Efficiency of Bidirectional Transformations

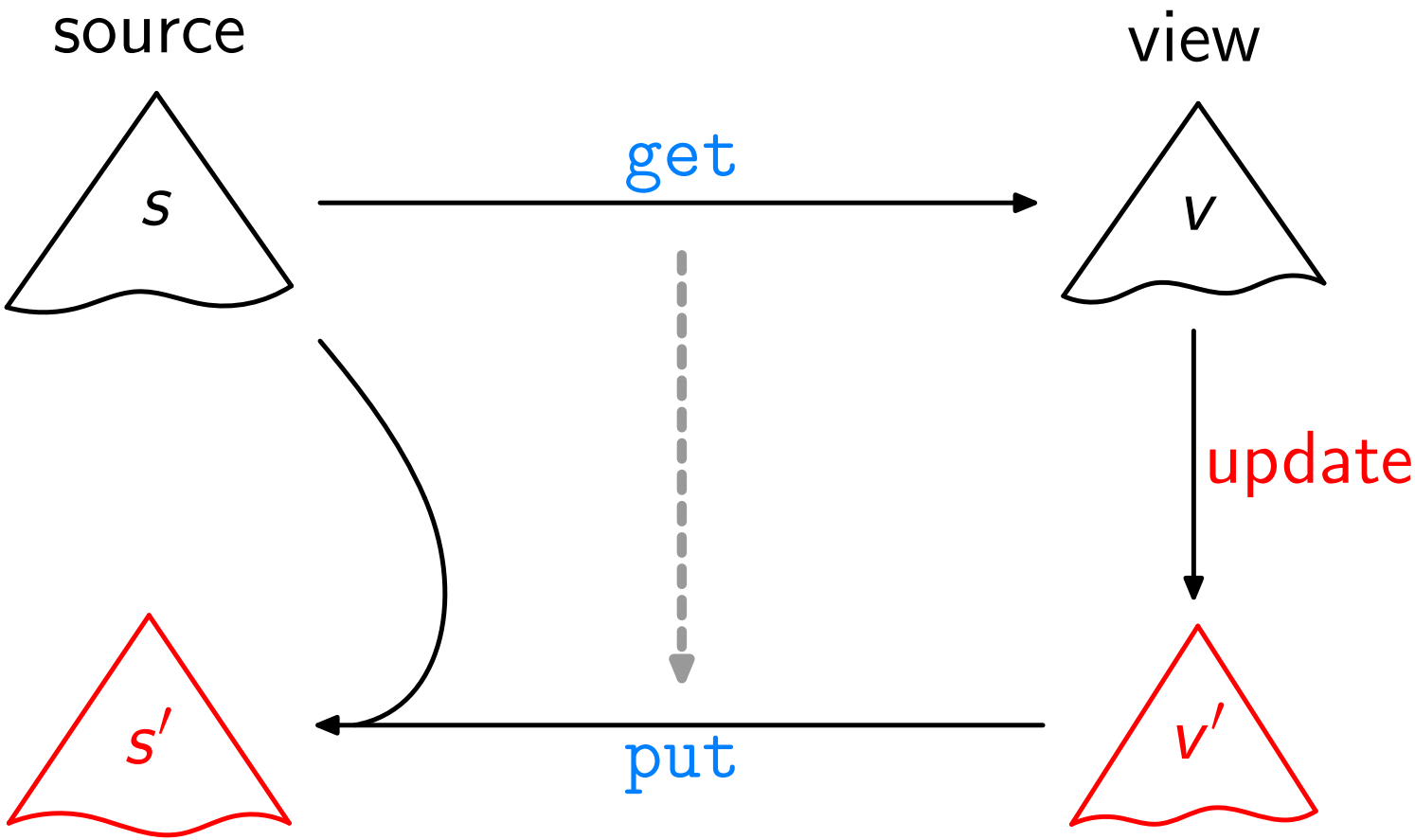
J. Voigtländer

University of Bonn

Dagstuhl Seminar "bx"

