Gradual Typing with Efficient Object Casts

Michael M. Vitousek
Adviser: Jeremy G. Siek

michael.vitousek@colorado.edu

University of Colorado at Boulder
Boulder, Colorado, USA

June 14, 2012

Introduction to gradual typing

Gradual typing: static typing and dynamic typing in the same program

Introduction to gradual typing

Gradual typing: static typing and dynamic typing in the same program

```
1: def bar(y:int):
2:     y × 2
3: m = 21
4: bar(m)
```

```
1: def baz(z:int):
2:     z^2
3: k = "hi"
4: baz(k)
```

Introduction to gradual typing

Gradual typing: static typing and dynamic typing in the same program, moderated by casts

1: **def** *bar*(y:int):
2: $\quad y \times 2$
3: $m = 21$::int $\Rightarrow$ dyn
4: *bar*(m::dyn $\Rightarrow$ int)

1: **def** *baz*(z:int):
2: $\quad z^2$
3: $k =$ "hi"::str $\Rightarrow$ dyn
4: *baz*(k::dyn $\Rightarrow$ int)

Casts accumulate over time

```
1: def find_files(files, fun):
2:   for file in files:
3:     if file.is_directory():
4:       find_in_dir(file::dyn ⇒ file, allfiles::dyn ⇒ list)
5:     else: print fun(file)
6: def find_in_dir(dir:file, allfiles:list) → unit:
7:   find_files(file.members()::list ⇒ dyn, allfiles::list ⇒ dyn)
```

- *allfiles* goes through many casts
- If data builds up on the object at every cast, each additional cast will take up more space

$$allfiles::\text{list} \Rightarrow \text{dyn} \Rightarrow \text{list} \Rightarrow \text{dyn} \Rightarrow \ldots$$

Threesome casts collapse under composition

- Siek and Wadler: threesomes

$$(T_1 \overset{T_2}{\Longrightarrow} T_3) \circ (T_3 \overset{T_4}{\Longrightarrow} T_5) = T_1 \overset{T_2 \sqcap T_4}{\Longrightarrow} T_5$$

- Threesomes provided a solution to space-efficient function casts
- We want them to do the same for imperative objects

References contain casts

1: $a{:}\{x{:}\mathsf{dyn}\} = \{x = 10\}$
2: $b{:}\{x{:}\mathsf{int}\} = a$
3: $b.x$
4: $b.x = 42$
5: $a.x =$ “Hello!”
6: $b.x$

References contain casts

1: $a{:}\{x{:}\mathsf{dyn}\} = \{x = 10{::}\mathsf{int} \overset{\mathsf{int}}{\Longrightarrow} \mathsf{dyn}\}$
2: $b{:}\{x{:}\mathsf{int}\} = a{::}\{x{:}\mathsf{dyn}\} \overset{\{x:\mathsf{int}\}}{\Longrightarrow} \{x{:}\mathsf{int}\}$
3: $b.x$
4: $b.x = 42$
5: $a.x = \text{“Hello!”}{::}\mathsf{str} \overset{\mathsf{str}}{\Longrightarrow} \mathsf{dyn}$
6: $b.x$

References contain casts

1: $a{:}\{x{:}\mathsf{dyn}\} = \{x = 10{::}\mathsf{int} \overset{\mathsf{int}}{\Longrightarrow} \mathsf{dyn}\}$

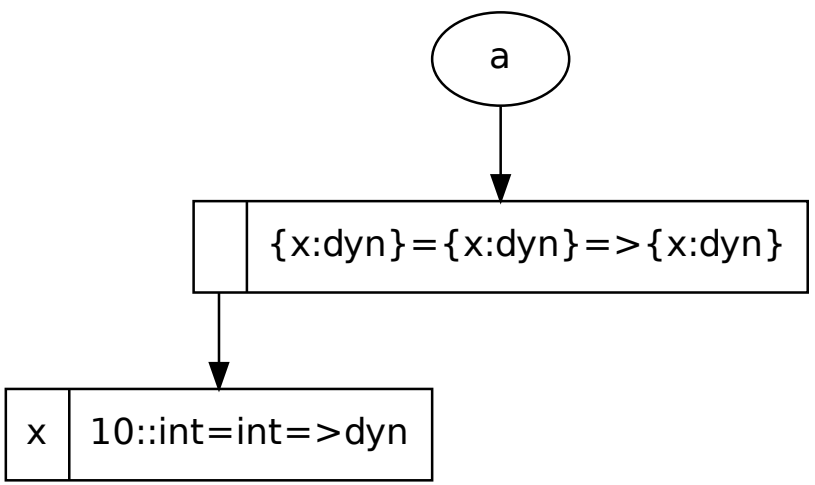
2: $b{:}\{x{:}\mathsf{int}\} = a{::}\{x{:}\mathsf{dyn}\} \overset{\{x:\mathsf{int}\}}{\Longrightarrow} \{x{:}\mathsf{int}\}$

