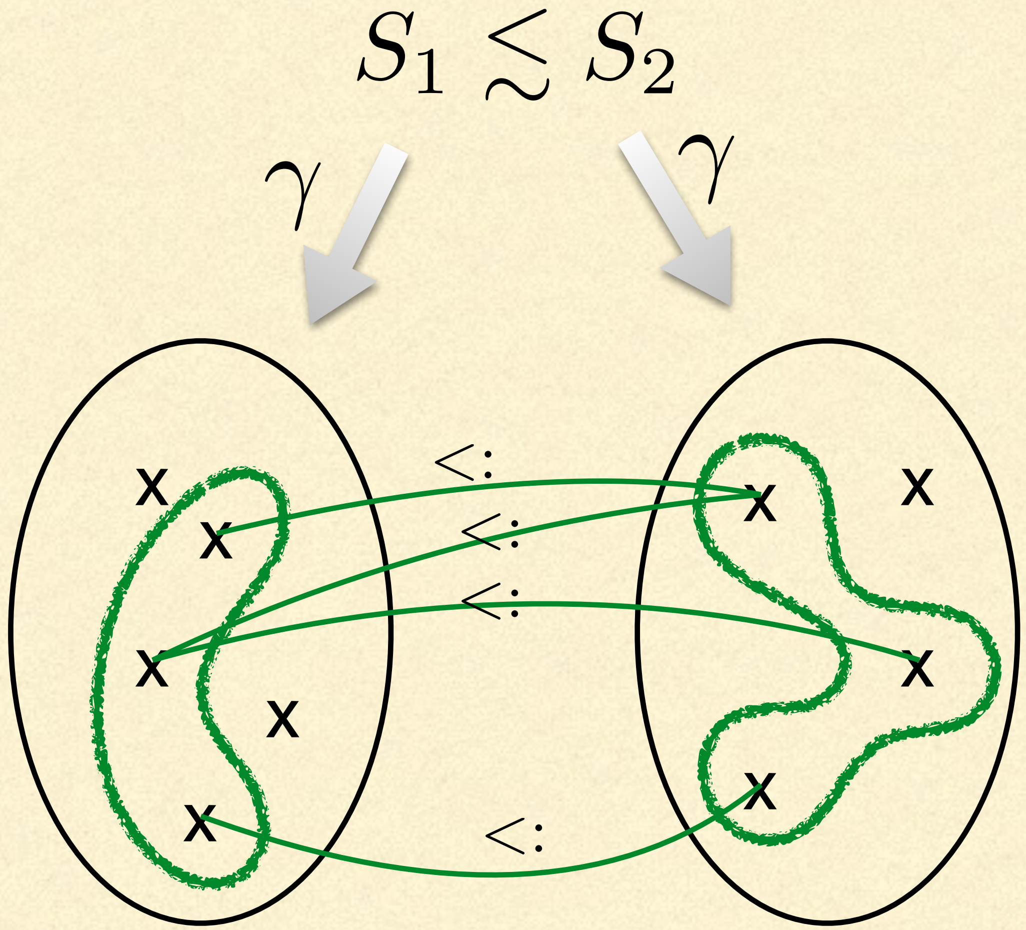
$$S_1 \lesssim S_2$$

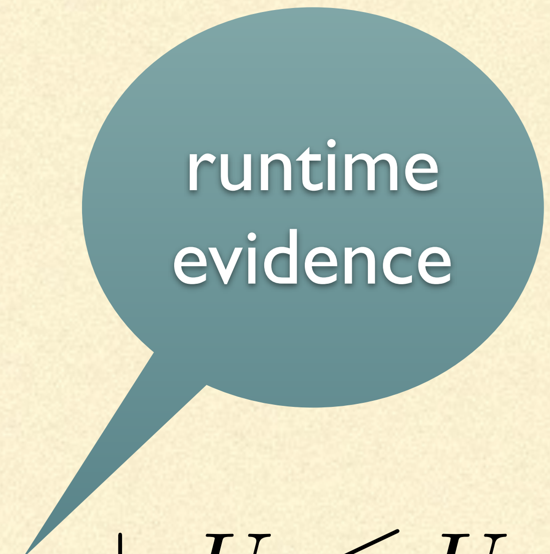




$$\varepsilon_{12} \vdash U_1 \lesssim U_2$$

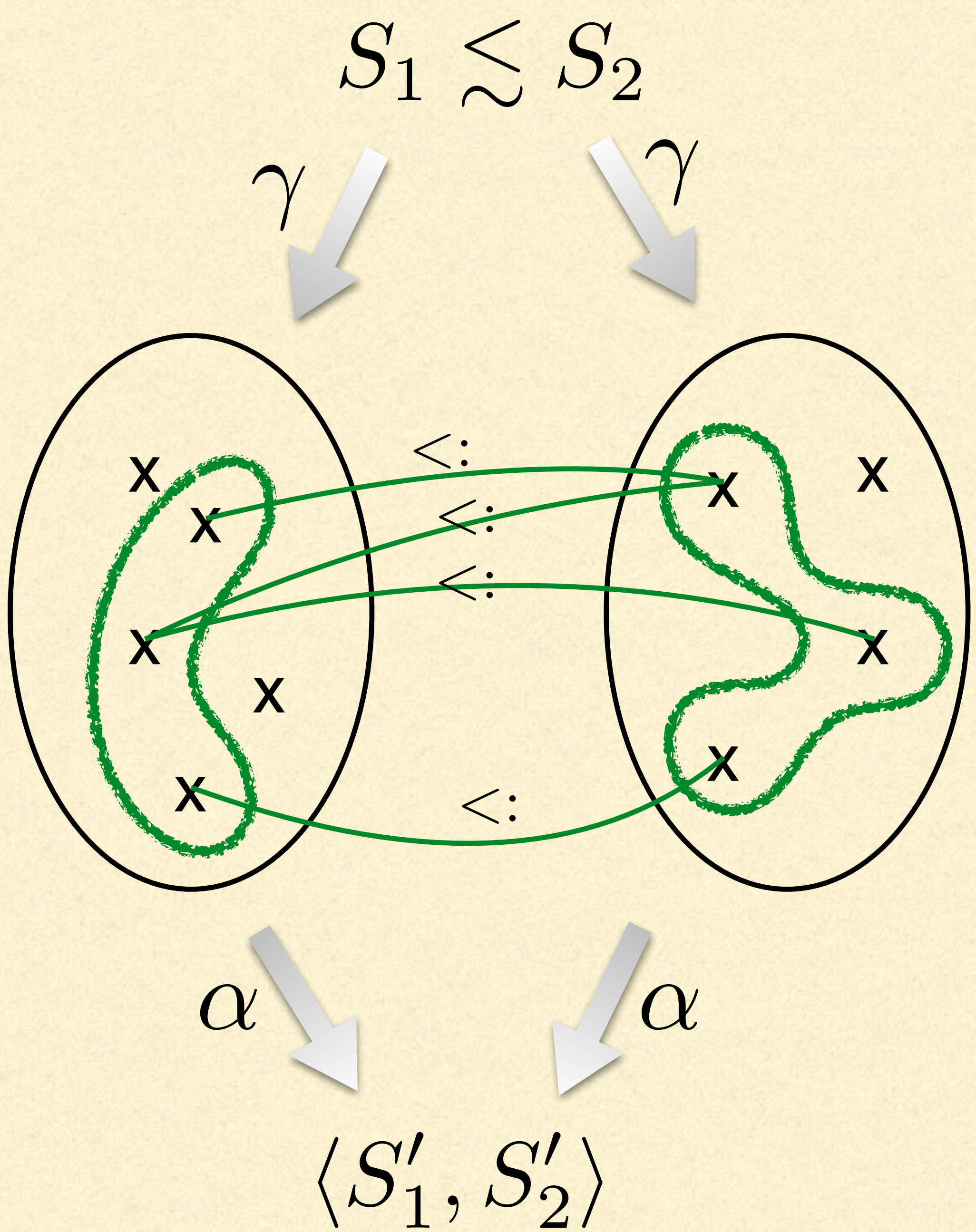
EVIDENCE (I.E., "CASTS")





$$\varepsilon_{12} \vdash U_1 \lesssim U_2$$

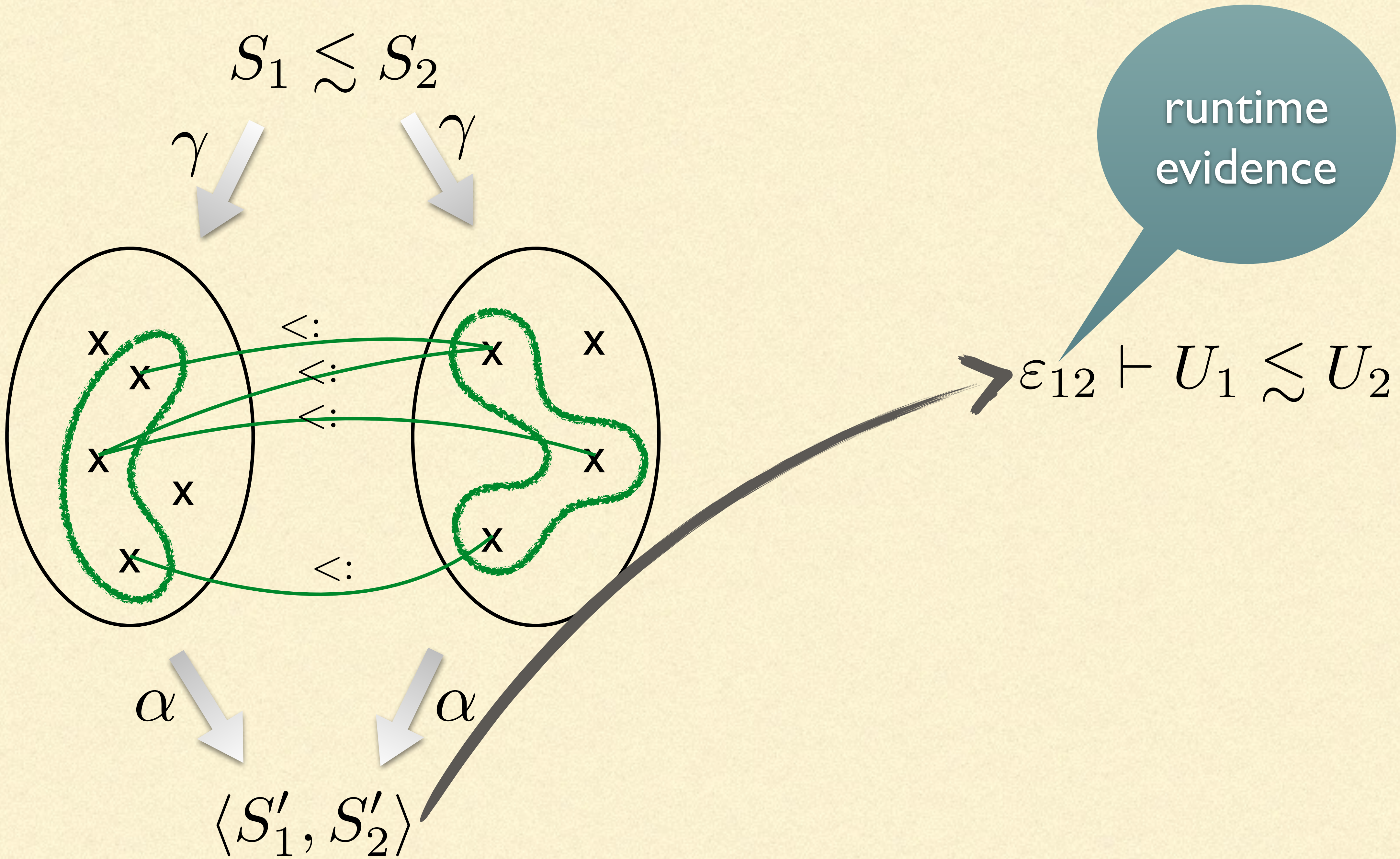
EVIDENCE (I.E., “CASTS”)



runtime evidence

$$\varepsilon_{12} \vdash U_1 \lesssim U_2$$

EVIDENCE (I.E., "CASTS")



