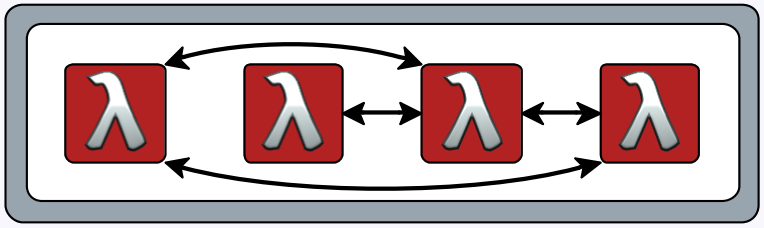
The Behavior of Gradual Types: A User Study

Preston Tunnell Wilson[1]
Ben Greenman[2] ✭
Justin Pombrio[1]
Shriram Krishnamurthi[1]

Intuition: Gradual Typing

Intuition: Gradual Typing

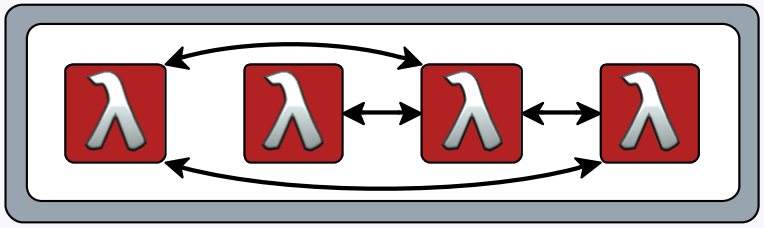
Dynamic Typing



un(i)typed components
value safe

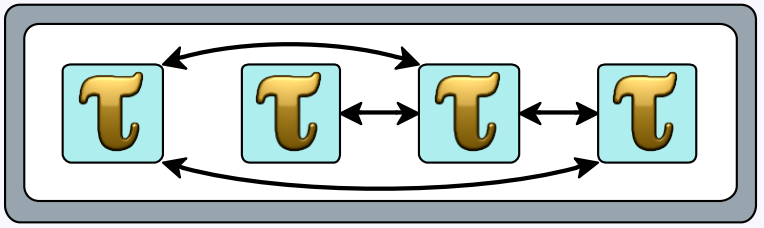
Intuition: Gradual Typing

Dynamic Typing



un(i)typed components
value safe

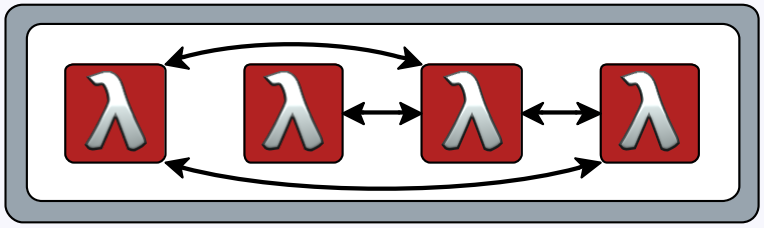
Static Typing



typed components
type safe

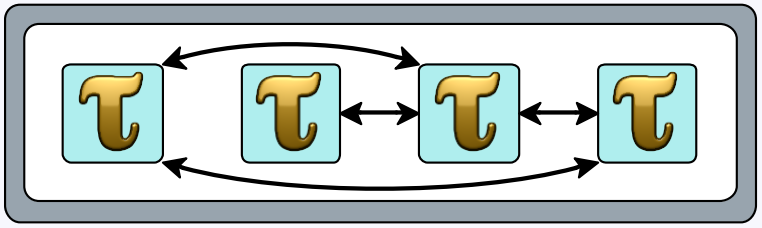
Intuition: Gradual Typing

Dynamic Typing



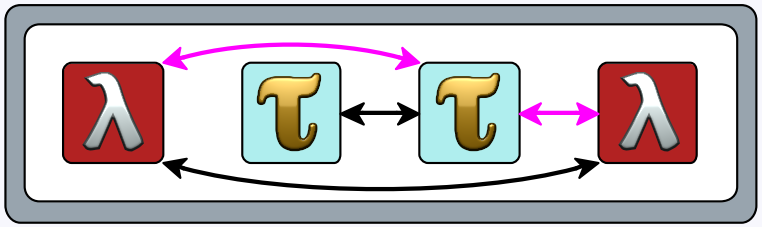
un(i)typed components
value safe

Static Typing



typed components
type safe

Gradual Typing



typed + untyped components
.... safe?

Gradual Typing Helps Programmers?

DLS'14

Combining static and dynamic typing within the same language **offers clear benefits** to programmers

Gradual Typing Helps Programmers?

ESOP'12

We conjecture that a **programmer would like** the guarantee that the values produced by their components are never used in violation to the interface specifications ...

Gradual Typing Helps Programmers?

POPL'17

... [run-time checks] inspect the top-level type (or type-tag) of each value, ensuring safe interaction and providing **the expected type safety** to programmers

Gradual Typing Helps Programmers?

A programmer **may favour** unsound monitoring over wrappers that change the semantics of their program.

ECOOP'17

Gradual Typing Helps Programmers?

Being sound, Safe TypeScript endows types with many of the properties that Java or C# programmers **might expect** but not find in TypeScript

The system **lives up to all expectations** that developers have of sound language implementations.

... programmers should be able to add or remove type annotations without any **unexpected** impacts on their program

POPL'15

SNAPL'17

SNAPL'15

POPL'08

OOPSLA'17

ECOOP'14

Data to Support Claims?

`DLS'14`

`ESOP'12`

`POPL'17`

`ECOOP'17`

`POPL'15`

`SNAPL'17`

`SNAPL'15`

`POPL'08`

`OOPSLA'17`

`ECOOP'14`

Data to Support Claims?

DLS'14	✗
ESOP'12	✗
POPL'17	✗
ECOOP'17	✗
POPL'15	✗
SNAPL'17	✗
SNAPL'15	✗
POPL'08	✗
OOPSLA'17	✗
ECOOP'14	✗

Three Different Approaches!

Three Different Approaches!

Deep

types are sound/enforced

Shallow

Erasure

Three Different Approaches!

DLS'14

ESOP'12

POPL'17

ECOOP'17

POPL'15

SNAPL'17

SNAPL'15

POPL'08

OOPSLA'17

ECOOP'14

Deep

types are sound/enforced

Shallow

typed code cannot
get stuck

Erasure

Three Different Approaches!

DLS'14

ESOP'12

POPL'17

ECOOP'17

POPL'15

SNAPL'17

SNAPL'15

POPL'08

OOPSLA'17

ECOOP'14

Deep

types are sound/enforced

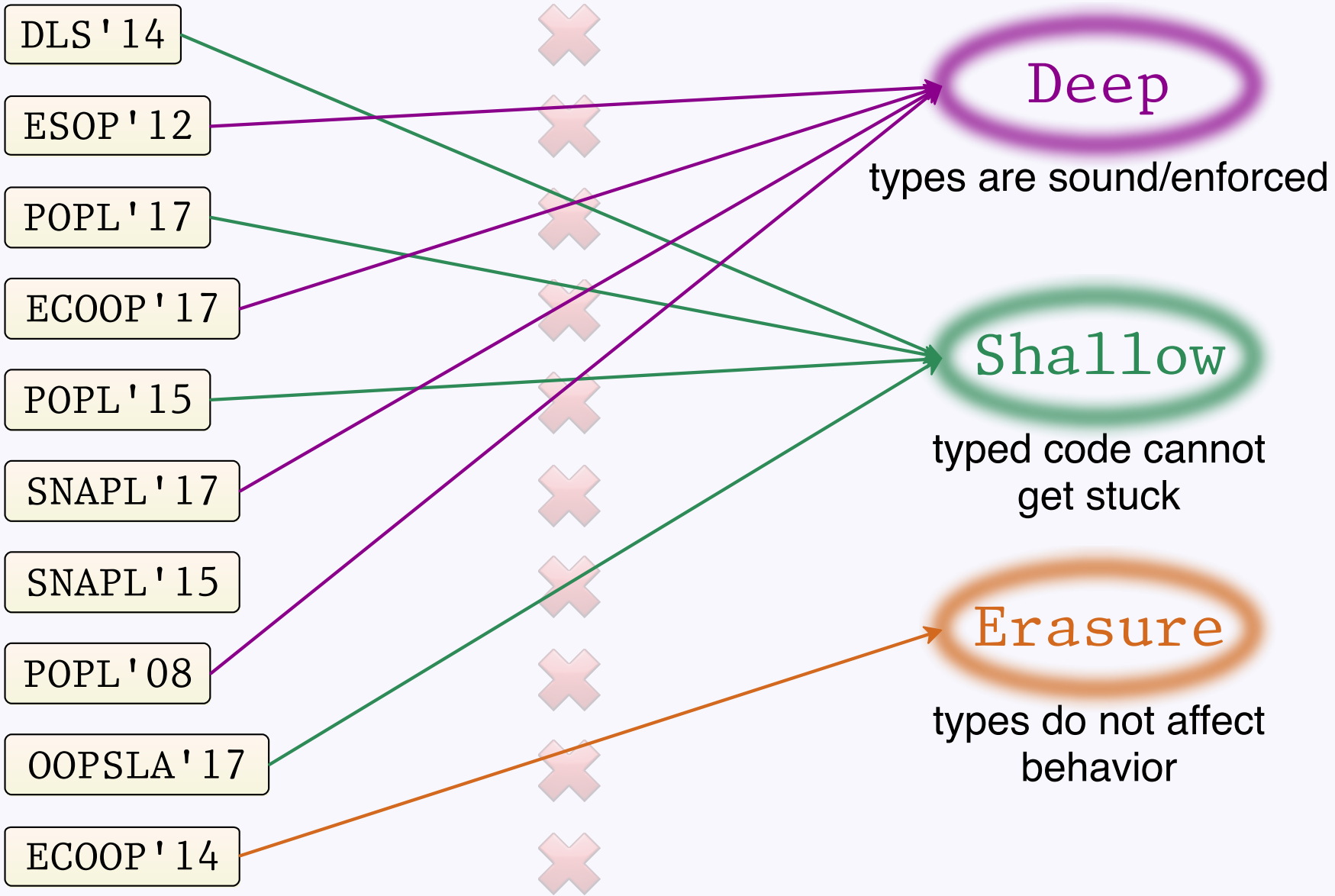
Shallow

typed code cannot get stuck

Erasure

types do not affect behavior

Three Different Approaches!