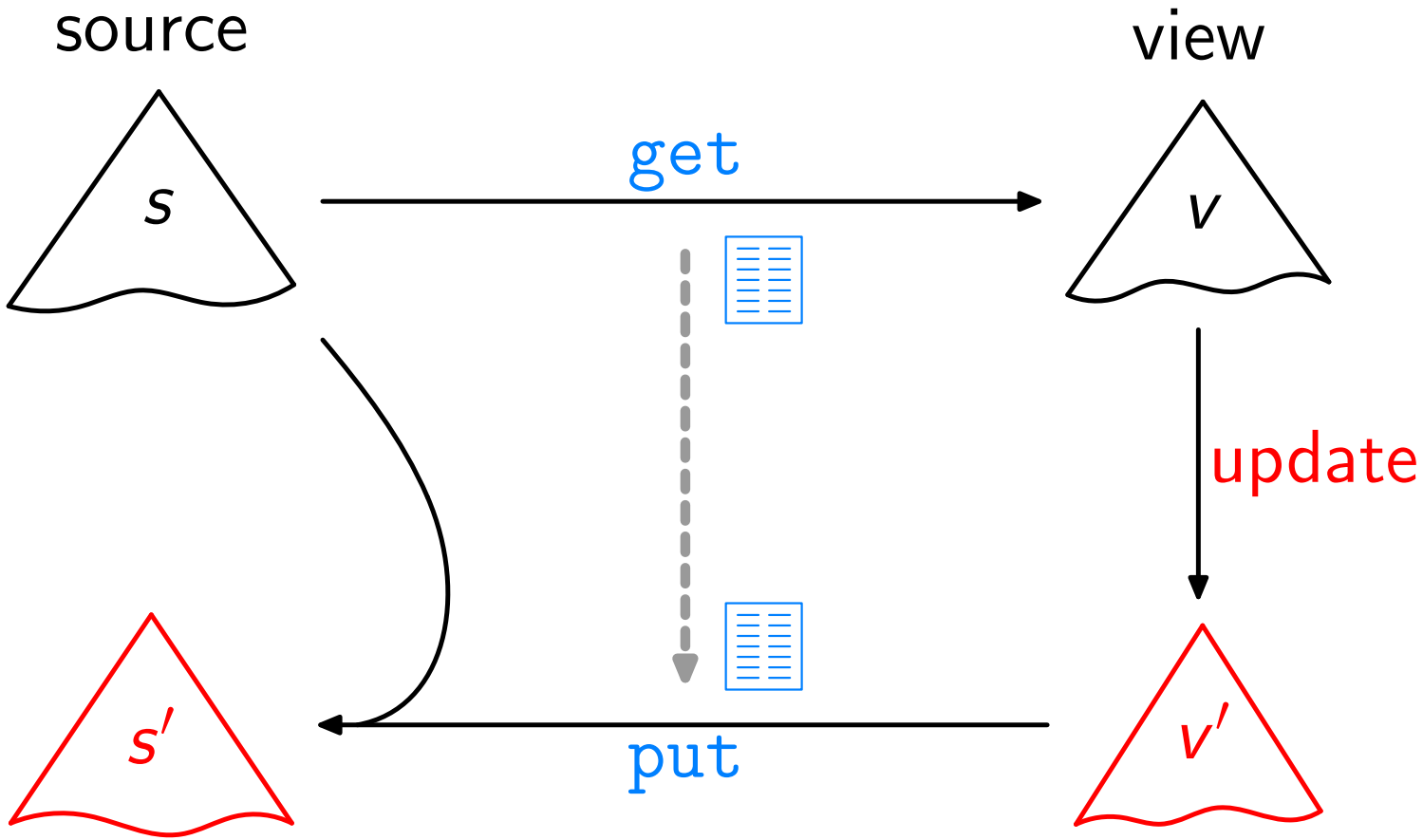
January 20th, 2011

Bidirectional Transformation



Bidirectionalization