COMPOSITION

$$\varepsilon_{23} \vdash S_2 \lesssim S_3 \qquad \varepsilon_{12} \vdash S_1 \lesssim S_2$$

$$\varepsilon_{12} \fatsemi \varepsilon_{23} \vdash S_1 \lesssim S_3$$

Runtime Proof
of Transitivity

COMPOSITION

$$\varepsilon_{23} \vdash S_2 \lesssim S_3 \qquad \varepsilon_{12} \vdash S_1 \lesssim S_2$$

$$\varepsilon_{12} \mathbin{;} \varepsilon_{23} \vdash S_1 \lesssim S_3$$

Runtime Proof of Transitivity

“cast error” if undefined

COMPOSITION

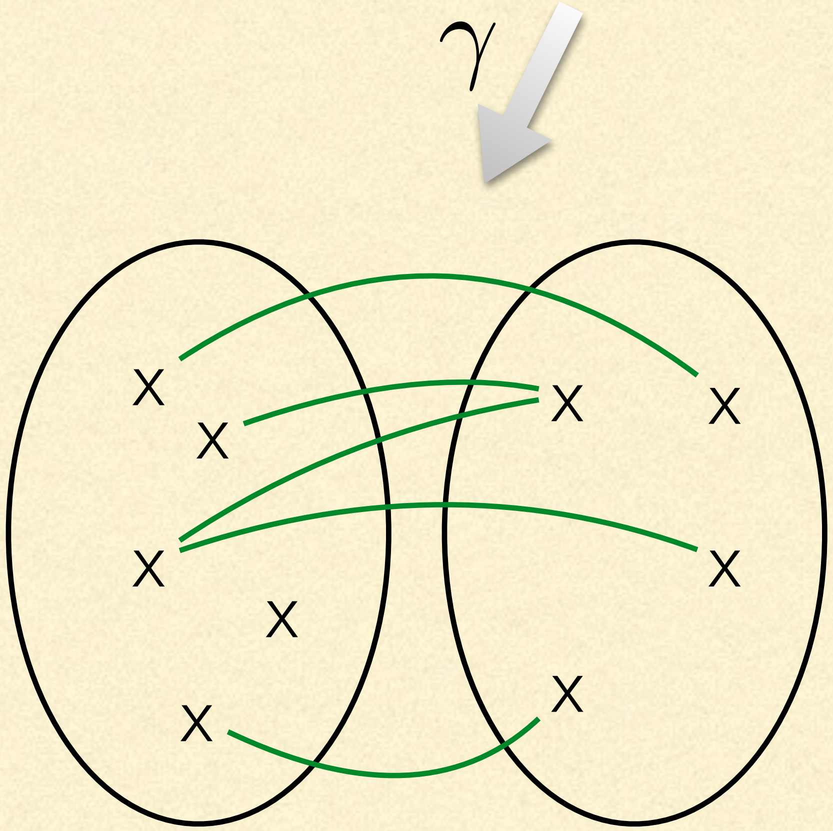
$$\varepsilon_{12} \fatsemi \varepsilon_{23}$$