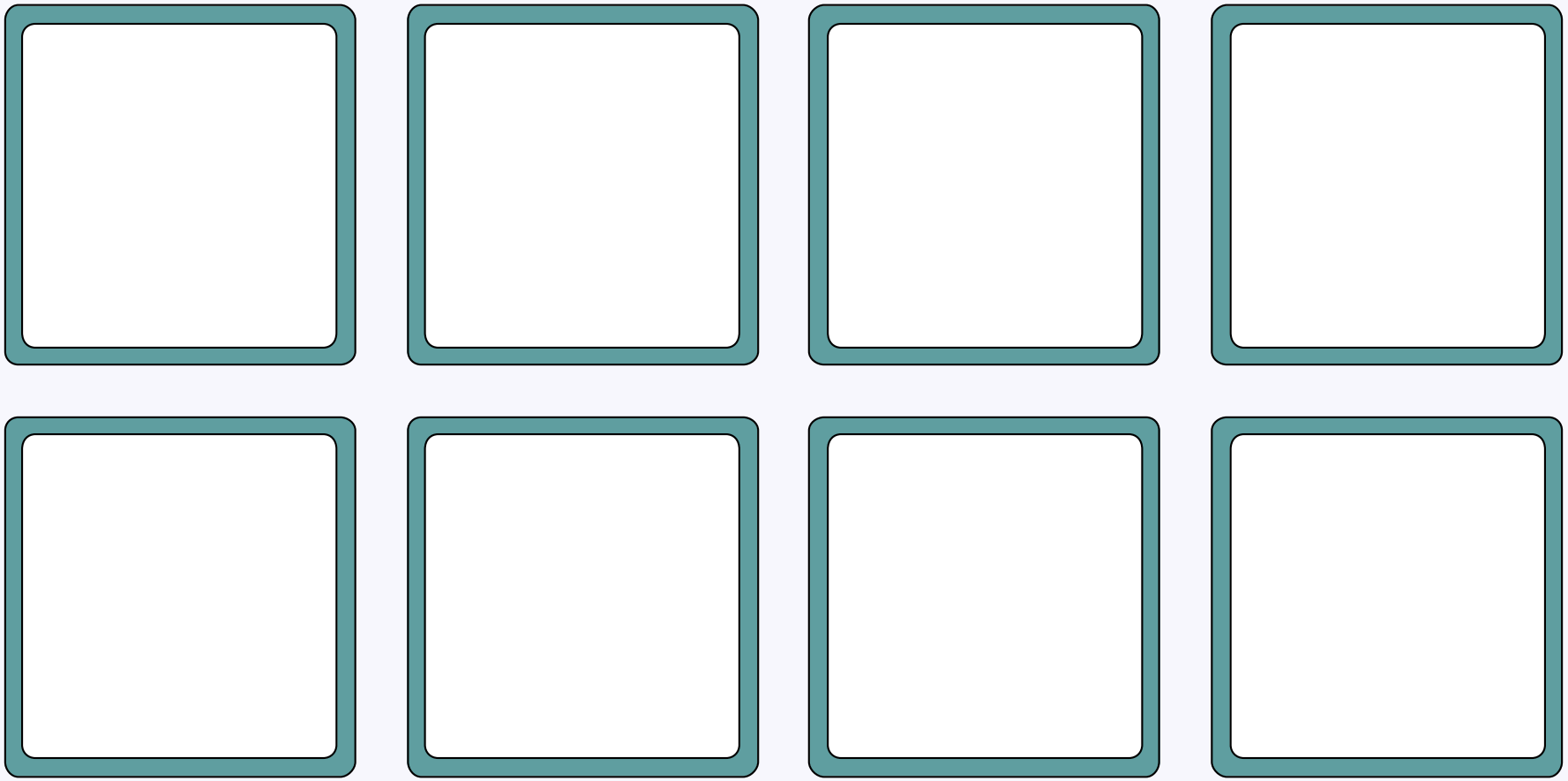
In this paper:

We begin to **address the lack of data** with a **developer survey** contrasting the different approaches to gradual typing

Deep vs. Shallow

Erasure

Survey based on 8 example programs



Survey Prompt

We are designing a language that mixes typed and untyped code.

Survey Prompt

We are designing a language that mixes typed and untyped code.

We want your opinion on what should happen when untyped values flow into typed expressions.

Survey Prompt

We are designing a language that mixes typed and untyped code.

We want your opinion on what should happen when untyped values flow into typed expressions.

Our language has static type checking but does not have type inference ... The following [programs] pass the static type checker.

Survey Prompt

We are designing a language that mixes typed and untyped code.

We want your opinion on what should happen when untyped values flow into typed expressions.

Our language has static type checking but does not have type inference ... The following [programs] pass the static type checker.

We are not looking for feedback on syntax.

Survey Prompt

We are designing a language that mixes typed and untyped code.

We want your opinion on what should happen when untyped values flow into typed expressions.

Our language has static type checking but does not have type inference ... The following [programs] pass the static type checker.

We are not looking for feedback on syntax.

Question 7

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

Question 7

Type annotations

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

Question 7

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

Array operations

Question 7

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

Question 7

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

Question 7

Multiple behaviors (unlabeled)

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

Error: line 4 expected `String` got `42`

Error: line 5 expected `Number` got `"bye"`

`"bye"`

Question 7

Multiple behaviors (unlabeled)

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

Error: line 4 expected `String` got 42

Error: line 5 expected `Number` got "bye"

"bye"

(Deep)
(Shallow)
(Erasure)

Question 7

	Like	Dislike
Expected		
Unexpected		

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

Error: line 4 expected `String` got `42`

Error: line 5 expected `Number` got `"bye"`

`"bye"`

Question 7

	Like	Dislike
Expected	LE	DE
Unexpected	LU	DU

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

Error: line 4 expected `String` got `42`

Error: line 5 expected `Number` got `"bye"`

`"bye"`

Question 7

	Like	Dislike
Expected	LE	DE
Unexpected	LU	DU

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

Error: line 4 expected `String` got `42`

Error: line 5 expected `Number` got `"bye"`

`"bye"`

LE	LU	DE	DU
○	○	○	○
○	○	○	○
○	○	○	○

Question 7

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

	LE	LU	DE	DU
Error: line 4 expected `String` got 42	○	○	○	○
Error: line 5 expected `Number` got `"bye"`	○	○	○	○
`"bye"`	○	○	○	○

Question 5

Two distinct behaviors

```
1 var obj0 = {
    k = 0;
    add = function(i : Number) { k = i }};
2 var t = "hello";
3 obj0.add(t);
4 var k : String = obj0.k;
5 k
```

Error: line 1 expected Number got "hello"

```
"hello"
```

Question 5

Two distinct behaviors

```
1 var obj0 = {
    k = 0;
    add = function(i : Number) { k = i }};
2 var t = "hello";
3 obj0.add(t);
4 var k : String = obj0.k;
5 k
```

Error: line 1 expected Number got "hello"

```
"hello"
```

(Deep,Shallow)
(Erasure)

Question 5

```
1 var obj0 = {
    k = 0;
    add = function(i : Number) { k = i }};
2 var t = "hello";
3 obj0.add(t);
4 var k : String = obj0.k;
5 k
```

	LE	LU	DE	DU
Error: line 1 expected Number got "hello"	○	○	○	○
"hello"	○	○	○	○

Question 5

Objects instead of closures, to avoid confusion

```
1 var obj0 = {
    k = 0;
    add = function(i : Number) { k = i }};
2 var t = "hello";
3 obj0.add(t);
4 var k : String = obj0.k;
5 k
```

	LE	LU	DE	DU
Error: line 1 expected Number got "hello"	○	○	○	○
"hello"	○	○	○	○

Followup Question

Agree/Disagree: Type annotations should not change the behavior of a program.

Summary: Survey Design

Solicit opinions on the semantics of a "new" language

Survey Prompt

We are designing a language that mixes typed and untyped code.
....

Gather `Like` × `Expect` preference on eight small programs designed to classify approaches

Question N

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

	LE	LU	DE	DU
Error: line 4 expected `String` got `42`	○	○	○	○
Error: line 5 expected `Number` got `"bye"`	○	○	○	○
"bye"	○	○	○	○

Explicit followup about optional typing

Followup Question

Agree/Disagree: Type annotations should not change the behavior of a program.

Summary: Survey Design

Solicit opinions on the semantics of a "new" language

Survey Prompt

> We are designing a language that mixes typed and untyped code.
> ....

Gather `Like` × `Expect` preference on eight small programs designed to classify approaches

Why 8?