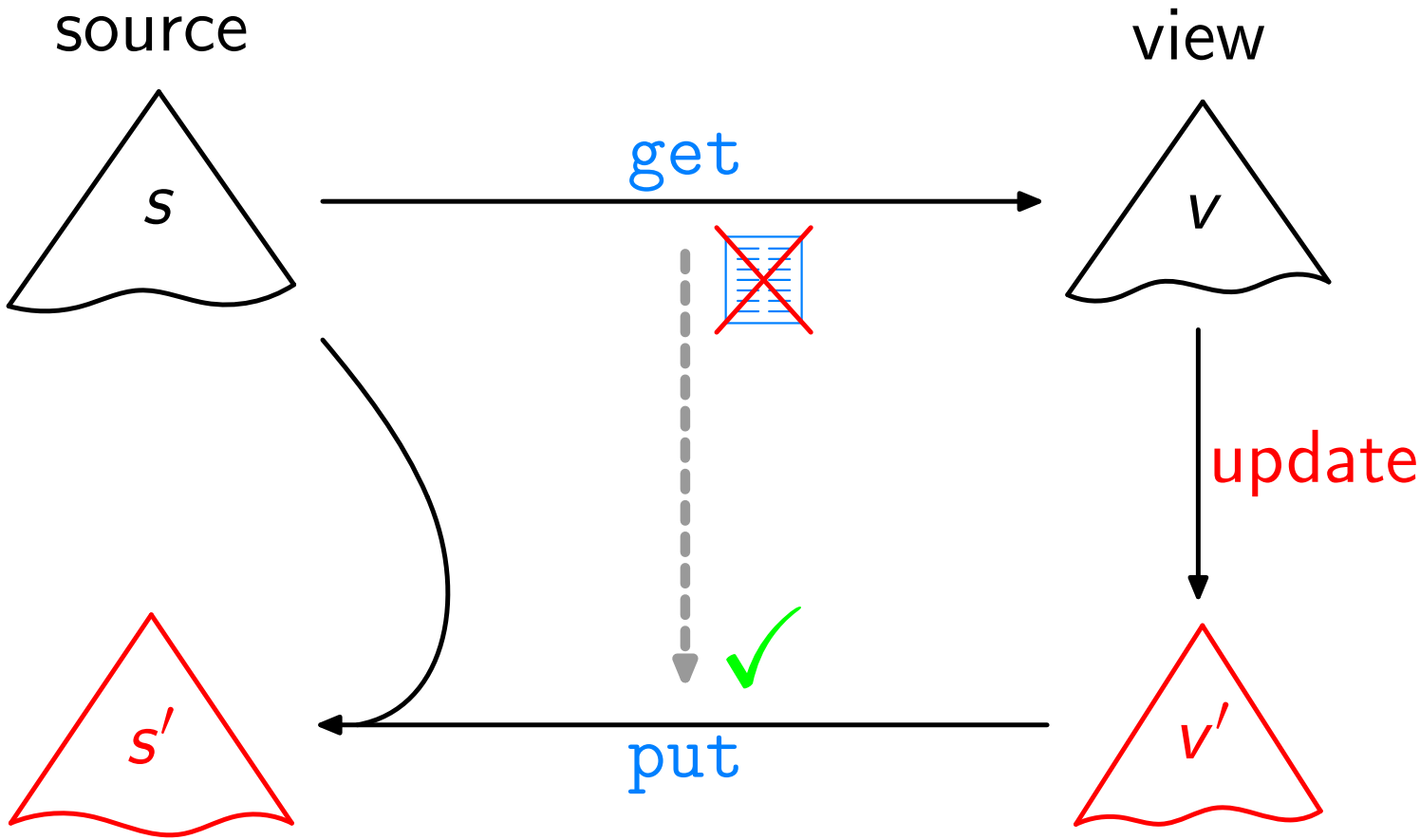
Bidirectional Transformation



Syntactic Bidirectionalization

[Matsuda et al., ICFP'07]