COMPOSITION

$$\langle S_1, S_{21} \rangle \fatsemi \langle S_{22}, S_3 \rangle$$

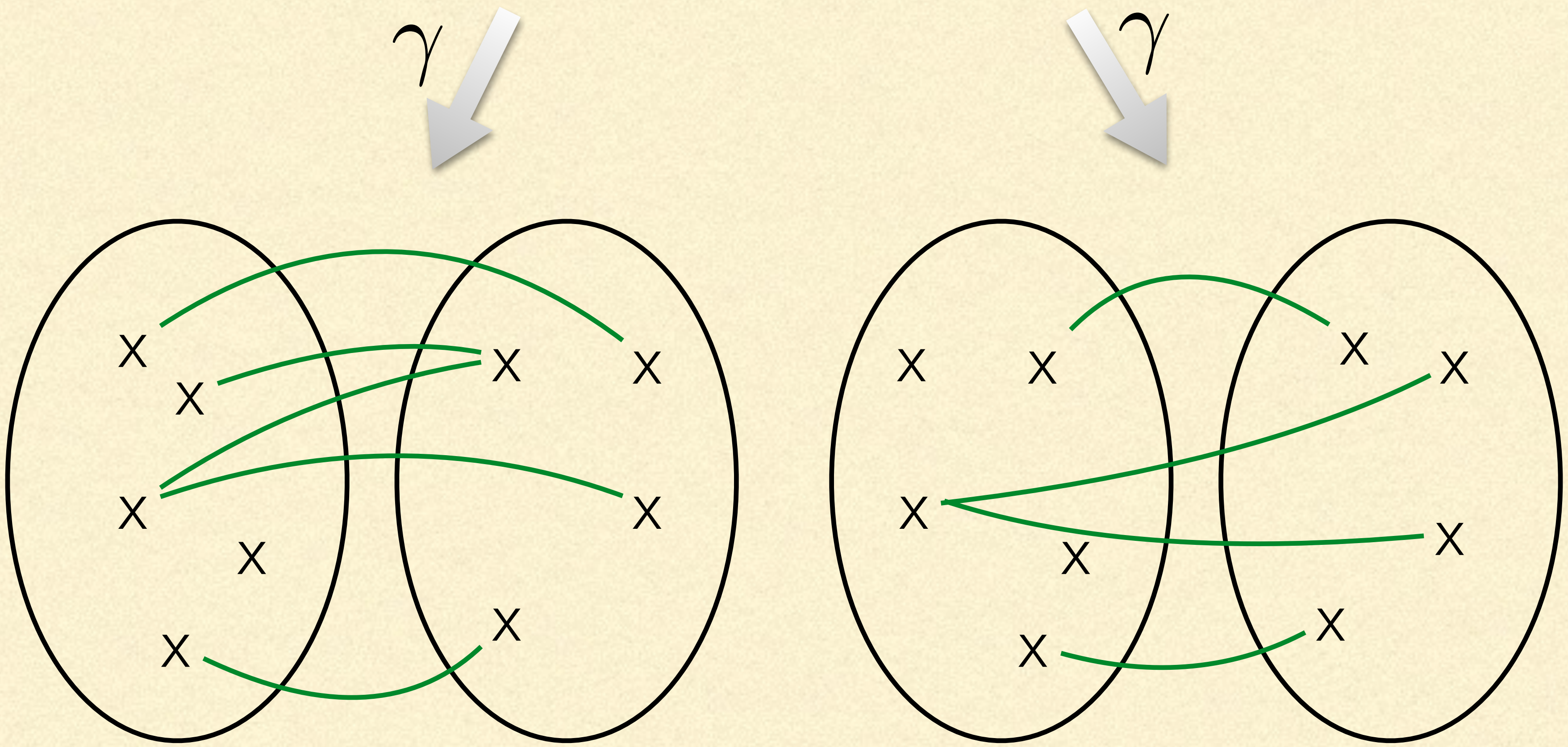
COMPOSITION

$$\langle S_1, S_{21} \rangle \fatsemi \langle S_{22}, S_3 \rangle$$



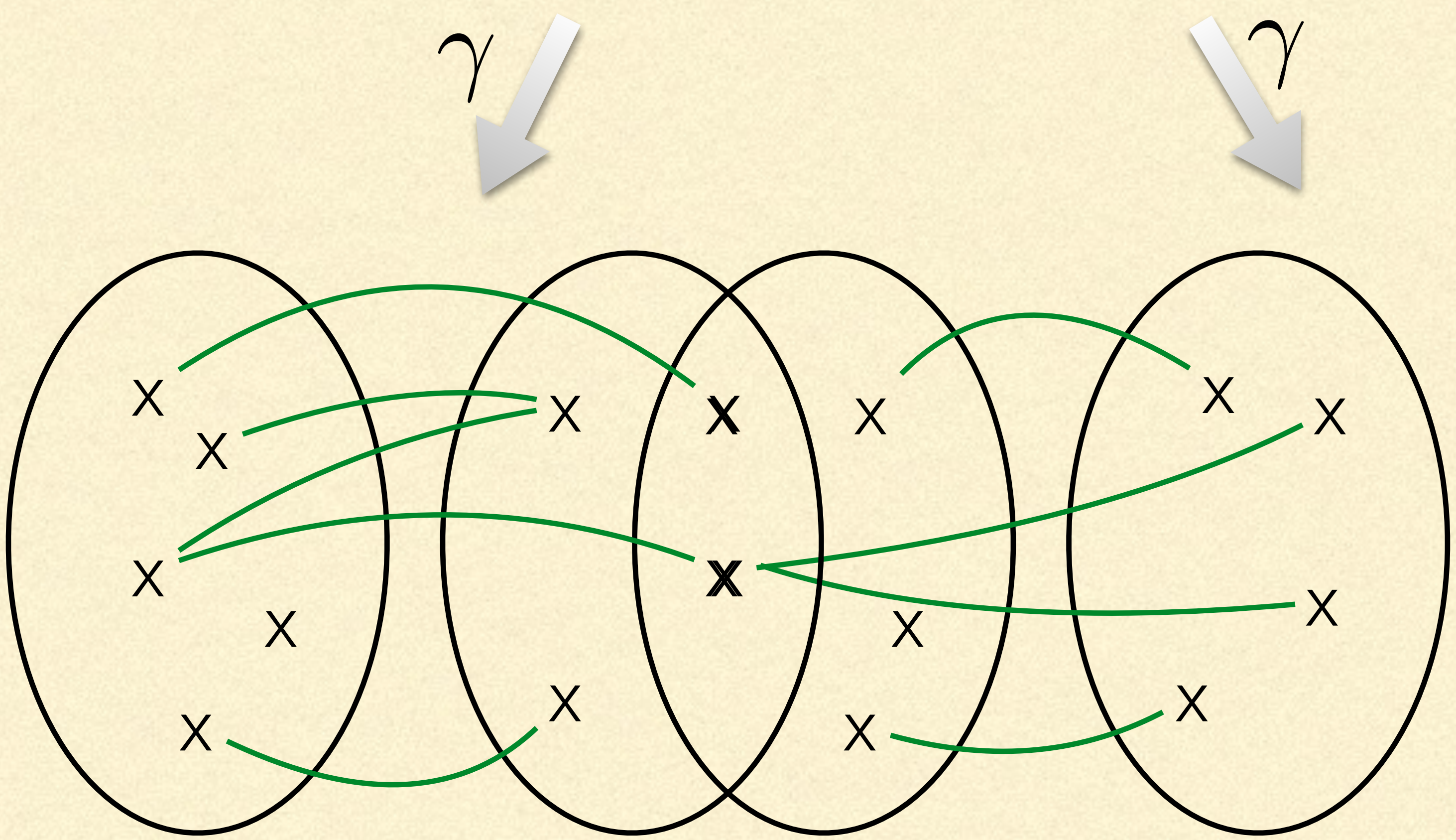
COMPOSITION

$$\langle S_1, S_{21} \rangle \fatsemi \langle S_{22}, S_3 \rangle$$



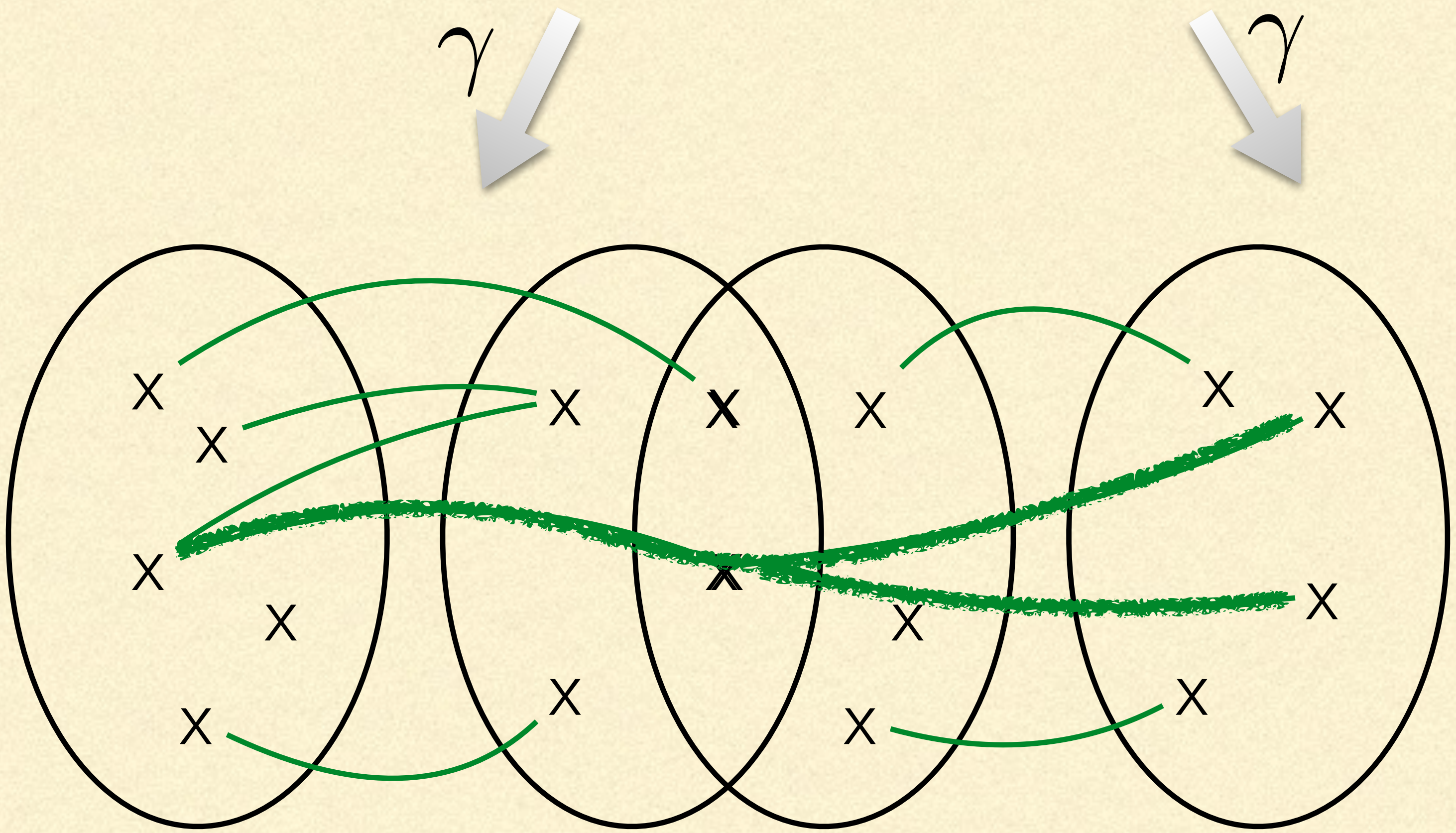
COMPOSITION

$$\langle S_1, S_{21} \rangle \fatsemi \langle S_{22}, S_3 \rangle$$



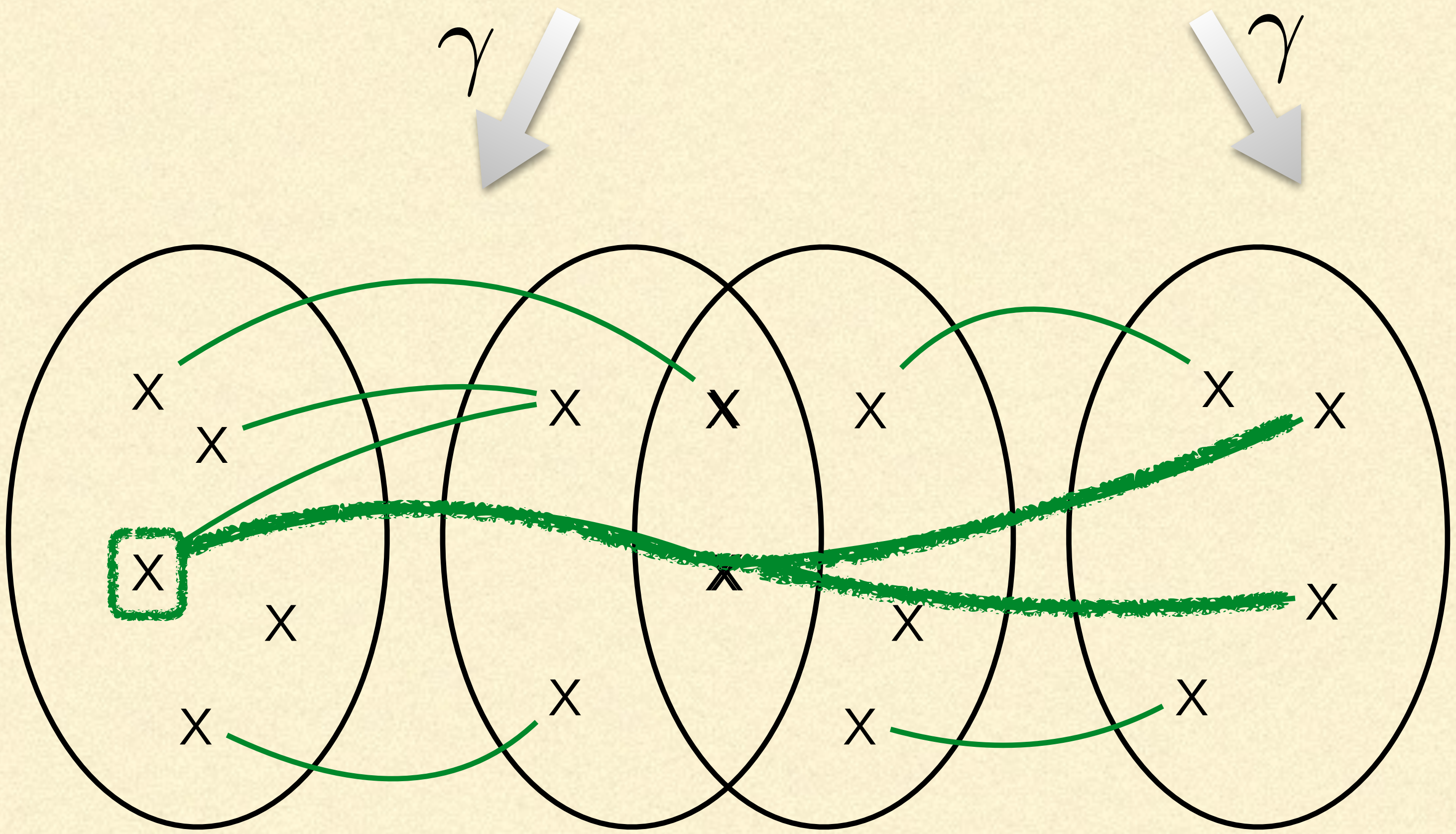
COMPOSITION

$$\langle S_1, S_{21} \rangle \fatsemi \langle S_{22}, S_3 \rangle$$



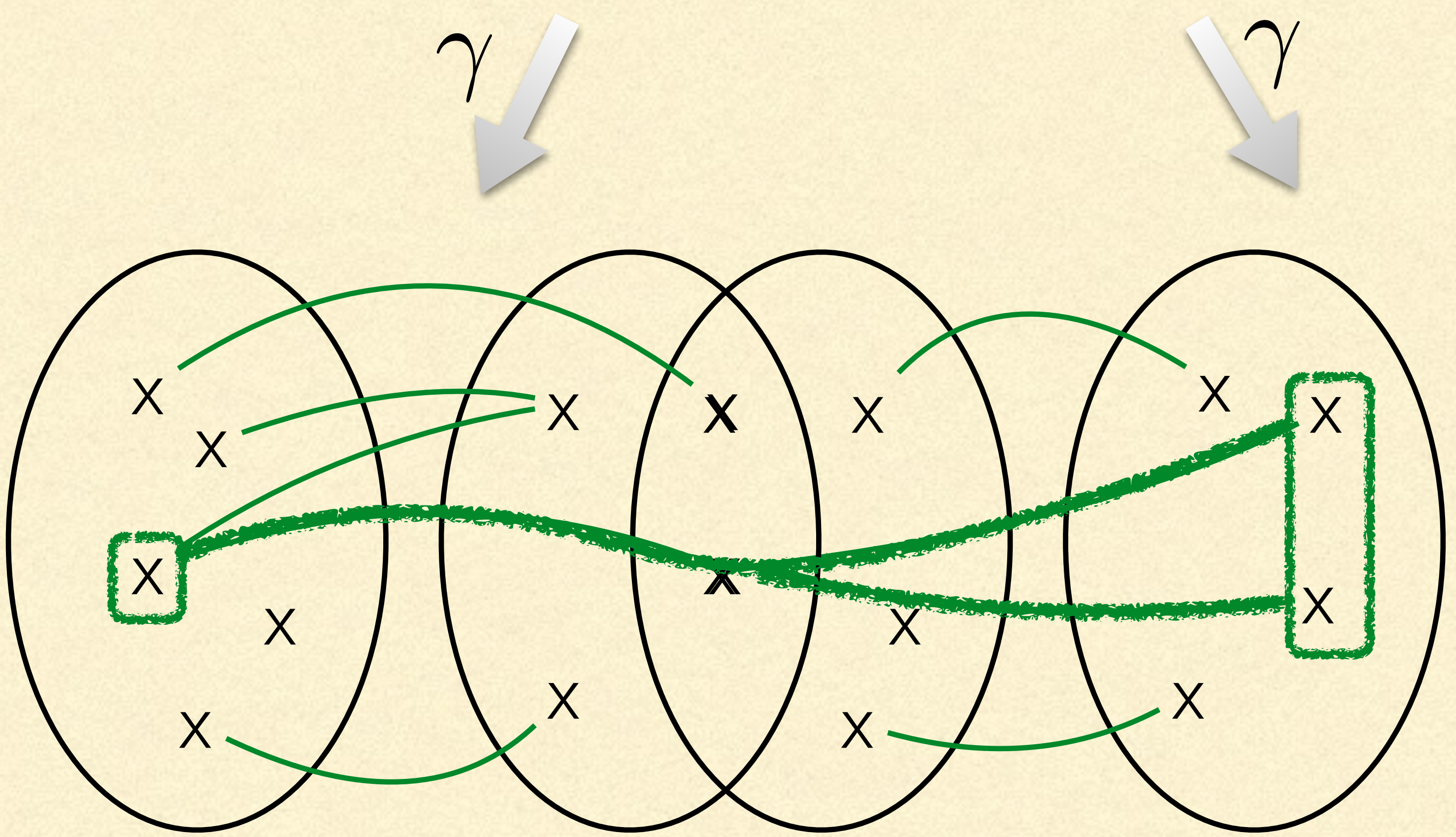