Question N

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

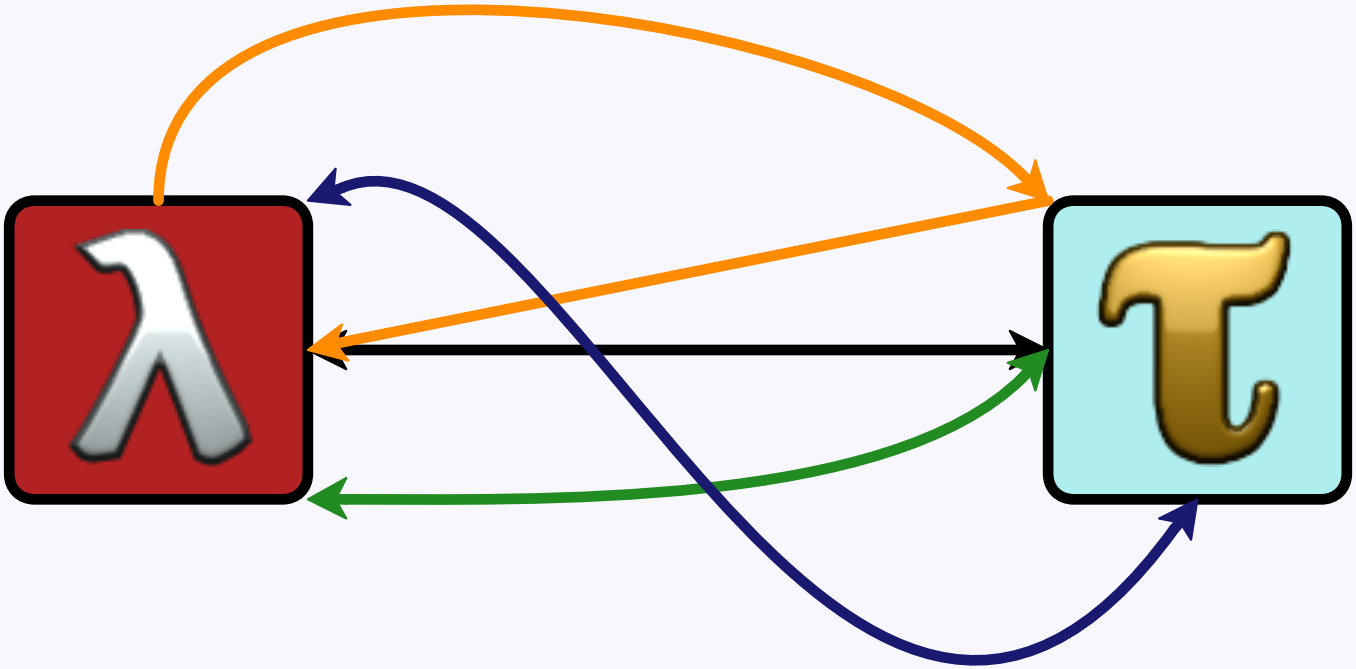
	LE	LU	DE	DU
Error: line 4 expected `String` got `42`	○	○	○	○
Error: line 5 expected `Number` got `"bye"`	○	○	○	○
"bye"	○	○	○	○

Explicit followup about optional typing

Followup Question

> *Agree/Disagree*: Type annotations should not change the behavior of a program.

Goal: cover all interactions



Summary: Survey Design

Solicit opinions on the semantics of a "new" language

Survey Prompt

We are designing a language that mixes typed and untyped code.
....

Gather `Like` × `Expect` preference on eight small programs designed to classify approaches

Question N

```
1 var x : Array(String) = ["hi" "bye"];
2 var y = x;
3 var z : Array(Number) = y;
4 z[0] = 42;
5 var a : Number = z[1];
6 a
```

	LE	LU	DE	DU
Error: line 4 expected `String` got `42`	○	○	○	○
Error: line 5 expected `Number` got `"bye"`	○	○	○	○
"bye"	○	○	○	○

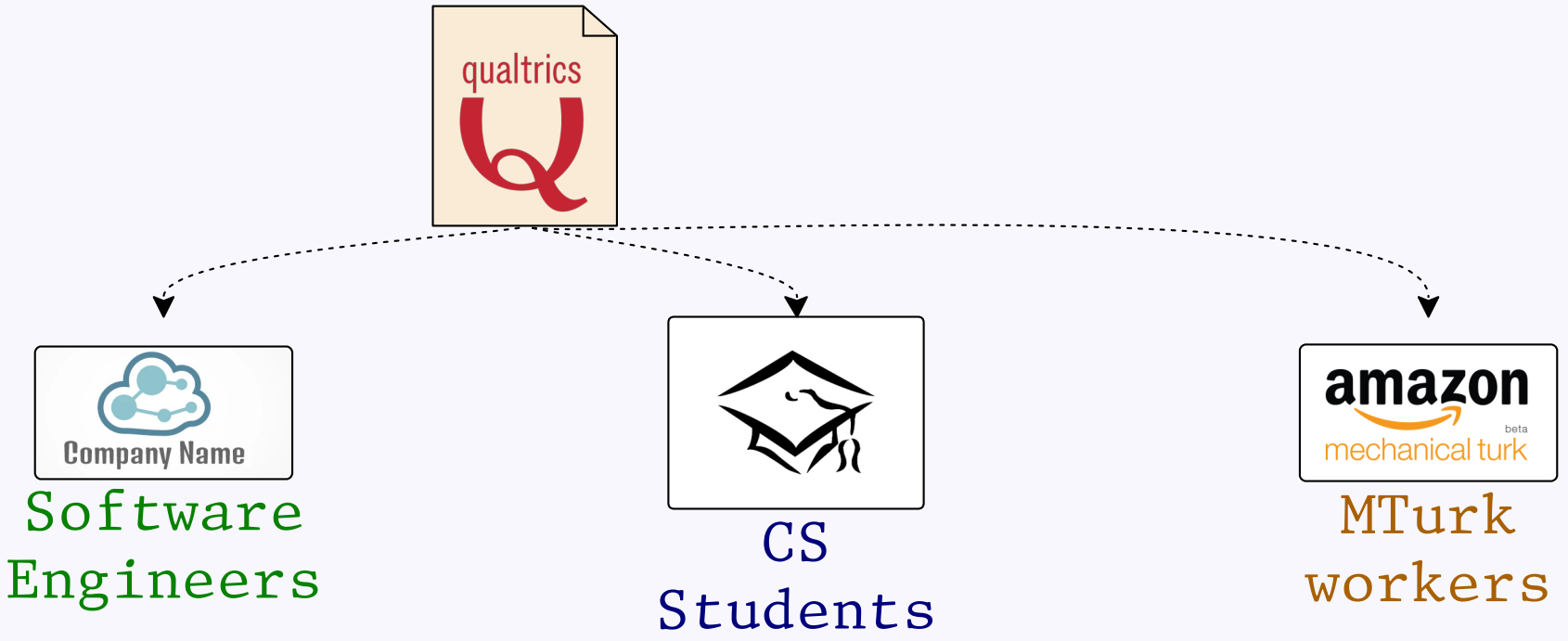
Explicit followup about optional typing

Followup Question

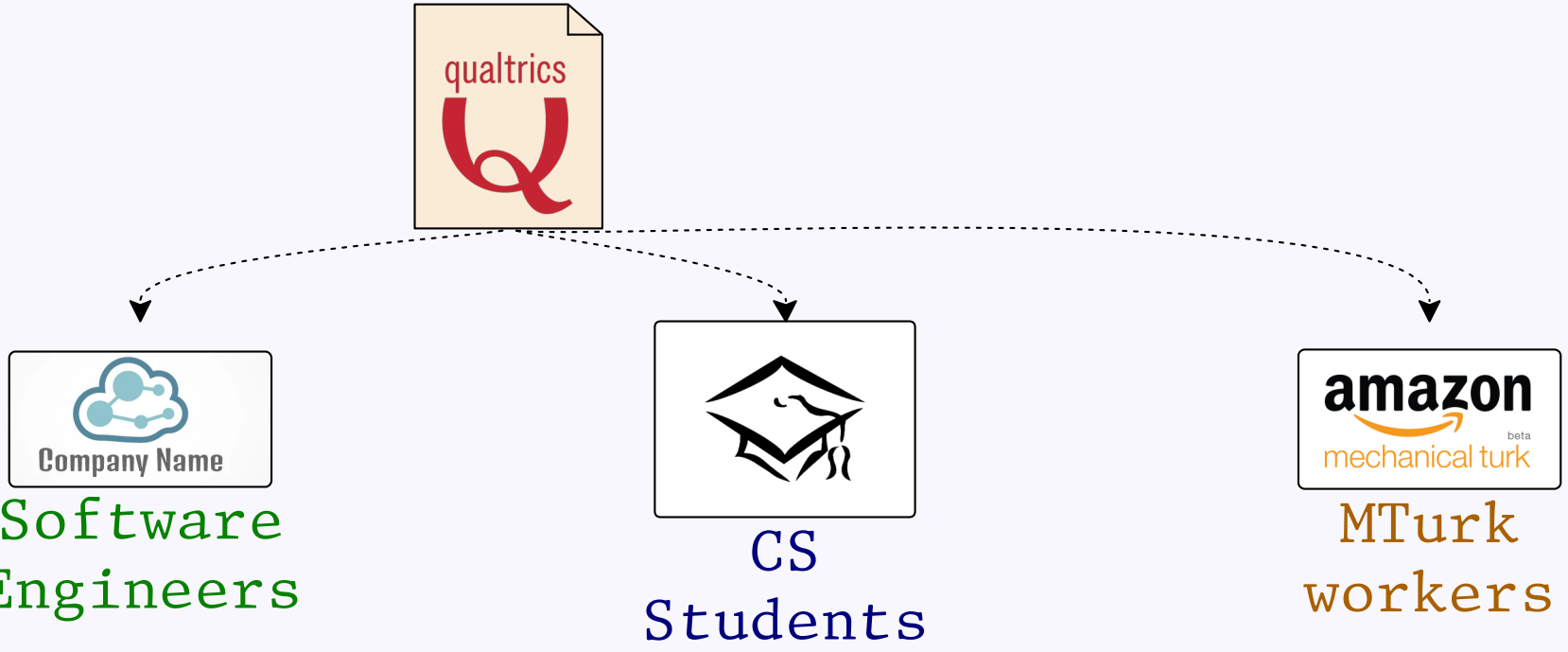
Agree/Disagree: Type annotations should not change the behavior of a program.

