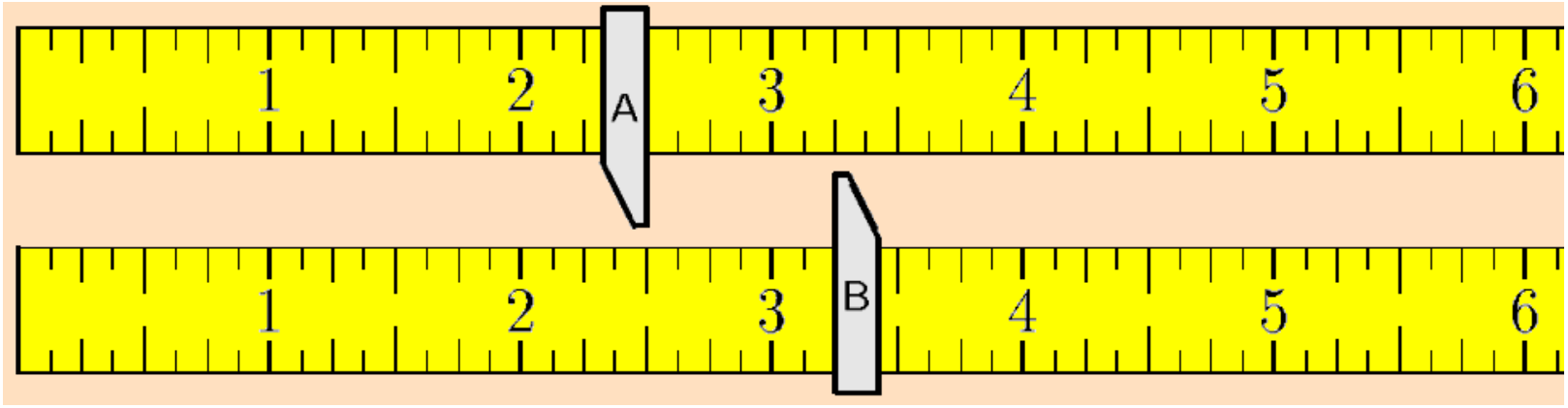
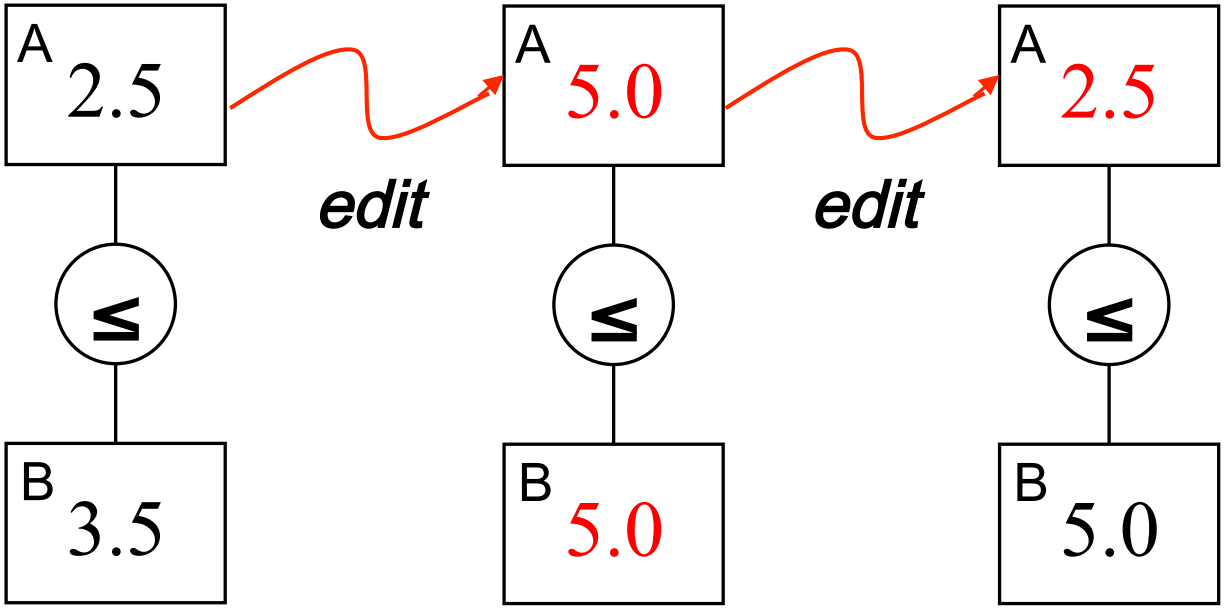
Incremental Updates and Non-Free Datatypes

Jeremy Gibbons, University of Oxford
(joint work with the TLCBX team)
BX at BIRS, Banff, December 2013