COMPOSITION

$$\langle S_1, S_{21} \rangle \fatsemi \langle S_{22}, S_3 \rangle$$



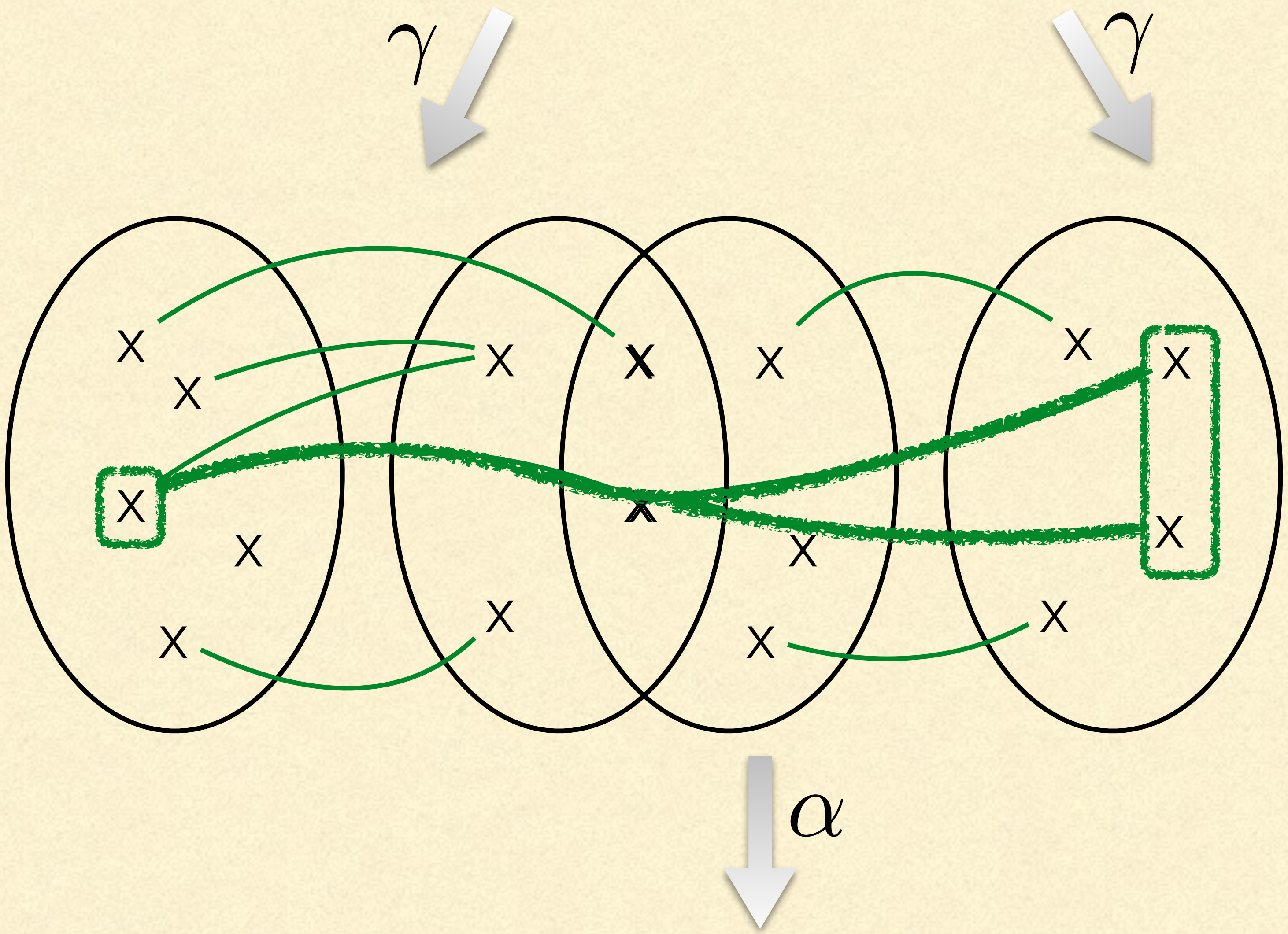
COMPOSITION

$$\langle S_1, S_{21}\rangle \fatsemi \langle S_{22}, S_3\rangle$$



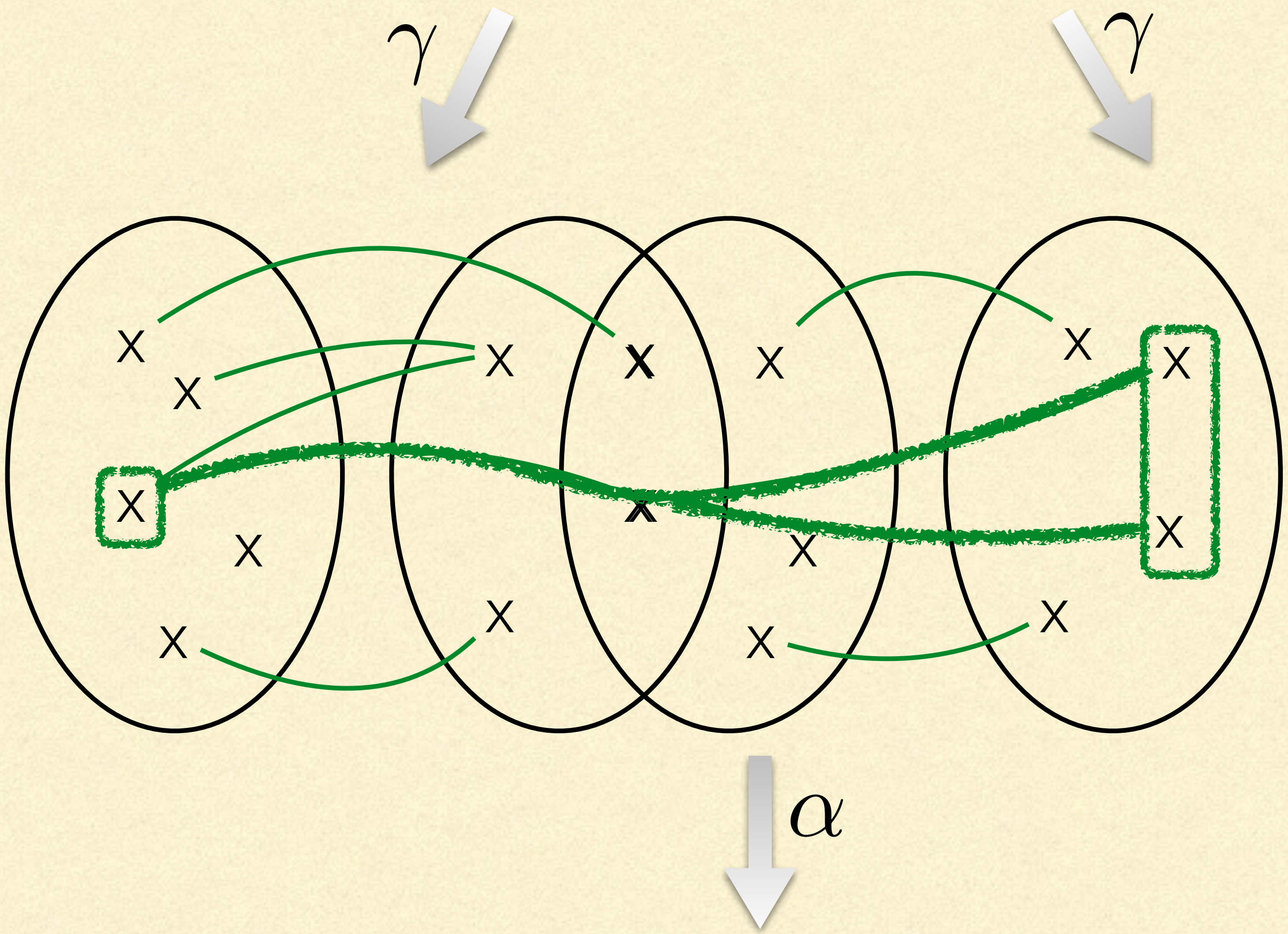
COMPOSITION

$$\langle S_1, S_{21} \rangle \,;\, \langle S_{22}, S_3 \rangle$$



COMPOSITION

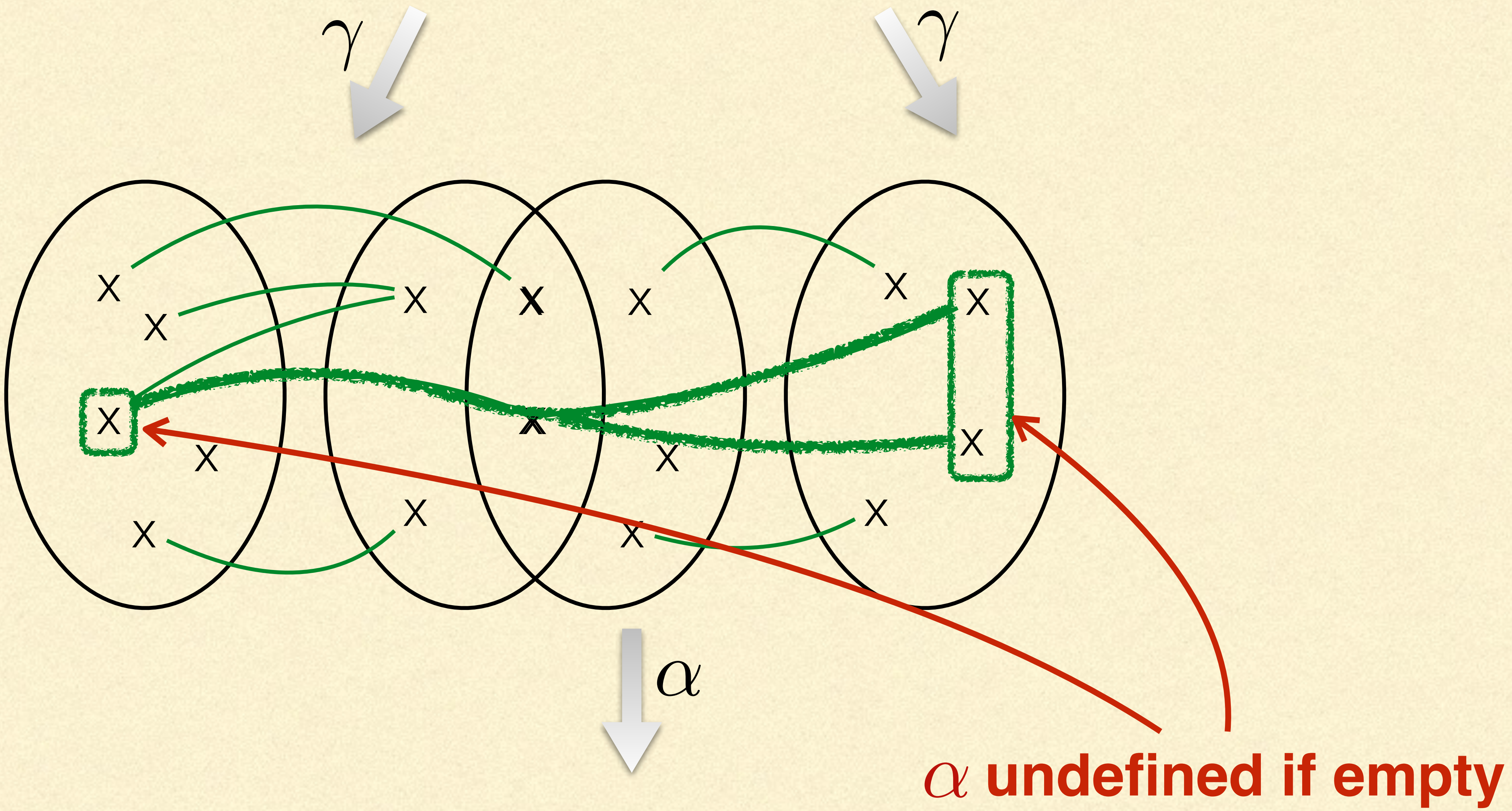
$$\langle S_1, S_{21} \rangle \fatsemi \langle S_{22}, S_3 \rangle$$



$$\langle \widetilde{T}_1', \widetilde{T}_3' \rangle$$

COMPOSITION

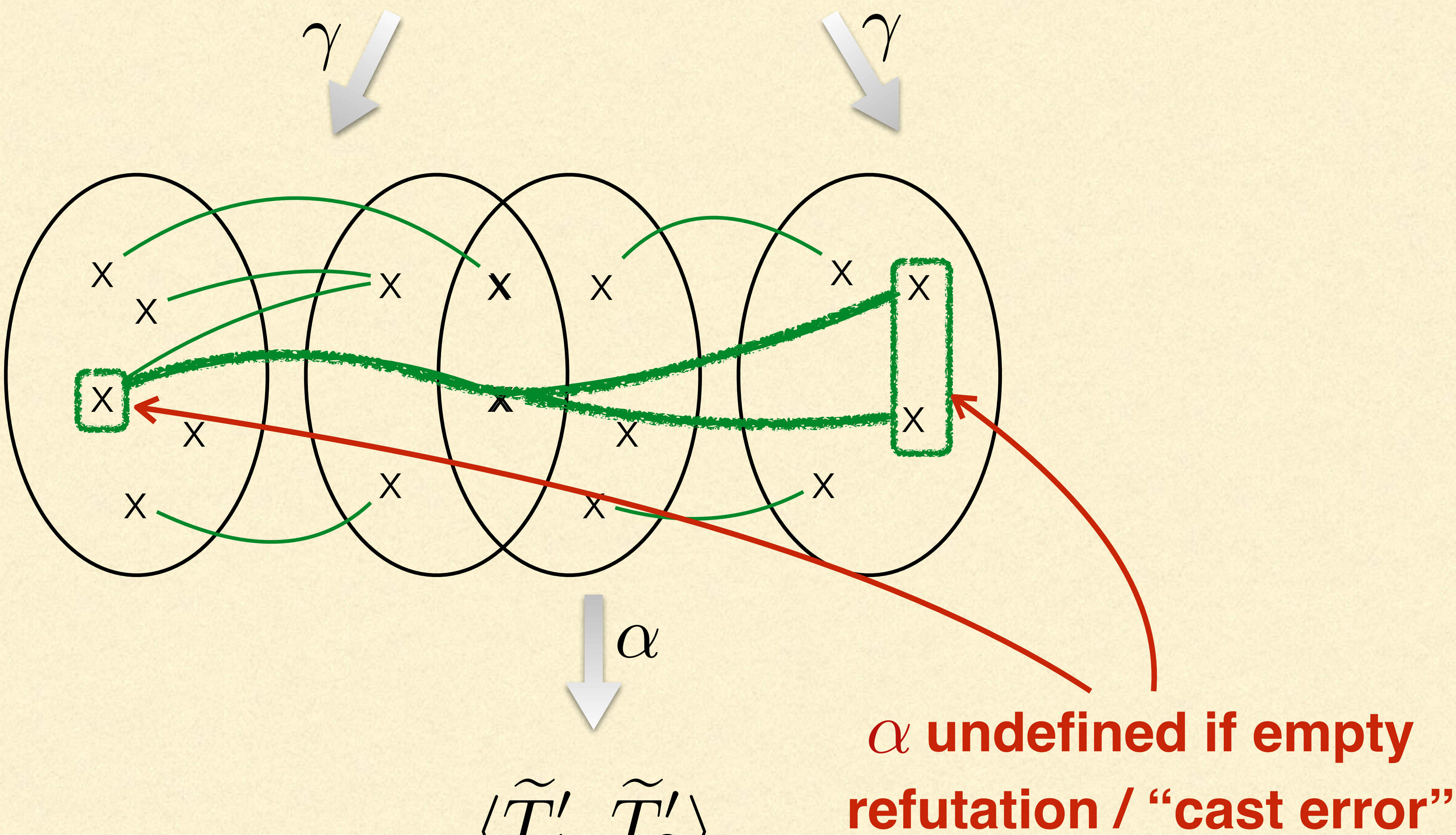
$$\langle S_1, S_{21}\rangle \fatsemi \langle S_{22}, S_3\rangle$$