3: $b.x$

4: $b.x = 42$

5: $a.x = \text{“Hello!”}{::}\mathsf{str} \overset{\mathsf{str}}{\Longrightarrow} \mathsf{dyn}$

6: $b.x$



References contain casts

1: $a{:}\{x{:}\text{dyn}\} = \{x = 10{::}\text{int} \overset{\text{int}}{\Longrightarrow} \text{dyn}\}$

2: $b{:}\{x{:}\text{int}\} = a{::}\{x{:}\text{dyn}\} \overset{\{x:\text{int}\}}{\Longrightarrow} \{x{:}\text{int}\}$

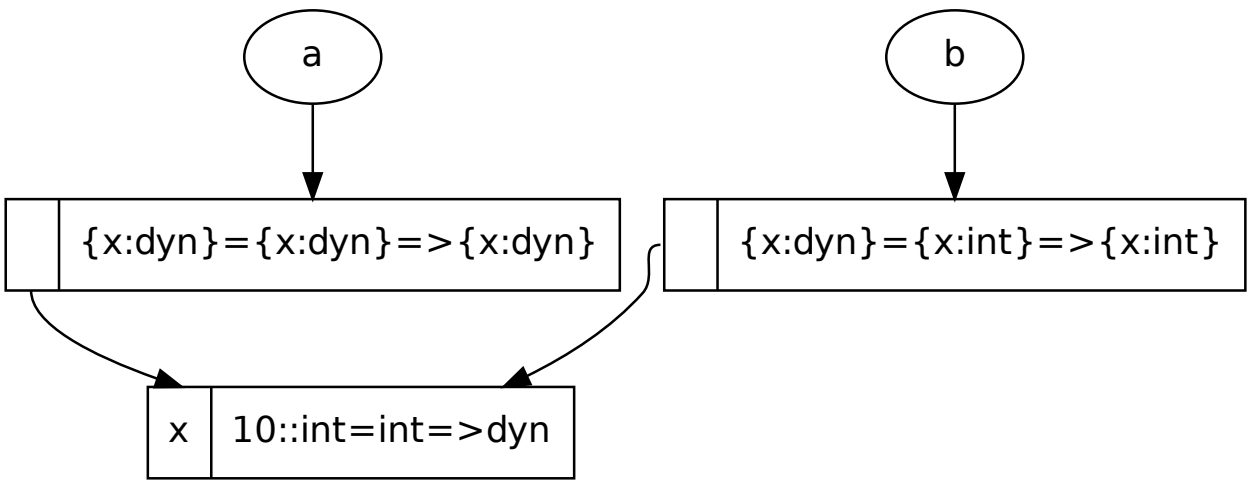
3: $b.x$

4: $b.x = 42$

5: $a.x = \text{“Hello!”}{::}\text{str} \overset{\text{str}}{\Longrightarrow} \text{dyn}$

6: $b.x$

$$\{x{:}\text{dyn}\} \overset{\{x:\text{dyn}\}}{\Longrightarrow} \{x{:}\text{dyn}\} \circ \{x{:}\text{dyn}\} \overset{\{x:\text{int}\}}{\Longrightarrow} \{x{:}\text{int}\} = \{x{:}\text{dyn}\} \overset{\{x:\text{int}\}}{\Longrightarrow} \{x{:}\text{int}\}$$



References contain casts

1: $a{:}\{x{:}\text{dyn}\} = \{x = 10{::}\text{int} \overset{\text{int}}{\Longrightarrow} \text{dyn}\}$