1. Least-change principle (Meertens)

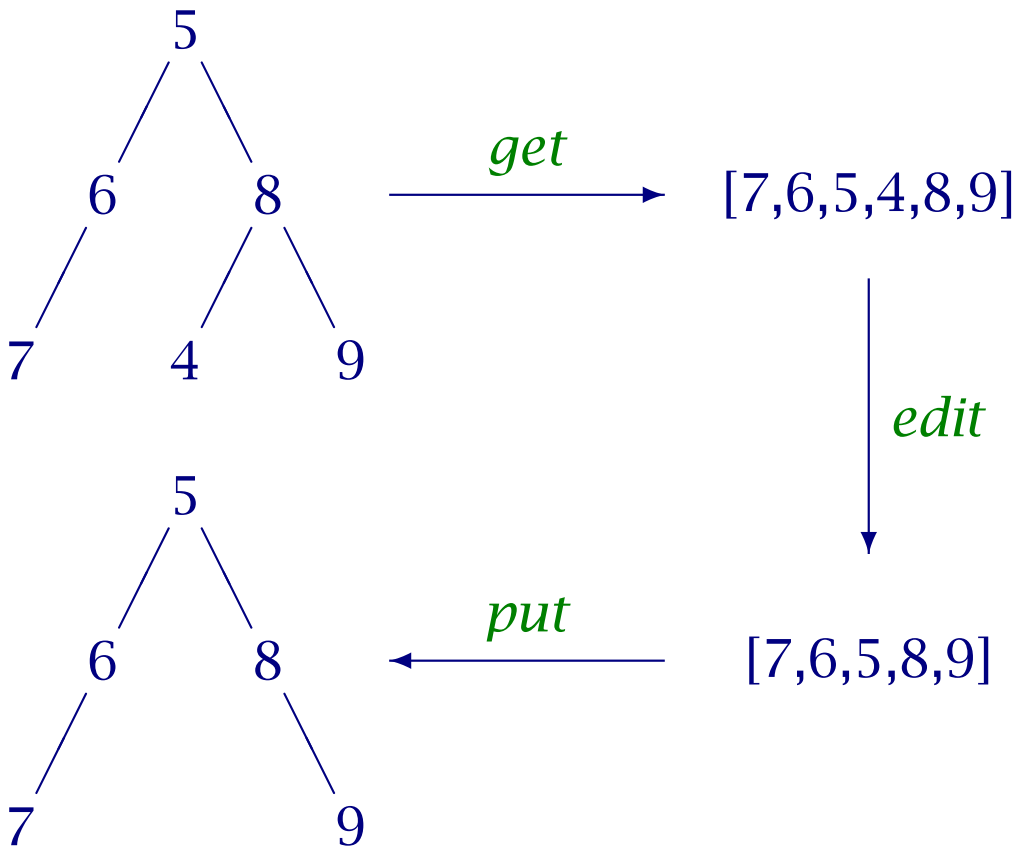




2. Incremental updates (Wang et al, ICFP 2011)

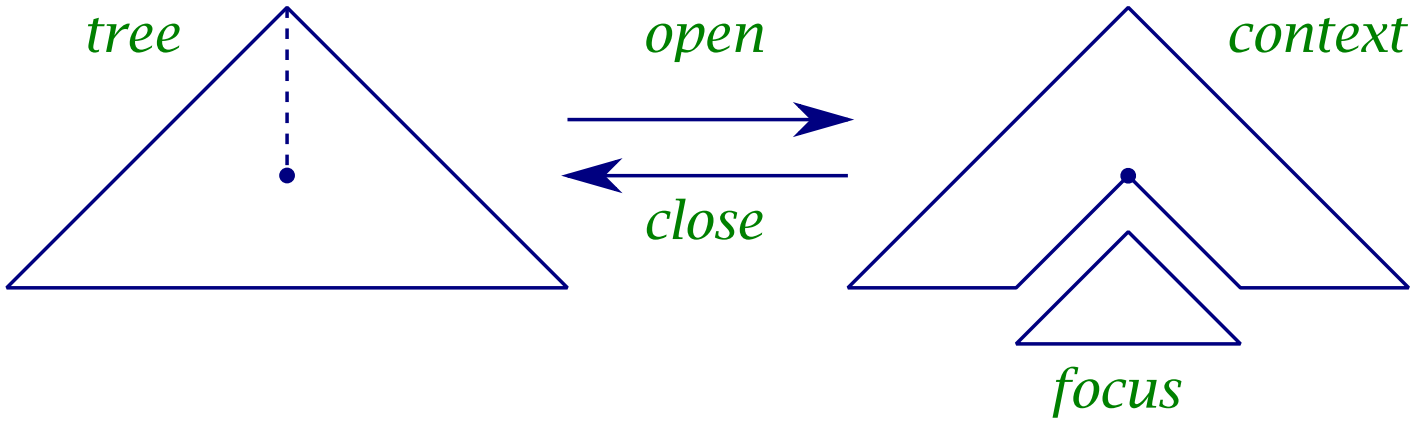
- exploit any *locality* in edits
- translate into *incrementality* of updates
- hopefully, small edits entail small updates