$$\langle \widetilde{T}'_1, \widetilde{T}'_3\rangle$$

COMPOSITION

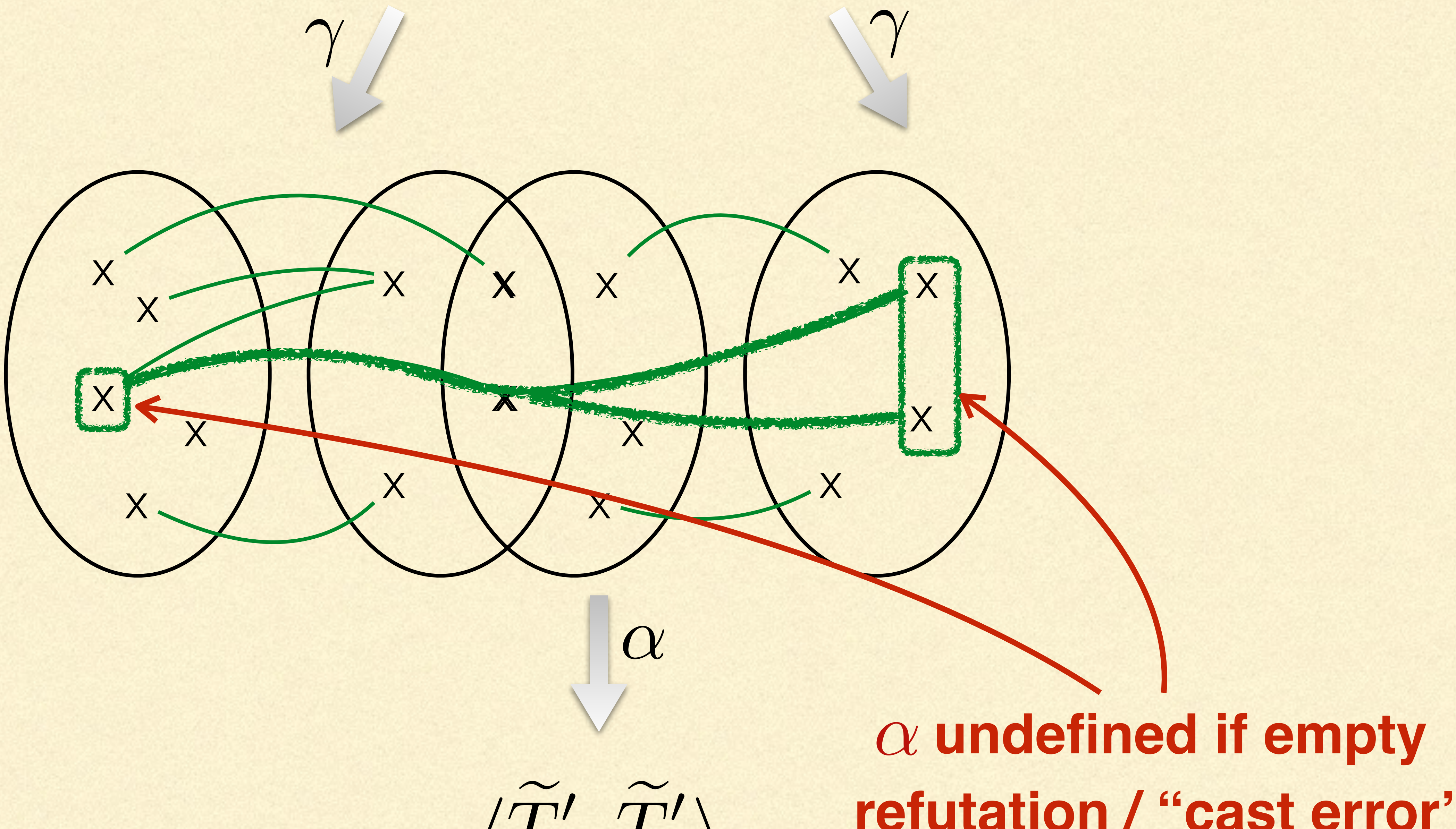
$$\langle S_1, S_{21} \rangle \fatsemi \langle S_{22}, S_3 \rangle$$



$$\langle \widetilde{T}'_1, \widetilde{T}'_3 \rangle$$

COMPOSITION

$$\langle S_1, S_{21} \rangle \fatsemi \langle S_{22}, S_3 \rangle$$



$$\langle \widetilde{T}'_1, \widetilde{T}'_3 \rangle$$

$$\alpha^2(\{\langle T_1, T_3 \rangle \in \gamma^2(\widetilde{T}_1, \widetilde{T}_3) \mid \exists T_2 \in \gamma(\widetilde{T}_{21}) \cap \gamma(\widetilde{T}_{22}).\, P(T_1, T_2) \wedge P(T_2, T_3)\})$$