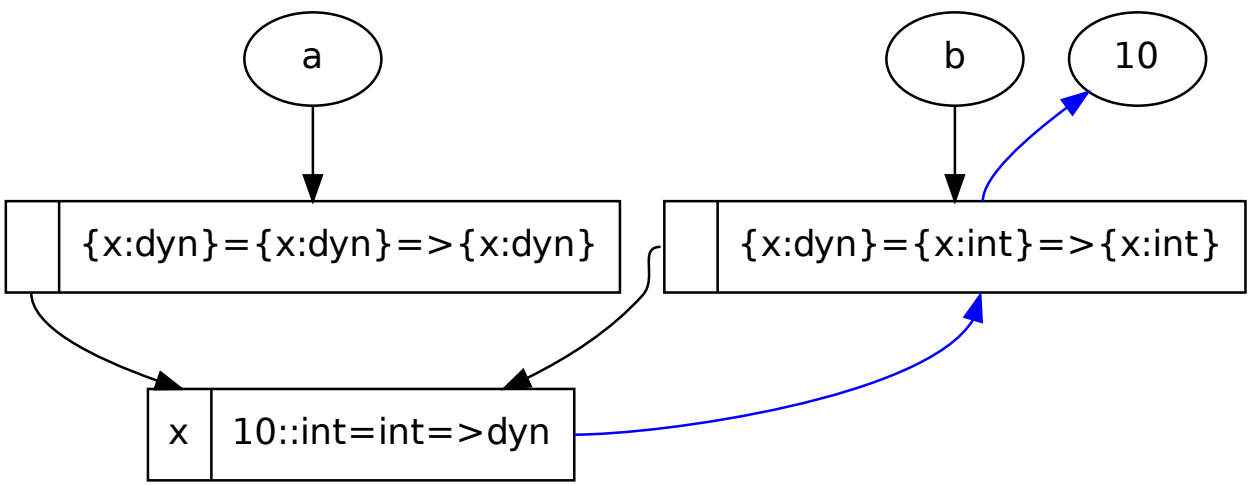
2: $b{:}\{x{:}\text{int}\} = a{::}\{x{:}\text{dyn}\} \overset{\{x:\text{int}\}}{\Longrightarrow} \{x{:}\text{int}\}$

3: $b.x$

4: $b.x = 42$

5: $a.x = \text{“Hello!”}{::}\text{str} \overset{\text{str}}{\Longrightarrow} \text{dyn}$

6: $b.x$



References contain casts

1: $a{:}\{x{:}\text{dyn}\} = \{x = 10{::}\text{int} \overset{\text{int}}{\Longrightarrow} \text{dyn}\}$

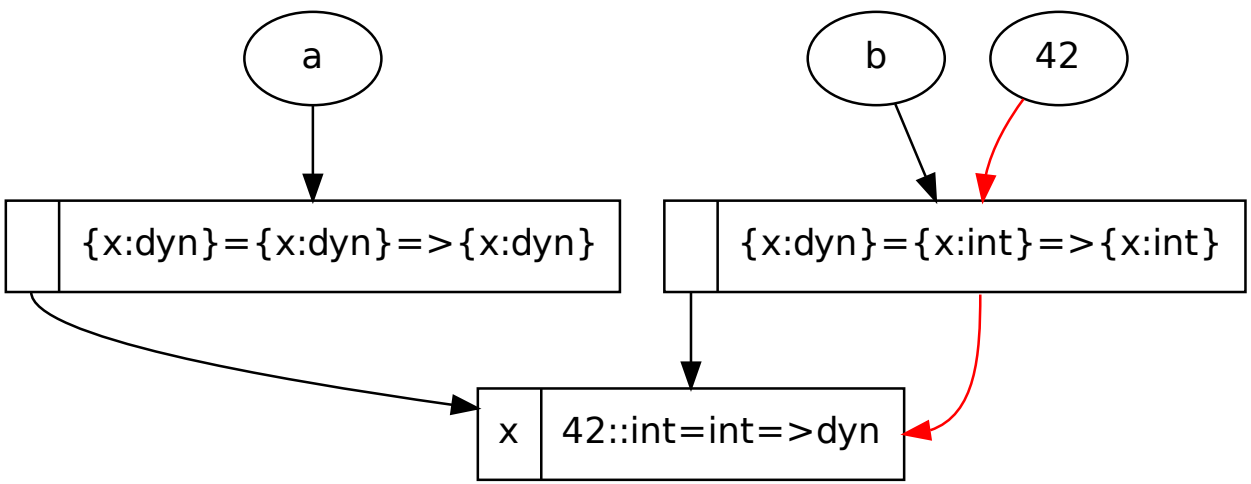
2: $b{:}\{x{:}\text{int}\} = a{::}\{x{:}\text{dyn}\} \overset{\{x:\text{int}\}}{\Longrightarrow} \{x{:}\text{int}\}$