Distribution

Distribution



Distribution



Software Engineers

CS Students

MTurk workers

34 participants

17 participants

90 participants
(96 filtered)

Results

Question 1

```
1 | var t = [4, 4];
2 | var x : Number = t;
3 | x
```

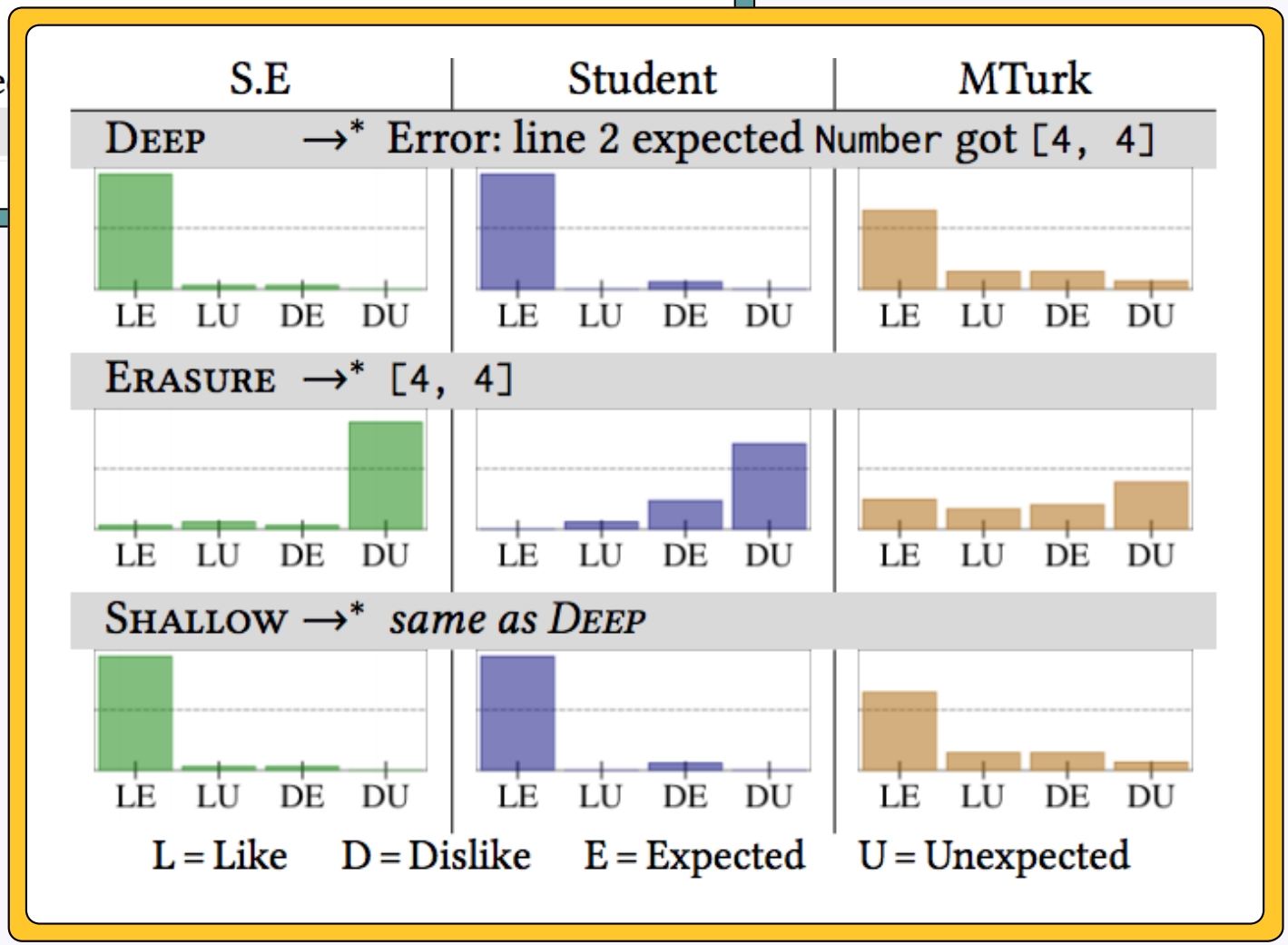
	LE	LU	DE	DU
Error: line 2 expected `Number` got `[4, 4]`	○	○	○	○
`[4, 4]`	○	○	○	○

Question 1

```
1 | var t = [4, 4];
2 | var x : Number = t;
3 | x
```

Error: line 2 expecte

```
[4, 4]
```



Followup Question

Agree/Disagree: Type annotations should not change the behavior of a program.

Followup Question

Agree/Disagree: Type annotations should not change the behavior of a program.