SOME GOOD NEWS

$evenk_c$: $\texttt{Int} \rightarrow \texttt{Bool}$

$\texttt{Dyn}).$

$(\langle\texttt{Dyn}\rangle\ \texttt{true}))$

$1)\ (\langle\texttt{Bool} \rightarrow \texttt{Bool}\rangle\ k)$

$oddk_c$: $\texttt{Int} \rightarrow \texttt{Bool}$

$\rightarrow \texttt{Bool}).$

$-\ 1)\ (\langle\texttt{Dyn} \rightarrow \texttt{Dyn}\rangle\ k)$

Wrap on each call!

Herman et al. Space-Efficient Gradual Typing

SOME GOOD NEWS

$evenk_c : \texttt{Int} \rightarrow \texttt{Bool}$ … $\texttt{Dyn}).$

… $(\langle\texttt{Dyn}\rangle\ \texttt{true}))$

… $1)\ (\langle\texttt{Bool} \rightarrow \texttt{Bool}\rangle\ k)$

$oddk_c : \texttt{Int} \rightarrow \texttt{Bool}$ … $\rightarrow \texttt{Bool}).$

… $1)\ (\langle\texttt{Dyn} \rightarrow \texttt{Dyn}\rangle\ k)$

Just Compose!

Herman et al. Space-Efficient Gradual Typing

CONSERVATION OF GOODNESS

$$even : \texttt{Dyn} \rightarrow \texttt{Dyn} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Dyn}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{true}\ \ \textsf{else}\ odd\ (n-1)$$

$$odd : \texttt{Int} \rightarrow \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ even\ (n-1)$$

looks like a tail call!

Cast Insertion

$$odd_c : \texttt{Int} \rightarrow \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ \langle \texttt{Bool} \rangle\ (even\ (\langle \texttt{Dyn} \rangle\ (n-1)))$$

uh oh!

Herman et al. Space-Efficient Gradual Typing

CONSERVATION OF GOODNESS

$$even : \texttt{Dyn} \rightarrow \texttt{Dyn} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Dyn}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{true}\ \ \textsf{else}\ odd\ (n - 1)$$

$$odd : \texttt{Int} \rightarrow \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ even\ (n - 1)$$

looks like a tail call!

Cast Insertion

$$odd_c : \texttt{Int} \rightarrow \texttt{Bool} \stackrel{\text{def}}{=} \lambda n{:}\texttt{Int}.\ \textsf{if}\ (n = 0)\ \textsf{then}\ \texttt{false}\ \textsf{else}\ \langle\texttt{Bool}\rangle\ (even\ (\langle\texttt{Dyn}\rangle\ (n{-}1)))$$

compose in tail position eagerly!

Herman et al. Space-Efficient Gradual Typing

DON'T BE (TOO) EAGER!

$$\text{odd } 6 \rightarrow^*$$
$$\langle \text{Bool} \rangle\ \langle \text{Bool} \rangle\ \text{odd } 2 \rightarrow^*$$
$$\langle \text{Bool} \rangle\ \langle \text{Bool} \rangle\ \langle \text{Bool} \rangle\ \text{odd } 0 \rightarrow^*$$
$$\langle \text{Bool} \rangle\ \langle \text{Bool} \rangle\ \langle \text{Bool} \rangle\ \text{false} \rightarrow^*$$
$$\text{false}$$

Space-Inefficient computation

$\langle \text{Bool} \rangle \fatsemi (\langle \text{Bool} \rangle \fatsemi \langle \text{Bool} \rangle)$ “right-to-left” composition

DON'T BE (TOO) EAGER!

$\mathsf{odd}\ 4 \rightarrow^*$

$\langle \mathsf{Bool} \rangle\ \langle \mathsf{Bool} \rangle\ \mathsf{odd}\ 2 \rightarrow$

$\langle \mathsf{Bool} \rangle\ \mathsf{odd}\ 2 \rightarrow^*$

$\langle \mathsf{Bool} \rangle\ \langle \mathsf{Bool} \rangle\ \mathsf{odd}\ 0 \rightarrow$

$\langle \mathsf{Bool} \rangle\ \mathsf{odd}\ 0 \rightarrow^*$

$\langle \mathsf{Bool} \rangle\ \ \mathsf{false} \rightarrow$

false

Space-efficient computation

$(\langle \mathsf{Bool} \rangle \mathbin{;} \langle \mathsf{Bool} \rangle) \mathbin{;} \langle \mathsf{Bool} \rangle$

"left-to-right" composition

ASSOCIATIVITY!

Herman et al. works because:

$$\langle \mathsf{Bool} \rangle \fatsemi (\langle \mathsf{Bool} \rangle \fatsemi \langle \mathsf{Bool} \rangle) = (\langle \mathsf{Bool} \rangle \fatsemi \langle \mathsf{Bool} \rangle) \fatsemi \langle \mathsf{Bool} \rangle$$

ASSOCIATIVITY!

AGT can be space-efficient if:

$$\varepsilon_1 \fatsemi (\varepsilon_2 \fatsemi \varepsilon_3) = (\varepsilon_1 \fatsemi \varepsilon_2) \fatsemi \varepsilon_3$$

TRAGEDY!

$$\varepsilon_1 = \langle [x : \mathsf{Int}, ?], [x : \mathsf{Int}] \rangle$$
$$\varepsilon_2 = \langle [?], [?] \rangle$$
$$\varepsilon_3 = \langle [y : \mathsf{Bool}], [y : \mathsf{Bool}] \rangle$$

$$(\varepsilon_1 \mathbin{;} \varepsilon_2) \mathbin{;} \varepsilon_3 =$$
$$\langle [x : \mathsf{Int}, ?], [?] \rangle \mathbin{;} \varepsilon_3 =$$
$$\langle [x : \mathsf{Int}, y : \mathsf{Bool}, ?], [y : \mathsf{Bool}] \rangle$$

$$\varepsilon_1 \mathbin{;} (\varepsilon_2 \mathbin{;} \varepsilon_3) \simeq$$
$$\varepsilon_1 \mathbin{;} \langle [y : \mathsf{Bool}, ?], [y : \mathsf{Bool}] \rangle$$

undefined!

POST-NON-MORTEM

$$\varepsilon_1 = \langle [x : \mathsf{Int}, ?], [x : \mathsf{Int}] \rangle$$
$$\varepsilon_2 = \langle [?], [?] \rangle$$
$$\varepsilon_3 = \langle [y : \mathsf{Bool}], [y : \mathsf{Bool}] \rangle$$

$$(\varepsilon_1 \mathbin{⨾} \varepsilon_2) \mathbin{⨾} \varepsilon_3 =$$
$$\langle [x : \mathsf{Int}, ?], [?] \rangle \mathbin{⨾} \varepsilon_3 =$$
$$\langle [x : \mathsf{Int}, y : \mathsf{Bool}, ?], [y : \mathsf{Bool}] \rangle$$

POST-NON-MORTEM

$$\varepsilon_1 = \langle[x : \mathsf{Int}, ?], [x : \mathsf{Int}]\rangle$$

$$\varepsilon_2 = \langle[?], [?]\rangle$$

$$\varepsilon_3 = \langle[y : \mathsf{Bool}], [y : \mathsf{Bool}]\rangle$$

$$(\varepsilon_1 \mathbin{;} \varepsilon_2) \mathbin{;} \varepsilon_3 =$$

$$\langle[x : \mathsf{Int}, ?], [?]\rangle \mathbin{;} \varepsilon_3 =$$

$$\langle[x : \mathsf{Int}, y : \mathsf{Bool}, ?], [y : \mathsf{Bool}]\rangle$$

Lost Info:

$$\alpha(\{\ [x : \mathsf{Int}], [\ ]\ \}) = [?]$$

BOUNDED GRADUAL ROWS

Redesign our Evidence Pairs
(Surface Language Stays the Same)

$$A \quad ::= \quad R \mid O$$
$$S \quad ::= \quad \mathsf{Unit} \mid S \rightarrow S \mid ? \mid [\overline{\ell_A : S}\ \overline{\ell_O : \bot}] \mid [\overline{\ell_A : S}\ \overline{\ell_O : \bot}\ ?]$$

MECHANIZATION: NOT SO SPACE (OR TIME) EFFICIENT

- 14 GSM (Grad Student Months)
- 57K Lines of Coq

With some ad hoc extras

- "Beware of running it since it takes about 3 days and about 24 GB of RAM at one point. :)"

MECHANIZATION: NOT SO SPACE (OR TIME) EFFICIENT

- 14 GSM (Grad Student Months)
- 57K Lines of Coq

With some ad hoc extras

- "Beware of running it since it takes about 3 days and about 24 GB of RAM at one point. :)"

Felipe before

MECHANIZATION: NOT SO SPACE (OR TIME) EFFICIENT

- 14 GSM (Grad Student Months)
- 57K Lines of Coq

With some ad hoc extras

- "Beware of running it since it takes about 3 days and about 24 GB of RAM at one point. :)"