Bidirectional Transformation



Semantic Bidirectionalization

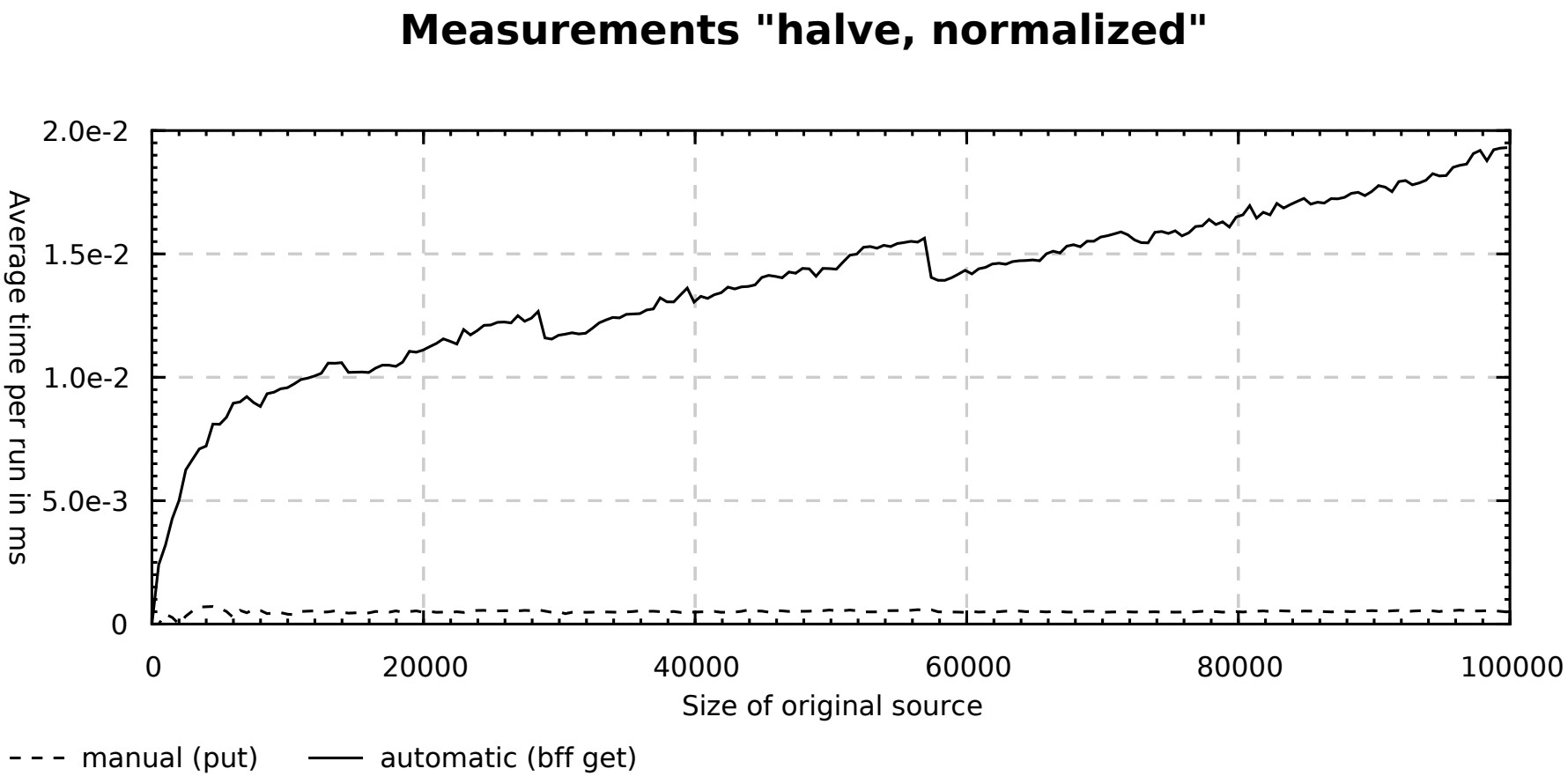
[V., POPL'09]

Semantic Bidirectionalization (Complete, for Lists)

```
put :: Eq α ⇒ [α] → [α] ⇀ [α]
put [] []                 = []
put s  v' | (s /= [])     =
  let t  = [0..((length s) − 1)]
      g  = IntMap.fromDistinctAscList (zip t s)
      h  = assoc (get t) v'
      h' = IntMap.union h g
  in  map (λi → fromJust (IntMap.lookup i h')) t

assoc :: Eq α ⇒ [Int] → [α] ⇀ IntMap α
assoc []     []     = IntMap.empty
assoc (i : is) (b : bs) =
  let m = assoc is bs
  in  case IntMap.lookup i m of
        Nothing              → IntMap.insert i b m
        Just c | (b == c)    → m
```

Some Experiments Done



Many Questions

- Do we even care (yet) about efficiency issues?

Many Questions

- Do we even care (yet) about efficiency issues?
- What is it that we should measure/compare? Efficiency in terms of what?

Many Questions

- Do we even care (yet) about efficiency issues?
- What is it that we should measure/compare? Efficiency in terms of what?
- What are the sources of inefficiency?

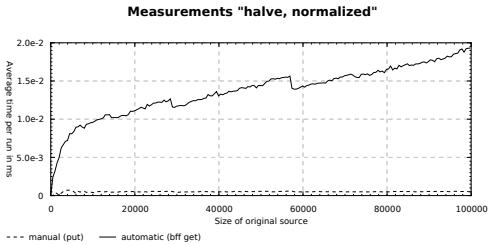
Many Questions

- Do we even care (yet) about efficiency issues?
- What is it that we should measure/compare? Efficiency in terms of what?
- What are the sources of inefficiency?
- How can we improve efficiency?