Software Engineers

44% agree

CS Students

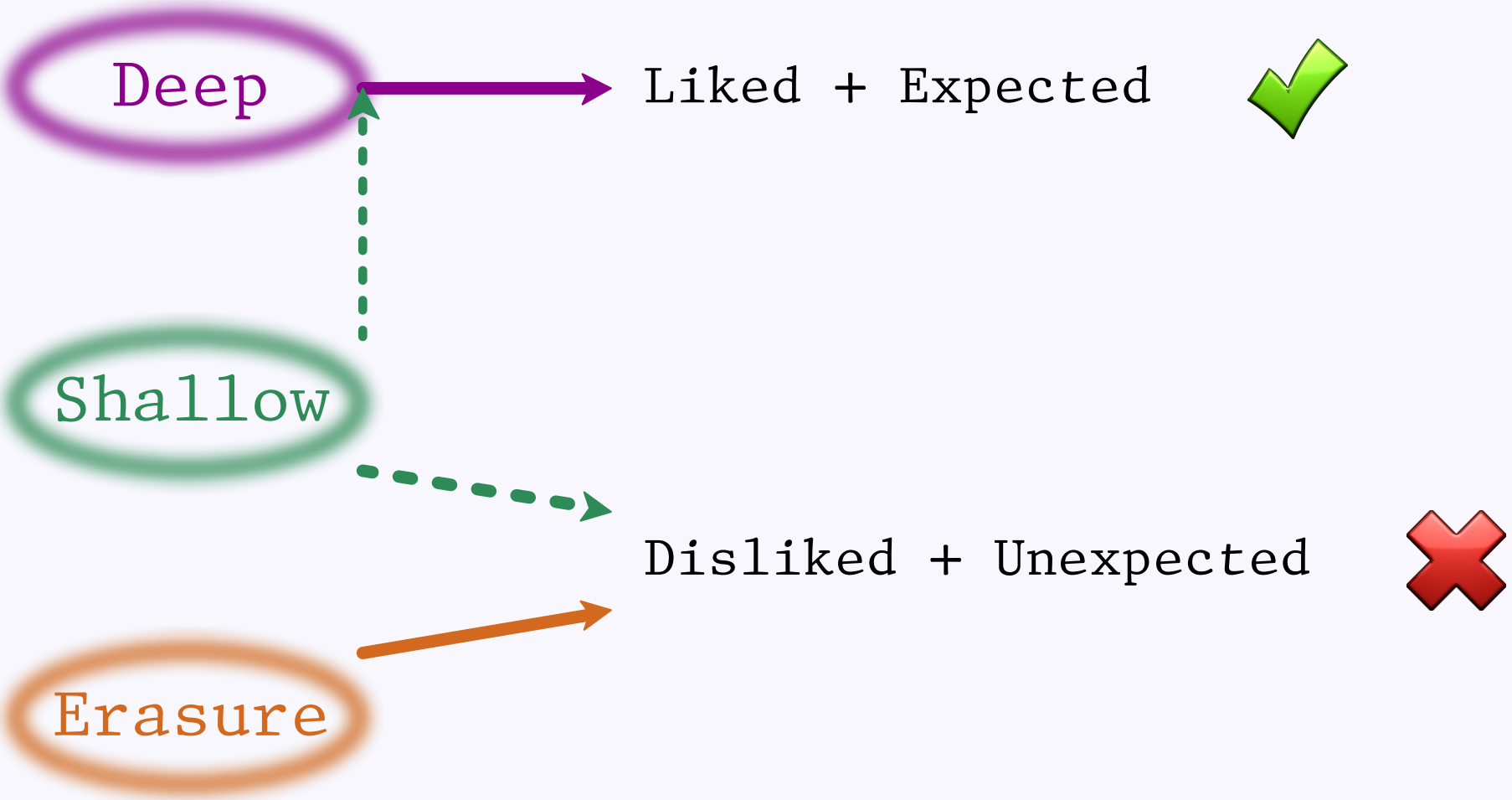
12% agree

MTurk workers

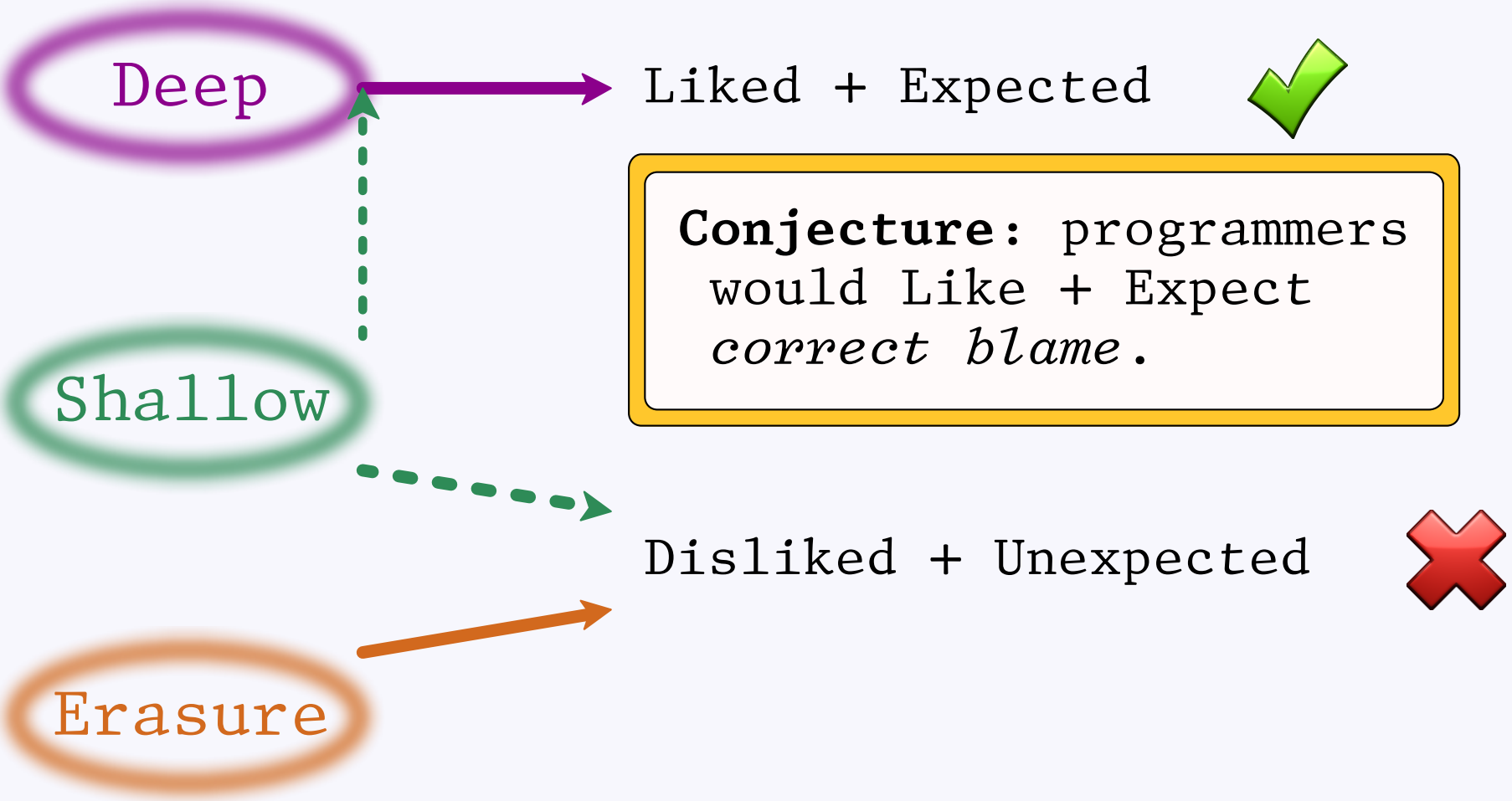
51% agree

Conclusions

Conclusions



Conclusions



Takeaways

Unless there's a strong reason, choose Deep

Programmers seem to expect it!

Non-Deep languages must **document** their design and rationale

Start with the survey examples

cs.brown.edu/research/plt/dl/dls2018

Threats to Validity

Indirect questions ("new" language)

Possible ambiguity:

- lack of type inference
- interpretation of code / error outputs
- runtime vs. static errors

Threats to Generalizability

Selective engineer + student populations

Very diverse MTurk population

Other implications:

- runtime performance
- quality of error messages

Concrete Types

concrete = every value carries a runtime type

Limits expressiveness of "untyped" code

Preferred by Dart users?

Another point to explore!

Followup Question

Agree/Disagree: Type annotations should not change the behavior of a program.

Software Engineers

44% agree

CS Students

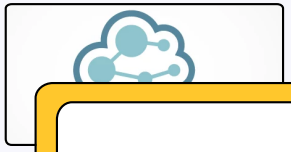
12% agree

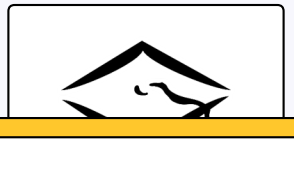
MTurk workers

51% agree

Followup Question

Agree/Disagree: Type annotations should not change the behavior of a program.





amazon

Pop.	Erasure Opinion	Agree	Disagree
S.E.	Like	1	0
	Dislike	14	19
Student	Like	0	0
	Dislike	2	14
MTurk	Like	19	11
	Dislike	18	25

Question 5

```
1 | var obj0 = {
  |     k = 0,
  |     add = function(i : Number) { k = i }};
2 | var t = "hello";
3 | obj0.add(t);
4 | var k : String = obj0.k;
5 | k
```

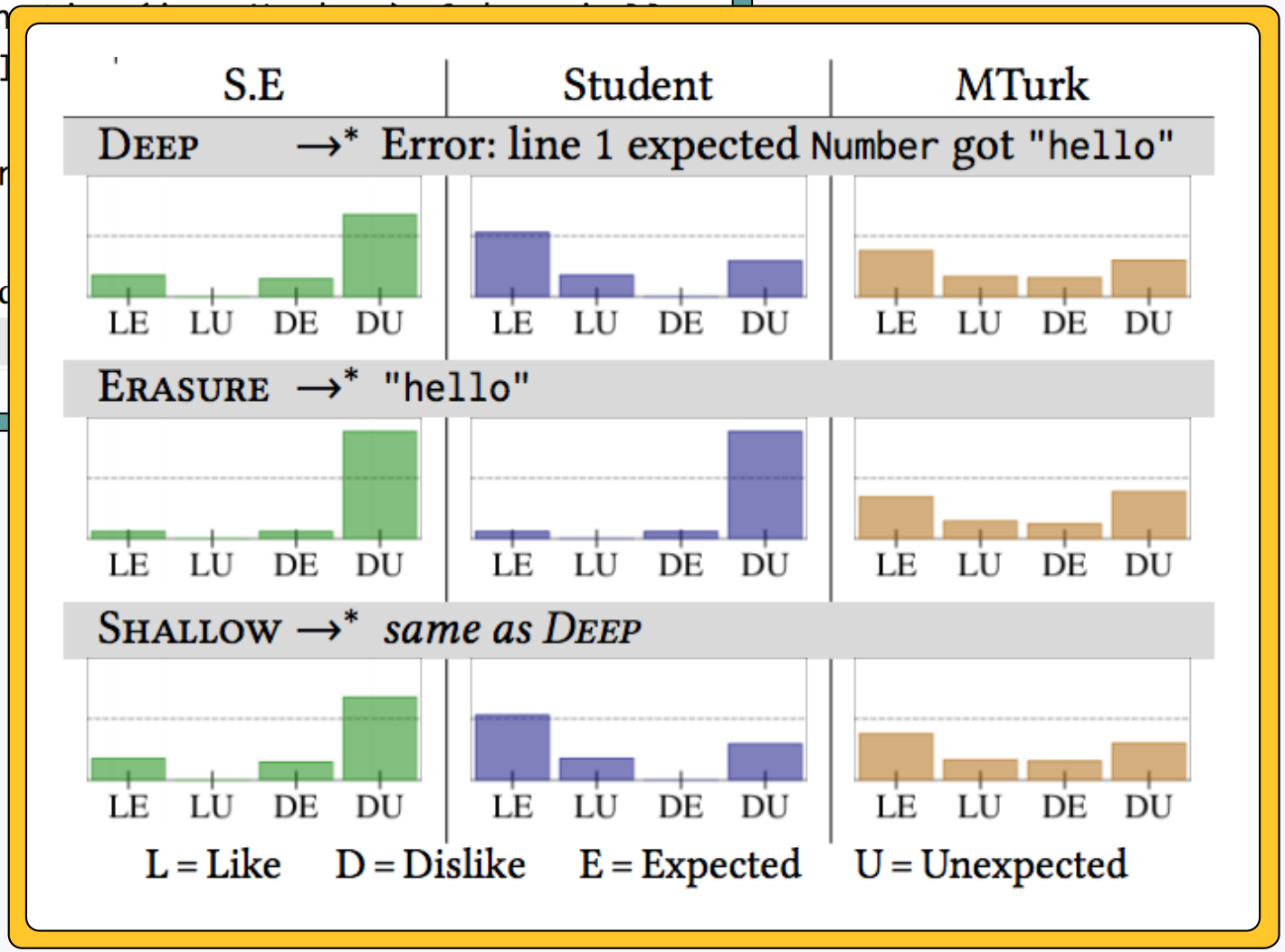
	LE	LU	DE	DU
Error: line 1 expected Number got "hello"	○	○	○	○
"hello"	○	○	○	○

Question 5

```
1 | var obj0 = {
  |     k = 0,
  |     add = fun
2 | var t = "hell
3 | obj0.add(t);
4 | var k : Strin
5 | k
```

Error: line 1 expected

```
"hello"
```



Question 5

```
1 | var obj0 = {
  |     k = 0,
  |     add = function(i : Number) { k = i }};
2 | var t = "hello";
3 | obj0.add(t);
4 | var k : String = obj0.k;
5 | k
```

	LE	LU	DE	DU
Error: line 1 expected Number got "hello"	○	○	○	○
"hello"	○	○	○	○

Why is the compiler complaining about line 1? The error message should be attached to **line 3**, that's the source of the problem!