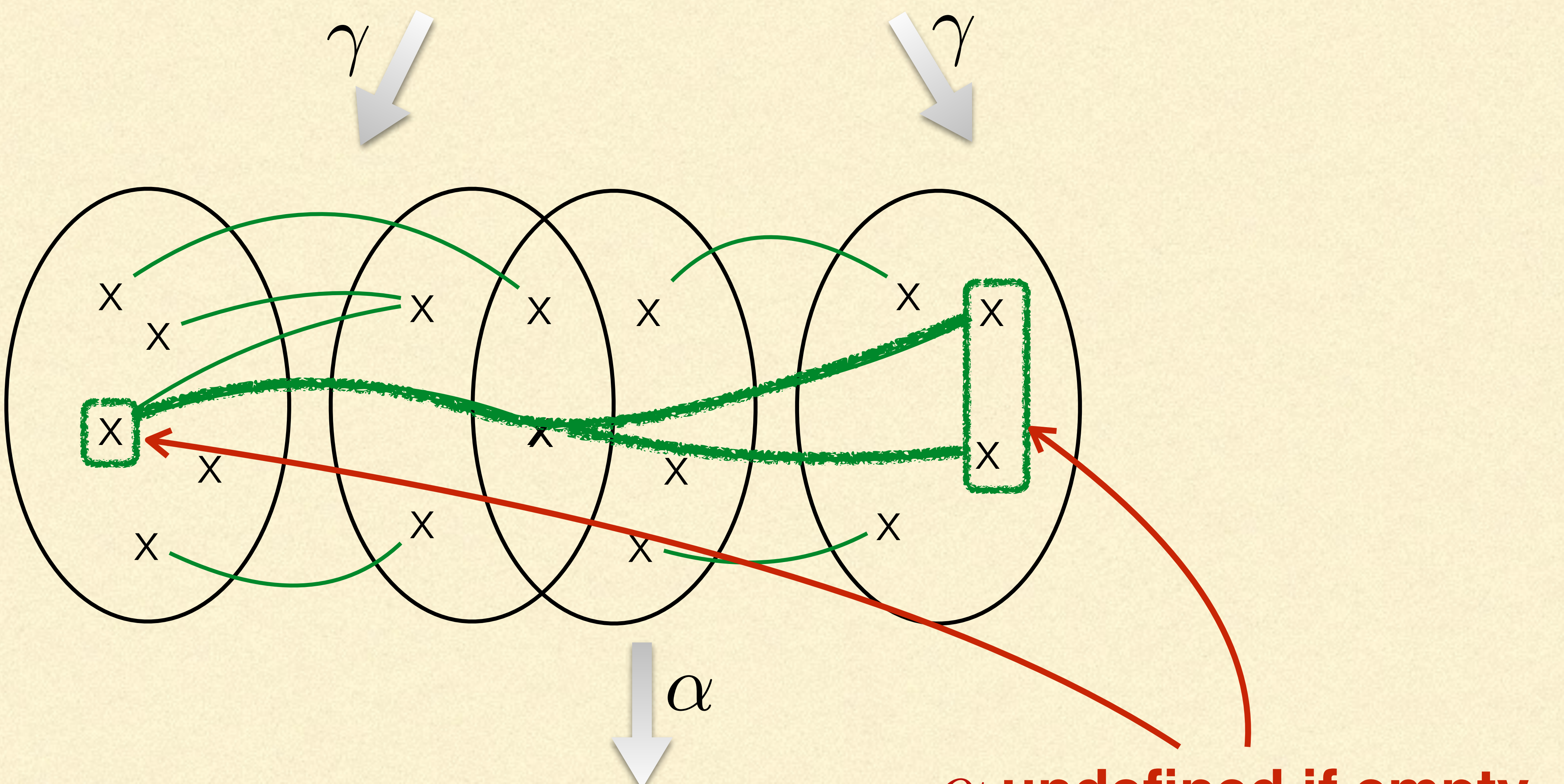
Felipe before

Felipe after

IS THERE ANOTHER WAY???

CONSISTENT TRANSITIVITY

$$\langle S_1, S_{21} \rangle \,;\, \langle S_{22}, S_3 \rangle$$



$$\langle \widetilde{T}'_1, \widetilde{T}'_3 \rangle$$

$$\alpha^2(\{\langle T_1, T_3 \rangle \in \gamma^2(\widetilde{T}_1, \widetilde{T}_3) \mid \exists T_2 \in \gamma(\widetilde{T}_{21}) \cap \gamma(\widetilde{T}_{22}).\, P(T_1, T_2) \wedge P(T_2, T_3)\})$$

ABSTRACT VIEW OF EVIDENCE

$$\gamma^{<:} : \lesssim \to \mathcal{P}^{+}(<:)$$

$$\gamma^{<:}(S_1, S_2) = \{ \langle T_1, T_2 \rangle \mid T_i \in \gamma(S_i) \text{ and } T_1 <: T_2 \} .$$

$$\alpha^{<:} : \mathcal{P}^{+}(<:) \to \lesssim$$

$$\alpha^{<:}(\mathcal{R}) = \langle \alpha(\pi_1(\mathcal{R})), \alpha(\pi_2(\mathcal{R})) \rangle .$$

Evidence for a consistent judgment is represented as a tuple of gradual types that characterize the space of possible static type relations.[h]

[h] Abstractions can lift to tuples other ways too (Cousot and Cousot 1994).

ABSTRACT VIEW OF EVIDENCE

$$\gamma^{<:} : \textsc{Evidence} \rightarrow \mathcal{P}^{+}(<:)$$
$$\gamma^{<:}(\varepsilon) = ???$$

$$\alpha^{<:} : \mathcal{P}^{+}(<:) \rightarrow \textsc{Evidence}$$
$$\alpha^{<:}(\mathcal{R}) = ???$$

A DIFFERENT APPROACH

Making Abstract Interpretations Complete

ROBERTO GIACOBAZZI

Università di Verona, Verona, Italy

FRANCESCO RANZATO

Università di Padova, Padova, Italy

AND

FRANCESCA SCOZZARI

École Polytechnique, Palaiseau, France

Two notions of completeness:

α-completeness

γ-completeness

A DIFFERENT APPROACH

Making Abstract Interpretations Complete…with respect to some operation

ROBERTO GIACOBAZZI

Università di Verona, Verona, Italy

FRANCESCO RANZATO

Università di Padova, Padova, Italy

AND

FRANCESCA SCOZZARI

École Polytechnique, Palaiseau, France

Two notions of completeness:

α-completeness

ɣ-completeness

A DIFFERENT APPROACH

Making Abstract Interpretations Complete…with respect to some operation

ROBERTO GIACOBAZZI

Università di Verona, Verona, Italy

FRANCESCO RANZATO

Università di Padova, Padova, Italy

AND

FRANCESCA SCOZZARI

École Polytechnique, Palaiseau, France

Two notions of completeness:

α-completeness

γ-completeness $\quad \gamma^{<:}(\varepsilon_1) \fatsemi^{\mathcal{R}} \gamma(\varepsilon_2) = \gamma^{<:}(\varepsilon_1 \fatsemi \varepsilon_2)$

A DIFFERENT APPROACH

Making Abstract Interpretations Complete…with respect to some operation

ROBERTO GIACOBAZZI

Università di Verona, Verona, Italy

FRANCESCO RANZATO

Università di Padova, Padova, Italy

AND

FRANCESCA SCOZZARI

École Polytechnique, Palaiseau, France

Two notions of completeness:

α-completeness

γ-completeness

$$\gamma^{<:}(\varepsilon_1) \fatsemi^{\mathcal{R}} \gamma(\varepsilon_2) = \gamma^{<:}(\varepsilon_1 \fatsemi \varepsilon_2)$$

Epsilon ASSOCIATES INC.

BONUS TRACKS

WTF GRADUAL TYPING???

The Dynamic Practice and Static Theory of Gradual Typing

Michael Greenberg
Pomona College, Claremont, CA, USA
http://www.cs.pomona.edu/~michael/
michael@cs.pomona.edu

Abstract

We can tease apart the research on gradual types into two 'lineages': a pragmatic, implementation-oriented dynamic-first lineage and a formal, type-theoretic, static-first lineage. The dynamic-first lineage's focus is on taming particular idioms—'pre-existing conditions' in untyped programming languages. The static-first lineage's focus is on interoperation and individual type system features, rather than the collection of features found in any particular language. Both appear in programming languages research under the name "gradual typing", and they are in active conversation with each other.

What are these two lineages? What challenges and opportunities await the static-first lineage? What progress has been made so far?

$\fatsemi : \mathrm{Ev} \times \mathrm{Ev} \rightharpoonup \mathrm{Ev}$ **Consistent Transitivity**

$$\langle ?, ? \rangle \fatsemi \langle ?, ? \rangle = \langle ?, ? \rangle$$

$$\langle S, S \rangle \fatsemi \langle ?, ? \rangle = \langle S, S \rangle \qquad \text{where } S \in \{ \mathsf{Int}, \mathsf{Bool} \}$$

$$\langle ?, ? \rangle \fatsemi \langle S, S \rangle = \langle S, S \rangle \qquad \text{where } S \in \{ \mathsf{Int}, \mathsf{Bool} \}$$

$$\langle S_{11} \to S_{12}, S_{21} \to S_{22} \rangle \fatsemi \langle ?, ? \rangle = \langle S_{11} \to S_{12}, S_{21} \to S_{22} \rangle \fatsemi \langle ? \to ?, ? \to ? \rangle$$

$$\langle ?, ? \rangle \fatsemi \langle S_{11} \to S_{12}, S_{21} \to S_{22} \rangle = \langle ? \to ?, ? \to ? \rangle \fatsemi \langle S_{11} \to S_{12}, S_{21} \to S_{22} \rangle$$

$$\langle ?, ? \rangle \fatsemi \langle [\overline{l_i : S_i}, *_1], [\overline{l_j : S_j}, *_2] \rangle = \langle [?], [?] \rangle \fatsemi \langle [\overline{l_i : S_i}, *_1], [\overline{l_j : S_j}, *_2] \rangle$$

$$\langle [\overline{l_i : S_i}, *_1], [\overline{l_j : S_j}, *_2] \rangle \fatsemi \langle ?, ? \rangle = \langle [\overline{l_i : S_i}, *_1], [\overline{l_j : S_j}, *_2] \rangle \fatsemi \langle [?], [?] \rangle$$

$$\langle S, S \rangle \fatsemi \langle S, S \rangle = \langle S, S \rangle \qquad \text{where } S \in \{ \mathsf{Int}, \mathsf{Bool} \}$$

$$\langle S_{11} \to S_{12}, S_{21} \to S_{22} \rangle \fatsemi \langle S_{31} \to S_{32}, S_{41} \to S_{42} \rangle = \langle S_{51} \to S_{52}, S_{61} \to S_{62} \rangle$$

$$\text{where } \langle S_{41}, S_{31} \rangle \fatsemi \langle S_{21}, S_{11} \rangle = \langle S_{61}, S_{51} \rangle, \quad \langle S_{12}, S_{22} \rangle \fatsemi \langle S_{32}, S_{42} \rangle = \langle S_{52}, S_{62} \rangle$$

$$\langle [\overline{l_i : S_{i1}}, *_1], [\overline{l_i : S_{i2}}, *_2] \rangle \fatsemi \langle [\overline{l_i : S_{i3}}, *_3], [\overline{l_i : S_{i4}}, *_4] \rangle = \langle [\overline{l_i : S_{i5}}, *_5], [\overline{l_i : S_{i6}}, *_6] \rangle$$

where $\overline{\langle S_{i1}, S_{i2} \rangle \fatsemi \langle S_{i3}, S_{i4} \rangle = \langle S_{i5}, S_{i6} \rangle}$,