Many Questions

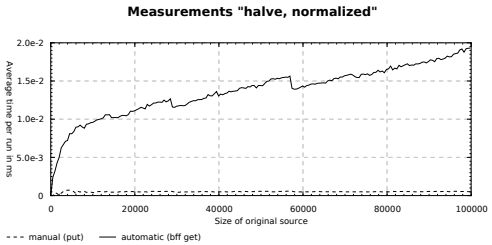
- Do we even care (yet) about efficiency issues?
- What is it that we should measure/compare? Efficiency in terms of what?
- What are the sources of inefficiency?
- How can we improve efficiency?
- Does it have side effects for qualitative/semantic issues?

What to Measure/Compare?



runtime of `put` v' s, variable $|s|$

What to Measure/Compare?

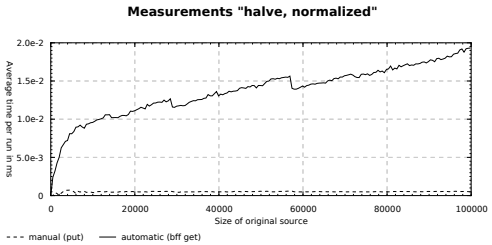


runtime of `put` v' s, variable $|s|$

Possible alternatives:

- measure whole roundtrips

What to Measure/Compare?

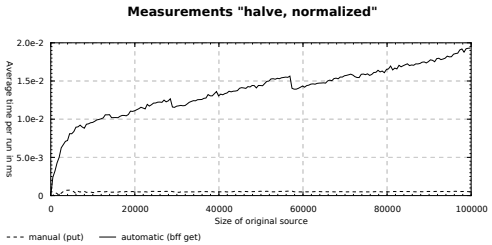


runtime of `put` v' s, variable $|s|$

Possible alternatives:

- measure whole roundtrips
- measure over histories, for incrementality

What to Measure/Compare?

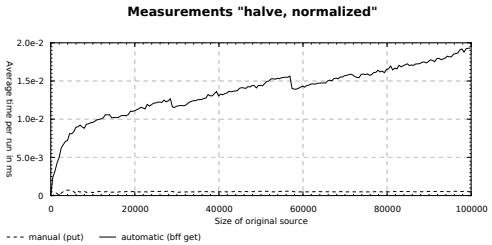


runtime of `put` v' s, variable $|s|$

Possible alternatives:

- measure whole roundtrips
- measure over histories, for incrementality
- account for more fine-grained cost division

What to Measure/Compare?

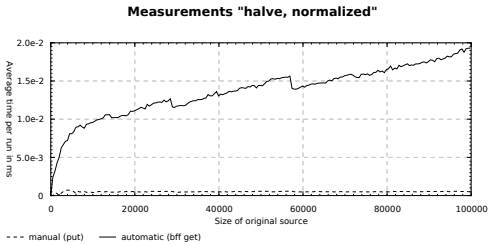


runtime of `put` v' s, variable $|s|$

Possible alternatives:

- measure whole roundtrips
- measure over histories, for incrementality
- account for more fine-grained cost division
- make v' variable as well

What to Measure/Compare?

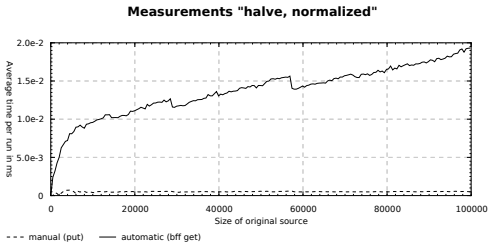


runtime of `put` v' s, variable $|s|$

Possible alternatives:

- measure whole roundtrips
- measure over histories, for incrementality
- account for more fine-grained cost division
- make v' variable as well
- express in terms of "update delta"

What to Measure/Compare?



runtime of `put` v' s, variable $|s|$

Possible alternatives:

- measure whole roundtrips
- measure over histories, for incrementality
- account for more fine-grained cost division
- make v' variable as well
- express in terms of "update delta"
- ???

Sources of Inefficiency?

- Syntactic Bidirectionalization:
  - explicit computation of complement
  - nondeterminism in syntactically inverted intermediate program

Sources of Inefficiency?

- Syntactic Bidirectionalization:
  - explicit computation of complement
  - nondeterminism in syntactically inverted intermediate program
- Semantic Bidirectionalization:
  - costly management of index map
  - a lot of abstraction overhead
  - lack of intensional knowledge about `get`

Sources of Inefficiency?

- Syntactic Bidirectionalization:
  - explicit computation of complement
  - nondeterminism in syntactically inverted intermediate program
- Semantic Bidirectionalization:
  - costly management of index map
  - a lot of abstraction overhead
  - lack of intensional knowledge about `get`
- Technique XYZ:
  - ???
  - ...