Question 5

```
1 | var obj0 = {
  |     k = 0,
  |     add = function(i : Number) { k = i }};
2 | var t = "hello";
3 | obj0.add(t);
4 | var k : String = obj0.k;
5 | k
```

	LE	LU	DE	DU
Error: line 3 expected Number got "hello"	○	○	○	○
"hello"	○	○	○	○

Why is the compiler complaining about line 1? The error message should be attached to **line 3**, that's the source of the problem!

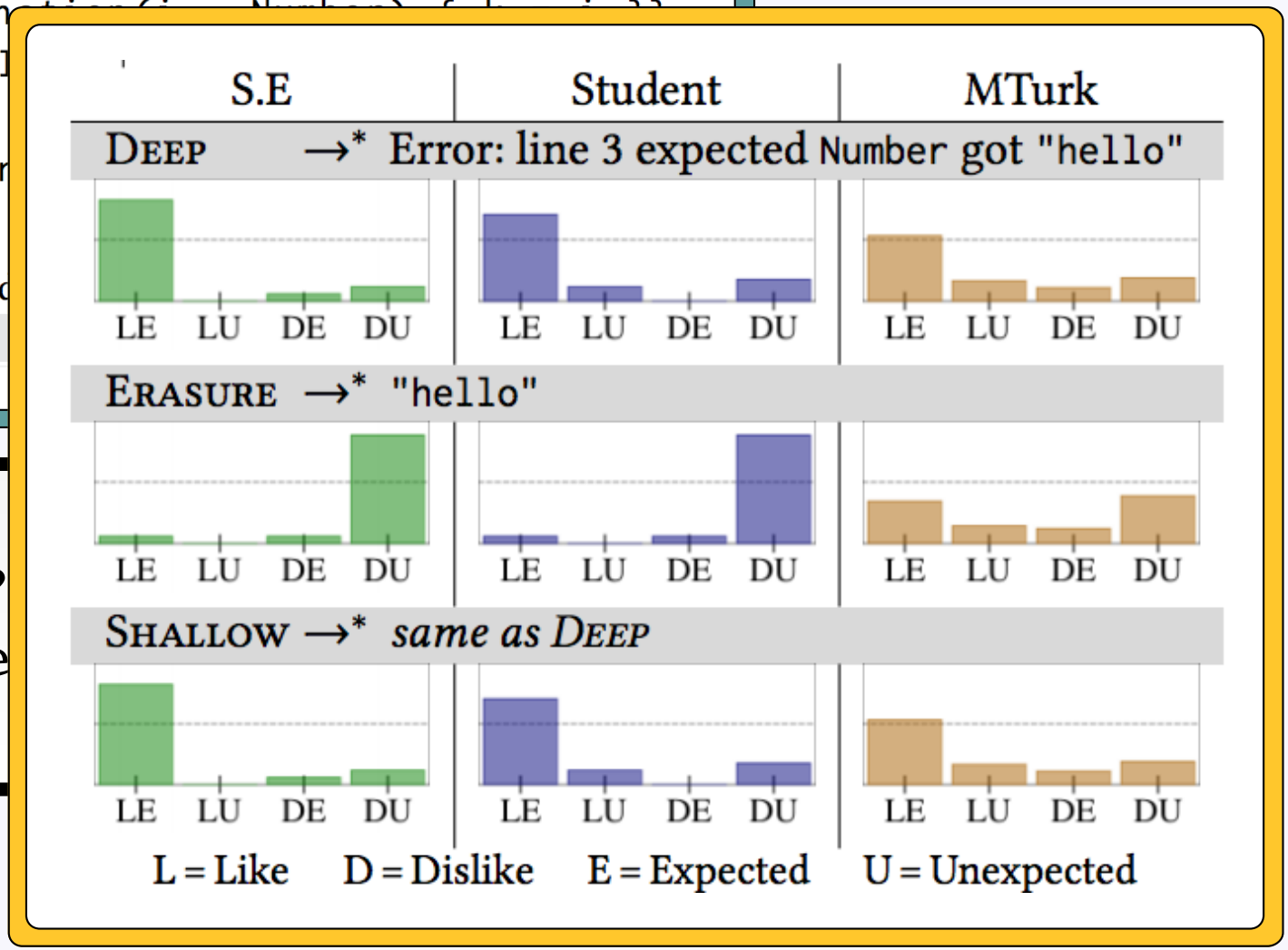
Question 5

```
1 | var obj0 = {
  |     k = 0,
  |     add = fun
2 | var t = "hell
3 | obj0.add(t);
4 | var k : Strin
5 | k
```

Error: line 3 expected

"hello"

Why is
line 1?
attache
of the



Question 7

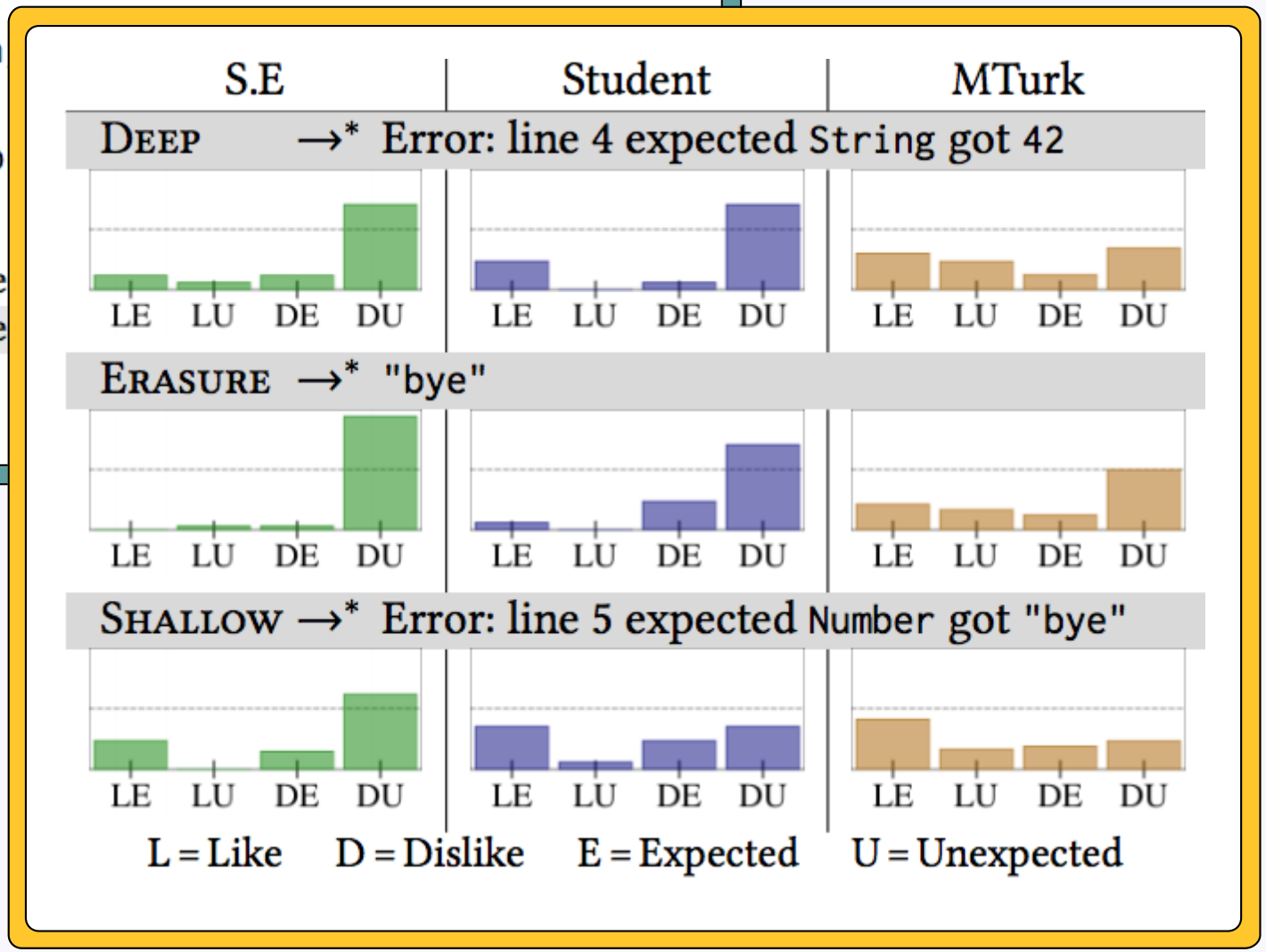
```
1 | var x : Array(String) = ["hi", "bye"];
2 | var y = x;
3 | var z : Array(Number) = y;
4 | z[0] = 42;
5 | var a : Number = z[1];
6 | a
```

	LE	LU	DE	DU
Error: line 4 expected String got 42	○	○	○	○
Error: line 5 expected Number got "bye"	○	○	○	○
"bye"	○	○	○	○

Question 7

```
1 | var x : Array(String) = ["hi", "bye"];
2 | var y = x;
3 | var z : Arra
4 | z[0] = 42;
5 | var a : Numb
6 | a
```

Error: line 4 expecte
Error: line 5 expecte
"bye"