$\langle *_1, *_2 \rangle$	$\langle *_3, *_4 \rangle$	$= \langle *_5, *_6 \rangle$
$\langle \emptyset, \emptyset \rangle$	$\langle *_3, *_4 \rangle$	$\langle \emptyset, \emptyset \rangle$
$\langle ?, ? \rangle$	$\langle ?, ? \rangle$	$\langle ?, ? \rangle$
else		$\langle ?, \emptyset \rangle$

$$\langle [\overline{l_i : S_{i1}}, \overline{l_j : S_{j1}}^{+}, *_1], [\overline{l_i : S_{i2}}, \overline{l_j : S_{j2}}^{+}, *_2] \rangle \fatsemi \langle [\overline{l_i : S_{i3}}, ?], [\overline{l_i : S_{i4}}, *_4] \rangle =$$
$$\langle [\overline{l_i : S_{i1}}, \overline{l_j : S_{j1}}^{+}, *_1], [\overline{l_i : S_{i2}}, \overline{l_j : S_{j2}}^{+}, *_2] \rangle \fatsemi \langle [\overline{l_i : S_{i3}}, \overline{l_j : ?}^{+}, ?], [\overline{l_i : S_{i4}}, *_4] \rangle =$$

$$\langle [\overline{l_i : S_{i1}}, ?], [\overline{l_i : S_{i2}}, ?] \rangle \fatsemi \langle [\overline{l_i : S_{i3}}, \overline{l_k : S_{k3}}^{+}, *_3], [\overline{l_i : S_{i4}}, \overline{l_k : S_{k4}}^{+}, *_4] \rangle =$$
$$\langle [\overline{l_i : S_{i1}}, \overline{l_k : ?}^{+}, ?], [\overline{l_i : S_{i2}}, \overline{l_k : ?}^{+}, ?] \rangle \fatsemi \langle [\overline{l_i : S_{i3}}, \overline{l_k : S_{k3}}^{+}, *_3], [\overline{l_i : S_{i4}}, \overline{l_k : S_{k4}}^{+}] *_4 \rangle$$

$$\langle [\overline{l_i : S_{i1}}, \overline{l_j : S_{j1}}^{+}, ?], [\overline{l_i : S_{i2}}, \overline{l_j : S_{j2}}^{+}, ?] \rangle \fatsemi \langle [\overline{l_i : S_{i3}}, \overline{l_k : S_{k3}}^{+}, ?], [\overline{l_i : S_{i4}}, \overline{l_k : S_{k4}}^{+}, *_4] \rangle =$$
$$\langle [\overline{l_i : S_{i1}}, \overline{l_j : S_{j1}}^{+}, \overline{l_k : ?}^{+}, ?], [\overline{l_i : S_{i2}}, \overline{l_j : S_{j2}}^{+}, \overline{l_k : ?}^{+}, ?] \rangle \fatsemi \langle [\overline{l_i : S_{i3}}, \overline{l_k : S_{k3}}, \overline{l_j : ?}^{+}, ?], [\overline{l_i : S_{i4}}, \overline{l_k : S_{k4}}, *_4] \rangle$$

Fig. 5. Consistent Transitivity: Part 1

$\boxed{\fatsemi : \mathrm{Ev} \times \mathrm{Ev} \rightharpoonup \mathrm{Ev}}$ **Consistent Transitivity (cont'd.)**

$$\langle [\overline{l_i : S_{i1}}, \overline{l_j : S_{j1}}^{\oplus_j}, \overline{l_k : S_k}^{\oplus_k}, *_1], [\overline{l_i : S_{i2}}, \overline{l_j : S_{j2}}^{\oplus_j}] \rangle \fatsemi \langle [\overline{l_i : S_{i3}}, \overline{l_j : S_{j3}}^{\oplus_j}], [\overline{l_i : S_{i4}}, *_4] \rangle =$$
$$\langle [\overline{l_i : S_{i5}}, \overline{l_j : S_{j5}}^{\oplus_j}, \overline{l_k : S_k}^{\oplus_k}, *_1], [\overline{l_i : S_{i6}}, *_4] \rangle$$

$$\langle [\overline{l_i : S_{i1}}, \overline{l_j : S_{j1}}^{\oplus_j}, \overline{l_q : S_{q1}}, \overline{l_k : S_k}^{\oplus_k}, *_1], [\overline{l_i : S_{i2}}, \overline{l_j : S_{j2}}^{\oplus_j}, \overline{l_q : S_{q2}}] \rangle \fatsemi \langle [\overline{l_i : S_{i3}}, \overline{l_j : S_{j3}}^{\oplus_j}, ?], [\overline{l_i : S_{i4}}, *_4] \rangle =$$
$$\langle [\overline{l_i : S_{i5}}, \overline{l_j : S_{j5}}^{\oplus_j}, \overline{l_q : S_{q1}}, \overline{l_k : S_k}^{\oplus_k}, *_1], [\overline{l_i : S_{i6}}, *_4] \rangle$$

$$\langle [\overline{l_i : S_{i1}}, \overline{l_m : S_{m1}}, \overline{l_j : S_{j1}}^{\oplus_j}, \overline{l_n : S_{n1}}^{\oplus_n}, \overline{l_k : S_k}^{\oplus_k}, *_1], [\overline{l_i : S_{i2}}, \overline{l_j : S_{j2}}^{\oplus_j}, ?] \rangle \fatsemi$$
$$\langle [\overline{l_i : S_{i3}}, \overline{l_m : S_{m3}}, \overline{l_p : S_{p3}}, \overline{l_j : S_{j3}}^{\oplus_j}, \overline{l_n : S_{n3}}^{\oplus_n}, \overline{l_r : S_{r3}}^{\oplus_r}, *_3], [\overline{l_i : S_{i4}}, \overline{l_m : S_{m4}}, \overline{l_p : S_{p4}}, *_4] \rangle =$$
$$\langle [\overline{l_i : S_{i5}}, \overline{l_m : S_{m5}}, \overline{l_p : S_{p5}}, \overline{l_j : S_{j5}}^{\oplus_j}, \overline{l_n : S_{n5}}^{\oplus_n}, \overline{l_r : S_{r5}}^{\oplus_r}, \overline{l_k : S_k}^{\oplus_k}, *_1], [\overline{l_i : S_{i6}}, \overline{l_m : S_{m6}}, \overline{l_p : S_{p6}}, *_4] \rangle$$

$$*_1 = ? \text{ if } \{ \overline{l_p}, \overline{l_r}^{\oplus_r} \} \neq \emptyset$$

$$\langle [\overline{l_i : S_{i1}}, \overline{l_m : S_{m1}}, \overline{l_j : S_{j1}}^{\oplus_j}, \overline{l_n : S_{n1}}^{\oplus_n}, \overline{l_q : S_{q1}}^{+}, \overline{l_k : S_k}^{\oplus_k}, *_1], [\overline{l_i : S_{i2}}, \overline{l_j : S_{j2}}^{\oplus_j}, \overline{l_q : S_{q2}}^{+}, ?] \rangle \fatsemi$$
$$\langle [\overline{l_i : S_{i3}}, \overline{l_m : S_{m3}}, \overline{l_p : S_{p3}}, \overline{l_j : S_{j3}}^{\oplus_j}, \overline{l_n : S_{n3}}^{\oplus_n}, \overline{l_r : S_{r3}}^{\oplus_r}, ?], [\overline{l_i : S_{i4}}, \overline{l_m : S_{m4}}, \overline{l_p : S_{p4}}, *_4] \rangle$$
$$=$$
$$\langle [\overline{l_i : S_{i1}}, \overline{l_m : S_{m1}}, \overline{l_j : S_{j1}}^{\oplus_j}, \overline{l_n : S_{n1}}^{\oplus_n}, \overline{l_q : S_{q1}}^{+}, \overline{l_k : S_k}^{\oplus_k}, *_1], [\overline{l_i : S_{i2}}, \overline{l_j : S_{j2}}^{\oplus_j}, \overline{l_q : S_{q2}}^{+}, ?] \rangle \fatsemi$$
$$\langle [\overline{l_i : S_{i3}}, \overline{l_m : S_{m3}}, \overline{l_p : S_{p3}}, \overline{l_j : S_{j3}}^{\oplus_j}, \overline{l_n : S_{n3}}^{\oplus_n}, \overline{l_r : S_{r3}}^{\oplus_r}, \overline{l_q : ?}^{+}, ?], [\overline{l_i : S_{i4}}, \overline{l_m : S_{m4}}, \overline{l_p : S_{p4}}, *_4] \rangle$$

where $\overline{\langle S_{i1}, S_{i2} \rangle \fatsemi \langle S_{i3}, S_{i4} \rangle = \langle S_{i5}, S_{i6} \rangle}$, $\quad \overline{\langle S_{j1}, S_{j2} \rangle \fatsemi \langle S_{j3}, S_{j3} \rangle = \langle S_{j5}, S_{j6} \rangle}$

$\overline{\langle S_{m1}, ? \rangle \fatsemi \langle S_{m3}, S_{m4} \rangle = \langle S_{m5}, S_{m6} \rangle}$, $\quad \overline{\langle S_{n1}, ? \rangle \fatsemi \langle S_{n3}, S_{n3} \rangle = \langle S_{n5}, S_{n6} \rangle}$

$\overline{\langle ?, ? \rangle \fatsemi \langle S_{p3}, S_{p4} \rangle = \langle S_{p5}, S_{p6} \rangle}$, $\quad \overline{\langle ?, ? \rangle \fatsemi \langle S_{r3}, S_{r3} \rangle = \langle S_{r5}, S_{r6} \rangle}$

$$\langle \oplus_{j+n+r}, \oplus_k \rangle \in \{ \langle \emptyset, + \rangle, \langle +, \emptyset \rangle, \langle +, + \rangle \}, \qquad \oplus_{j+n+r} = \begin{cases} \oplus_j & \oplus_n, \oplus_r \text{ do not appear} \\ \emptyset & \langle \oplus_j, \oplus_n, \oplus_r \rangle = \langle \emptyset, \emptyset, \emptyset \rangle \\ + & \text{otherwise} \end{cases}$$

$\langle S_1, S_2 \rangle \fatsemi \langle S_3, S_4 \rangle$ undefined otherwise

Fig. 6. Consistent Transitivity: Part 2