How to Improve Efficiency?

- Obviously, by removing sources of inefficiency. ☺

How to Improve Efficiency?

- Obviously, by removing sources of inefficiency. ☺
- Algorithm/data structure engineering?

Semantic Bidirectionalization (Complete, for Lists)

```
put :: Eq α ⇒ [α] → [α] ⇀ [α]
put [] []                 = []
put s  v' | (s /= []) =
  let t  = [0..((length s) − 1)]
      g  = IntMap.fromDistinctAscList (zip t s)
      h  = assoc (get t) v'
      h' = IntMap.union h g
  in  map (λi → fromJust (IntMap.lookup i h')) t

assoc :: Eq α ⇒ [Int] → [α] ⇀ IntMap α
assoc []     []     = IntMap.empty
assoc (i : is) (b : bs) =
  let m = assoc is bs
  in  case IntMap.lookup i m of
        Nothing              → IntMap.insert i b m
        Just c | (b == c) → m
```

How to Improve Efficiency?

- Obviously, by removing sources of inefficiency. ☺
- Algorithm/data structure engineering?

How to Improve Efficiency?

- Obviously, by removing sources of inefficiency. ☺
- Algorithm/data structure engineering?
- Partial application/evaluation:

$$\langle \texttt{get}, \texttt{put} \rangle :: S \to (V, V \rightharpoonup S)$$

Semantic Bidirectionalization (Complete, for Lists)

```
put :: Eq α ⇒ [α] → [α] ⇀ [α]
put [] []                 = []
put s  v' | (s /= [])     =
  let t  = [0..((length s) − 1)]
      g  = IntMap.fromDistinctAscList (zip t s)
      h  = assoc (get t) v'
      h' = IntMap.union h g
  in  map (λi → fromJust (IntMap.lookup i h')) t

assoc :: Eq α ⇒ [Int] → [α] ⇀ IntMap α
assoc []       []       = IntMap.empty
assoc (i : is) (b : bs) =
  let m = assoc is bs
  in  case IntMap.lookup i m of
        Nothing              → IntMap.insert i b m
        Just c | (b == c)    → m
```

How to Improve Efficiency?

- Obviously, by removing sources of inefficiency. ☺
- Algorithm/data structure engineering?
- Partial application/evaluation:

$$\langle \texttt{get}, \texttt{put} \rangle :: S \to (V, V \rightharpoonup S)$$

How to Improve Efficiency?

- Obviously, by removing sources of inefficiency. ☺
- Algorithm/data structure engineering?
- Partial application/evaluation:

$$\langle \texttt{get}, \texttt{put} \rangle :: S \to (V, V \rightharpoonup S)$$

- Inlining `get`, plus equational reasoning?

Semantic Bidirectionalization (Complete, for Lists)

```
put :: Eq α ⇒ [α] → [α] ⇀ [α]
put [] []                 = []
put s  v' | (s /= []) =
  let t  = [0..((length s) − 1)]
      g  = IntMap.fromDistinctAscList (zip t s)
      h  = assoc (get t) v'
      h' = IntMap.union h g
  in  map (λi → fromJust (IntMap.lookup i h')) t

assoc :: Eq α ⇒ [Int] → [α] ⇀ IntMap α
assoc []     []     = IntMap.empty
assoc (i : is) (b : bs) =
  let m = assoc is bs
  in  case IntMap.lookup i m of
        Nothing              → IntMap.insert i b m
        Just c | (b == c) → m
```

How to Improve Efficiency?

- Obviously, by removing sources of inefficiency. ☺
- Algorithm/data structure engineering?
- Partial application/evaluation:

$$\langle \texttt{get}, \texttt{put} \rangle :: S \to (V, V \rightharpoonup S)$$

- Inlining `get`, plus equational reasoning?
- More standard program transformations?
- ???

Many Questions

- Do we even care (yet) about efficiency issues?
- What is it that we should measure/compare? Efficiency in terms of what?
- What are the sources of inefficiency?
- How can we improve efficiency?
- Does it have side effects for qualitative/semantic issues?

References

K. Matsuda, Z. Hu, K. Nakano, M. Hamana, and M. Takeichi.
Bidirectionalization transformation based on automatic derivation of view complement functions.
In *International Conference on Functional Programming, Proceedings*, pages 47–58. ACM Press, 2007.

J. Voigtländer.
Bidirectionalization for free!
In *Principles of Programming Languages, Proceedings*, pages 165–176. ACM Press, 2009.