3: $b.x$

4: $b.x = 42$

5: $a.x = \text{“Hello!”}{::}\text{str} \overset{\text{str}}{\Longrightarrow} \text{dyn}$

6: $b.x$



References contain casts

1: $a{:}\{x{:}\mathsf{dyn}\} = \{x = 10{::}\mathsf{int} \overset{\mathsf{int}}{\Longrightarrow} \mathsf{dyn}\}$

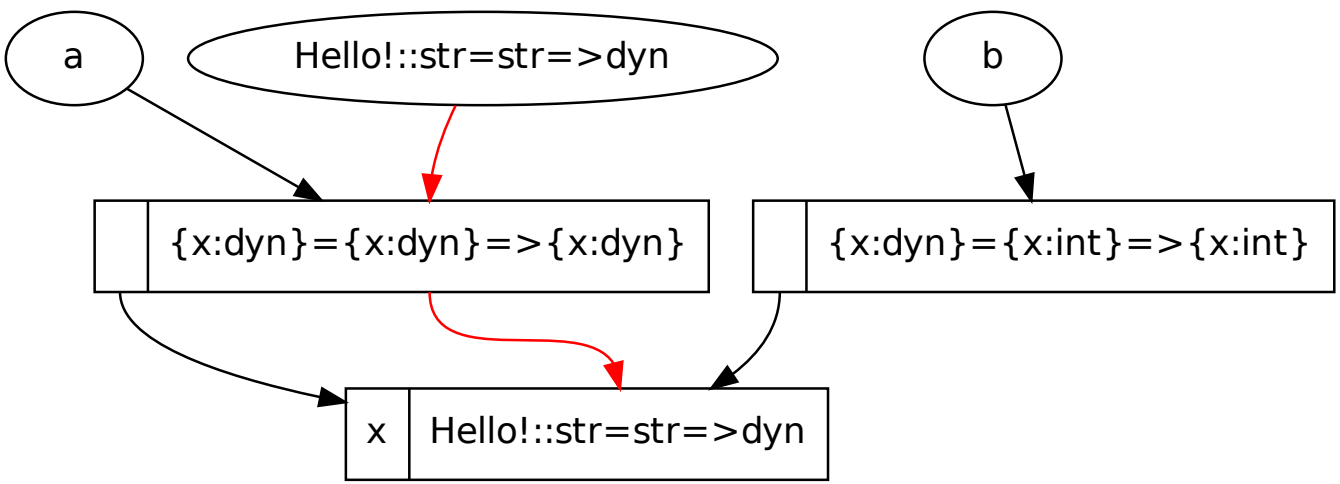
2: $b{:}\{x{:}\mathsf{int}\} = a{::}\{x{:}\mathsf{dyn}\} \overset{\{x:\mathsf{int}\}}{\Longrightarrow} \{x{:}\mathsf{int}\}$

3: $b.x$

4: $b.x = 42$

5: $a.x = \text{“Hello!”}{::}\mathsf{str} \overset{\mathsf{str}}{\Longrightarrow} \mathsf{dyn}$

6: $b.x$



References contain casts

1: $a{:}\{x{:}\text{dyn}\} = \{x = 10{::}\text{int} \overset{\text{int}}{\Longrightarrow} \text{dyn}\}$