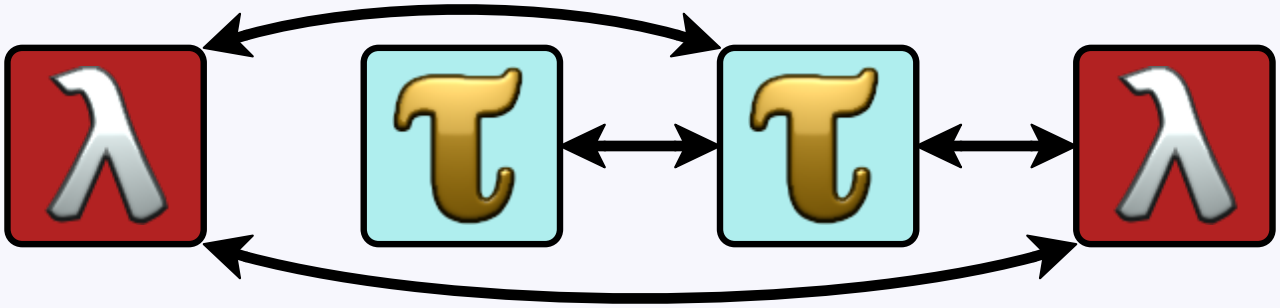
`Shallow` cannot get 'stuck' if:

1. Total reduction relation for dynamic code

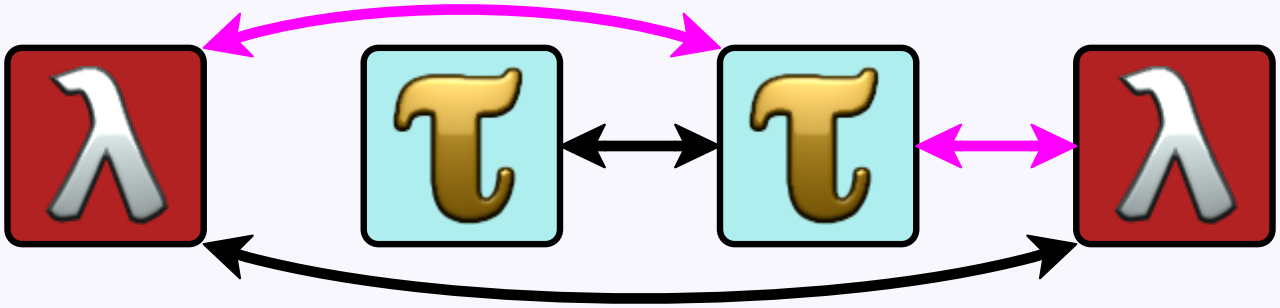
2. Partial reduction relation for static code (possible to get stuck)

3. Shallow checks can distinguish stuck vs. non-stuck states

Three Approaches to G.T.

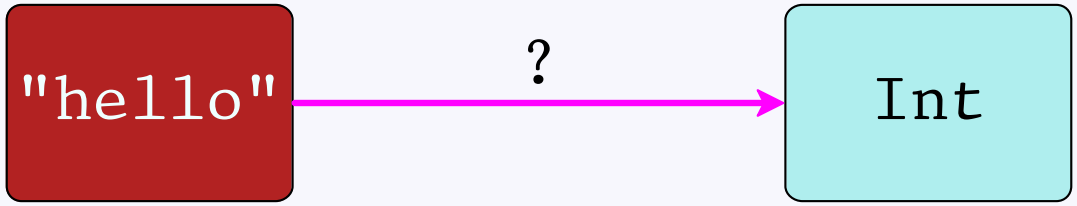

λ
τ
τ
λ

How to enforce the type boundaries?