2.1. A small example



2.2. Context–focus treatment of trees

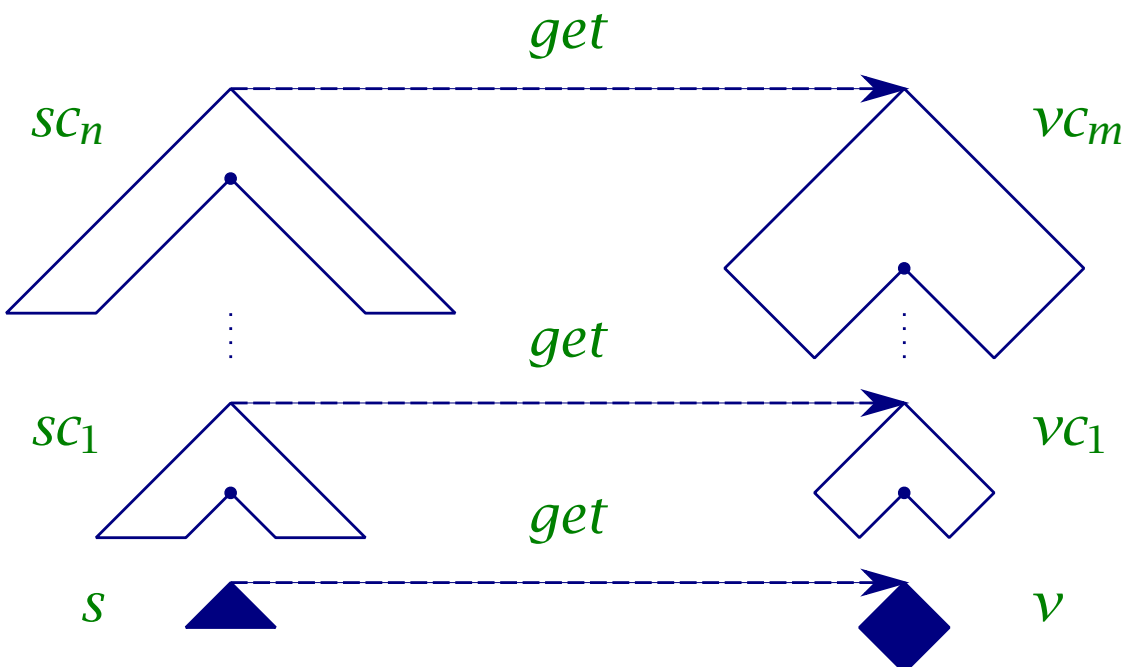
Zipper-style:



- trees as algebraic datatypes
- structural recursion (fold) for *get* functions
- ‘regions’ are subterms

2.3. Source–view alignment

For any view position v, the *get* function induces an alignment between ancestors of v and corresponding source positions:



The alignment captures locality preservation. There is always such an alignment, but it may be more or less useful: in the worst case, all ancestors of v align with the root of the source.

2.4. Limitations

- works ok for balanced trees, but not for skewed ones
- works particularly badly for ‘cons’ lists
- ad hoc approach for ‘cat’ lists—see ICFP paper
- what about graphs? kind of important

3. Non-free datatypes

The problem comes from viewing datatypes as algebraic:
unique decomposition of each data structure into components.

That's too rigid. A data structure might have overlapping 'parts'.
Consider cat lists, for example; and graphs.

If such datatypes are generated from constructors, then the constructors generally obey non-trivial laws:

- associativity for lists
- abiding properties for matrices
- commutativity of juxtaposition for graphs

It's hard to fit these laws into the algebraic view.

Still, algebra is very powerful, and we don't want to 'throw out the baby with the bath water'.

3.1. Container datatypes

An alternative treatment of datatypes (Abbott et al): a datatype is

- a set S of shapes
- for each $s : S$, a set $P(s)$ of positions

A particular data structure of this type, with elements of type X, is a pair (s, f) where $s : S$ specifies the shape and $f : P(s) \to X$ the element at each position in that shape.

For example, lists have $S = \mathbb{N}$ and $P(n) = \{1..n\}$.