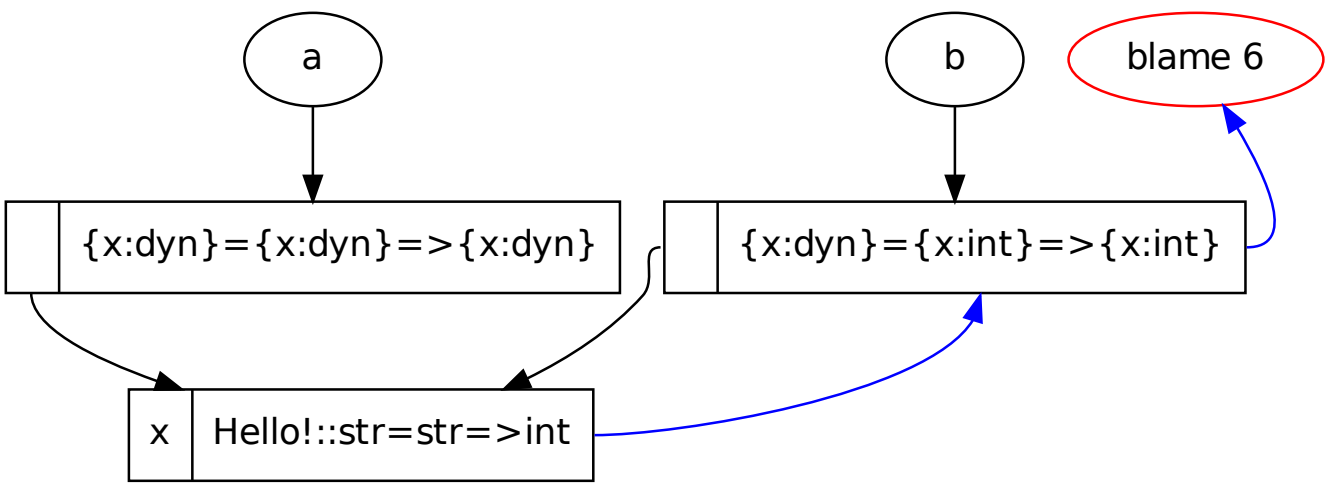
2: $b{:}\{x{:}\text{int}\} = a{::}\{x{:}\text{dyn}\} \overset{\{x:\text{int}\}}{\Longrightarrow} \{x{:}\text{int}\}$

3: $b.x$

4: $b.x = 42$

5: $a.x = \text{“Hello!”}{::}\text{str} \overset{\text{str}}{\Longrightarrow} \text{dyn}$

6: $b.x$



Monotonic objects

Monotonic objects

No update can invalidate any view of an object anywhere in the program

Monotonic object casts prevent invalid updates

1: $a{:}\{x{:}\mathsf{dyn}\} = \{x = 10{::}\mathsf{int} \overset{\mathsf{int}}{\Longrightarrow} \mathsf{dyn}\}$

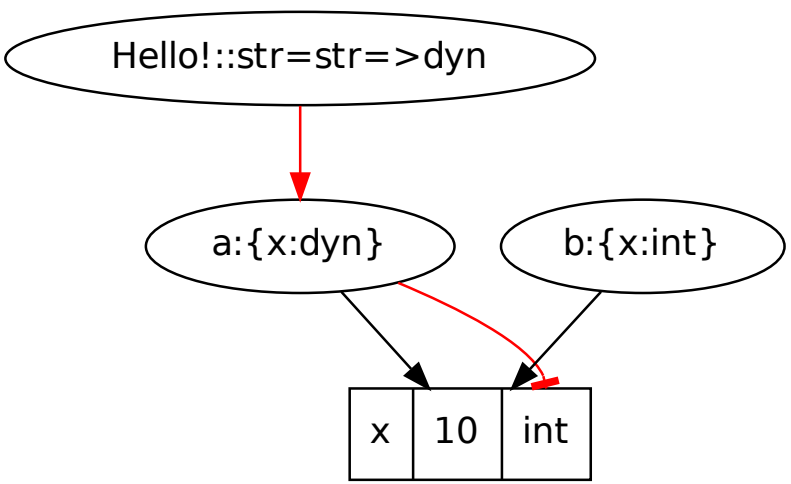
2: $b{:}\{x{:}\mathsf{int}\} = a{::}\{x{:}\mathsf{dyn}\} \overset{\{x:\mathsf{int}\}}{\Longrightarrow} \{x{:}\mathsf{int}\}$

3: $b.x$

4: $b.x = 42$

5: $a.x = \text{“Hello!”}{::}\mathsf{str} \overset{\mathsf{str}}{\Longrightarrow} \mathsf{dyn}$

6: $b.x$



Conclusions

- Space efficient object casts — every reference uses a constant amount of space
- Consistent with semantics of imperative objects
- Efficient reads and writes
  - Straightforward approach has slight overhead on uncasted objects
  - Monotonic objects have faster reads, but restrictive