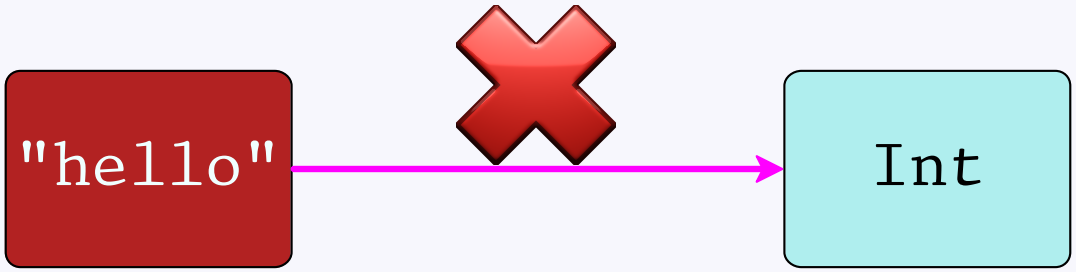
Example: Base Type

Deep Shallow Erasure

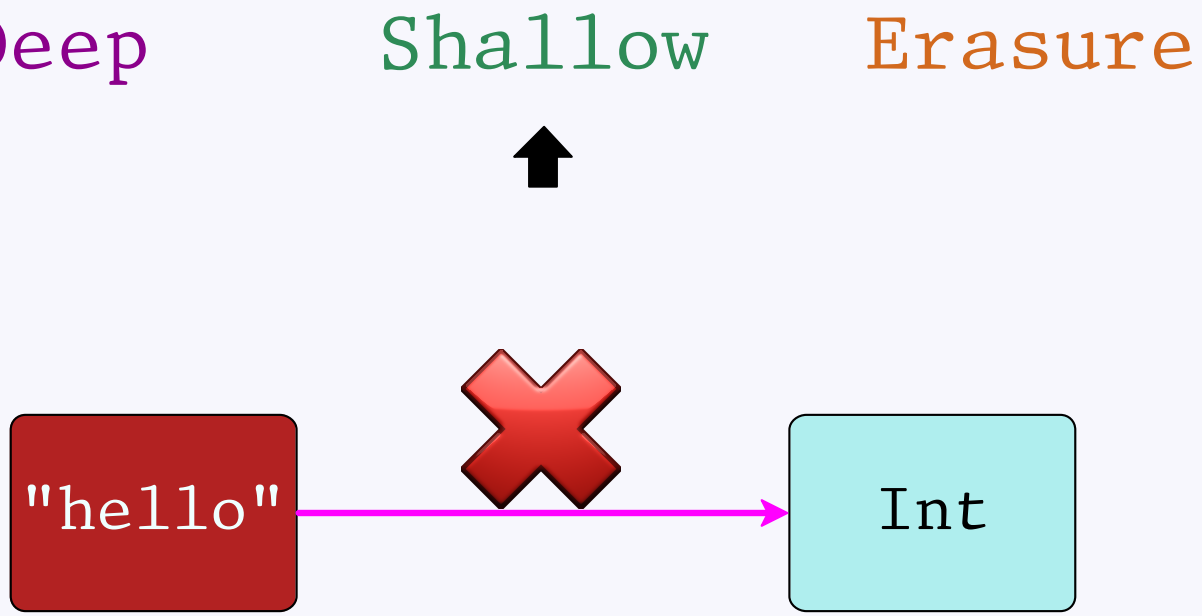


Example: Base Type

Deep Shallow Erasure

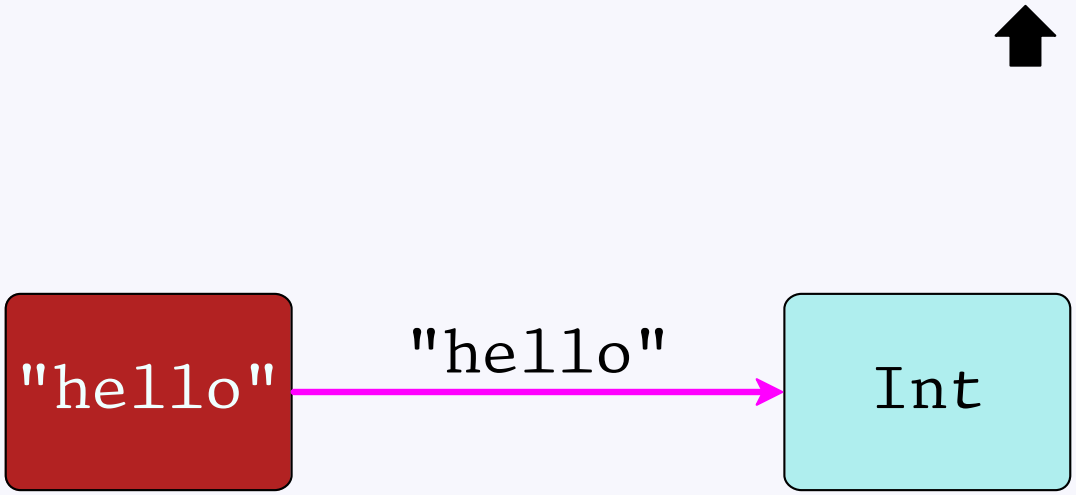


Example: Base Type



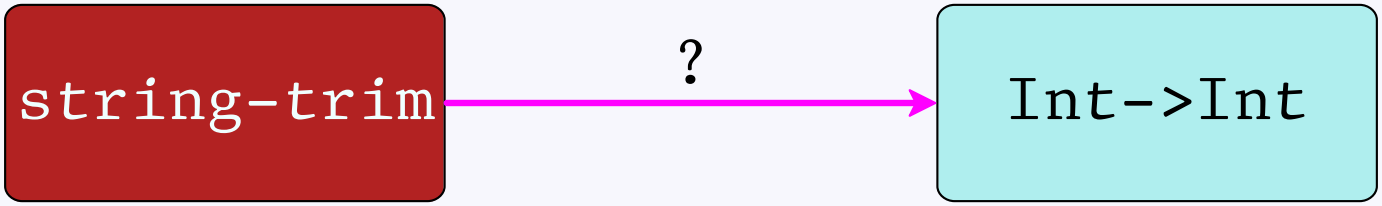
Example: Base Type

Deep Shallow Erasure



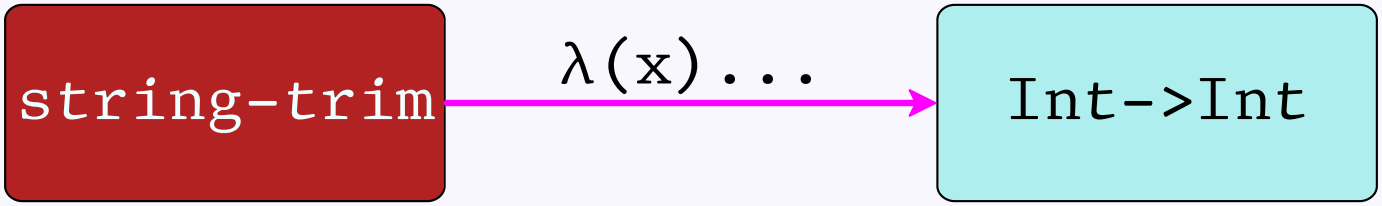
Example: Higher-Order Type

Deep Shallow Erasure



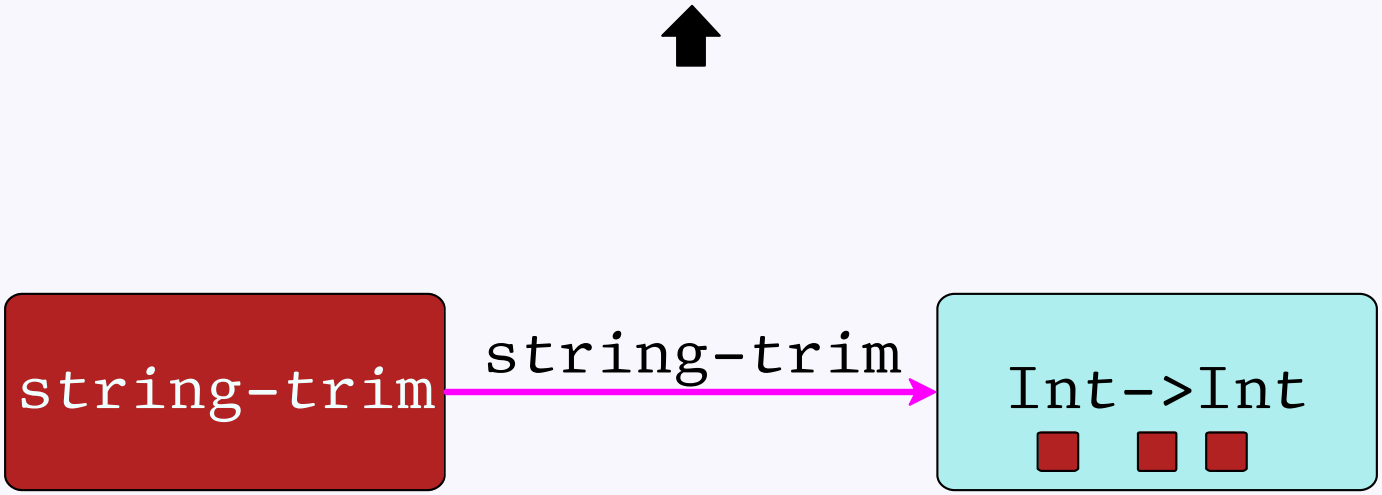
Example: Higher-Order Type

Deep Shallow Erasure



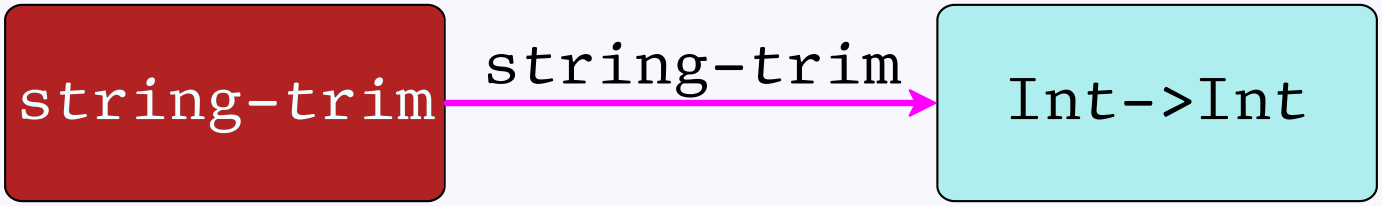
Example: Higher-Order Type

Deep Shallow Erasure



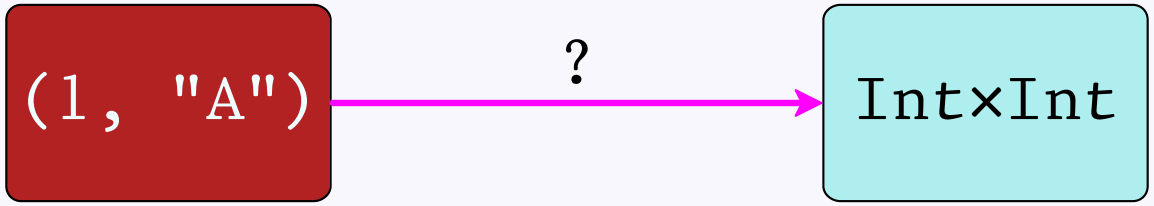
Example: Higher-Order Type

Deep Shallow Erasure



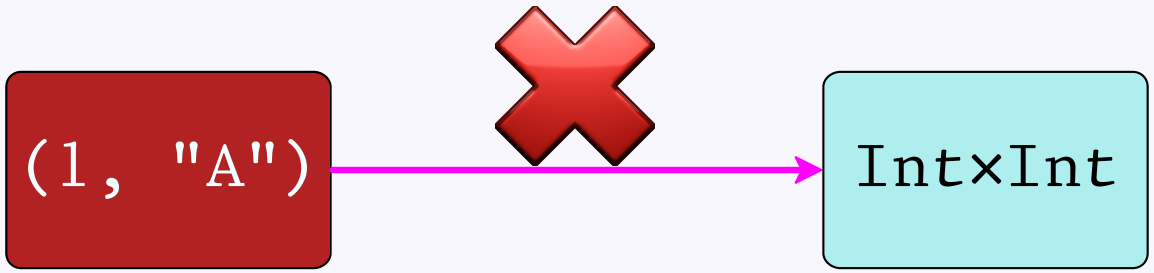
Example: Inductive Type

Deep Shallow Erasure



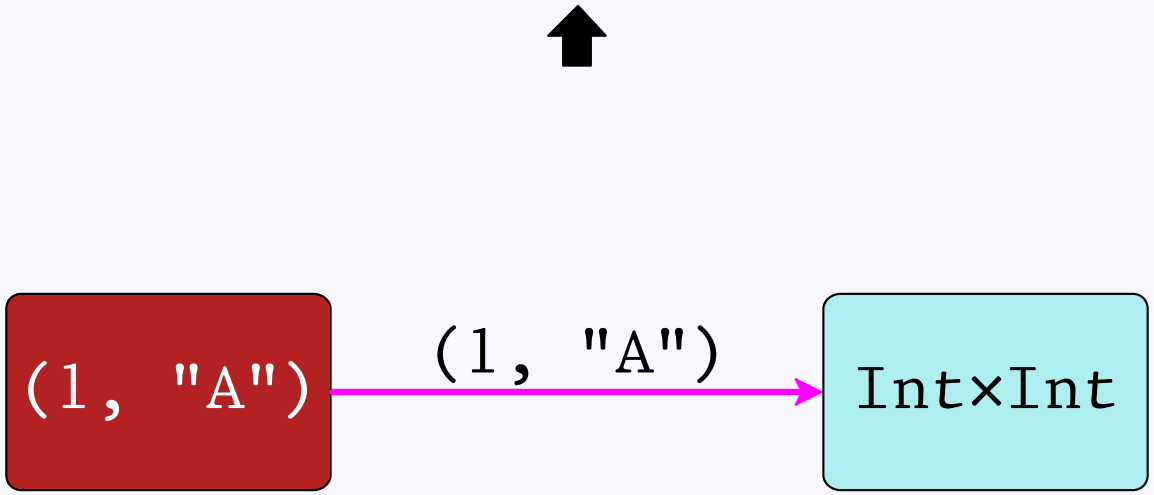
Example: Inductive Type

Deep Shallow Erasure



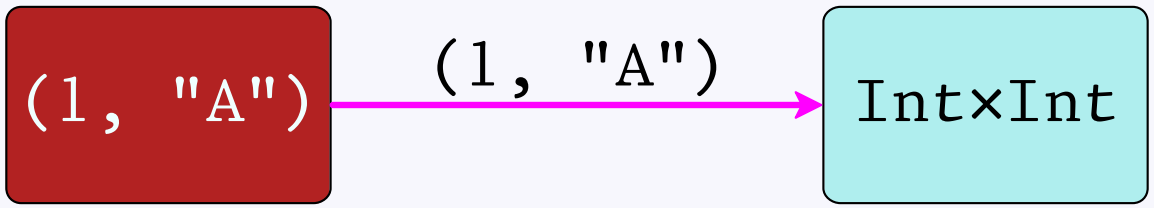
Example: Inductive Type

Deep Shallow Erasure



Example: Inductive Type

Deep Shallow Erasure



Three Approaches, Summary

	Deep	Shallow	Erasure
invariant	t	C(t)	v
base types	check	check	-
coinductive types	wrap	check	-
inductive types	traverse	check	-
boundaries	static, higher-order	static, selectors	-