3.2. Properties of container datatypes

Container datatypes have nice algebraic structure too:

- closed under products, coproducts and composition
- initial algebras and final coalgebras
- natural transformations rearrange positions (independent of elements)
- a notion of subterm, for *directed containers* (every position determines a substructure)

Crucially, they provide a nice handle for datatypes with laws.
In particular, the laws we are interested in are generally about shapes and positions, *but not about elements*.

3.3. Quotiented datatypes

Often, in fact, the laws can be expressed as quotients on the sets of positions. For example, bags are like lists, but quotiented by permutations on the positions.

Sets of shapes quotiented by an equivalence relation have been studied by mathematicians for years, in enumerative combinatorics.

Typical questions are: how many ordered binary trees are there on n elements? How many unordered binary trees? How many cyclic lists?

Ordered binary trees are expressible algebraically, but unordered trees and cyclic lists require a quotient.

3.4. Combinatorial species

Joyal (1981) established combinatorial species as a framework for enumerative combinatorics. See also Yorgey (2010).

Technically, a species $\mathsf{F} = (\mathsf{F}^\bullet, \mathsf{F}^\leftrightarrow)$ is

- a mapping $\mathsf{F}^\bullet$ that takes a finite set U of labels to a finite set $\mathsf{F}^\bullet(U)$ of 'F-structures'
- a mapping $\mathsf{F}^\leftrightarrow$ that lifts any bijection $\sigma : U \leftrightarrow V$ between label sets U and V (a 'relabelling') to a bijection $\mathsf{F}^\leftrightarrow(\sigma) : \mathsf{F}^\bullet(U) \leftrightarrow \mathsf{F}^\bullet(V)$ between F-structures
- this lifting should respect identity and composition of relabellings.

(Or: an endofunctor on the category $\mathbb{B}$ of finite sets and bijections.)

$\mathsf{F}^\bullet(U)$ corresponds to 'the set of structures with labels from U'.
Lifting of bijections means that the actual labels don't matter: we might as well use $\{1..|U|\}$ as U itself.

3.5. Polynomial species

Empty, unit, singleton, sum, product, composition work in the obvious way.

- $\mathsf{0}^\bullet(U) = \varnothing$
- $\mathsf{1}^\bullet(\varnothing) = \{*\}$, and $\mathsf{1}^\bullet(U) = \varnothing$ otherwise
- $\mathsf{X}^\bullet(U) = \{*\}$ for $|U| = 1$, and $\mathsf{X}^\bullet(U) = \varnothing$ otherwise
- $(\mathsf{F} + \mathsf{G})^\bullet(U) = \mathsf{F}^\bullet(U) \uplus \mathsf{G}^\bullet(U)$
- $(\mathsf{F} \times \mathsf{G})^\bullet(U) = \Sigma_{U=V \uplus W} \, (\mathsf{F}^\bullet(V) \times \mathsf{G}^\bullet(W))$
- $(\mathsf{F} \circ \mathsf{G})^\bullet(U) = \Sigma_{U=U_1 \uplus \cdots \uplus U_n} \, (\mathsf{F}^\bullet(\mathsf{G}^\bullet(U_1)) \times \cdots \times \mathsf{F}^\bullet(\mathsf{G}^\bullet(U_n)))$
  (provided that $G^\bullet(\varnothing) = \varnothing$, to remain finite)

3.6. Regular species

Least fixpoints work, so all regular datatypes correspond to species.

For example,

$$\mathsf{L} = \mu Z \cdot 1 + \mathsf{X} \times Z$$

yields the species of lists:

$$\mathsf{L}^\bullet(n) = \{1..n!\}$$

That is, there are $n!$ lists on n elements.

A mathematician would write

$$\mathsf{L} = 1 + \mathsf{X} \times \mathsf{L}$$

and exploit the *implicit species theorem*:

> A recursive equation $\mathsf{F} = \phi(\mathsf{F})$ has a least fixed point,
> provided that $\phi(\varnothing) = \varnothing$ and ϕ is 'guarded'.

3.7. Non-regular species

But we are not limited to regular datatypes: more exotic datatypes too.

For bags, $\mathsf{B}^\bullet(n)$ is (any) singleton set, as there is only one way of making a bag out of n elements. Bags of size two are simply a restriction of bags: ${\mathsf{B}_2}^\bullet(U)$ is $\{*\}$ if $|U| = 2$ and $\varnothing$ otherwise.

Subbags $\mathcal{P}$ can be expressed as $\mathsf{B} \times \mathsf{B}$, a partition of the labels into a subbag and its complement.

So simple graphs (undirected, no self loops) are $\mathcal{P} \circ (\mathsf{B}_2 \times \mathsf{B})$, sets of two-sets (of edges).

3.8. My conjecture

Quotiented directed containers will provide a fruitful framework for the study of BX on non-free datatypes, especially graphs.