Sets and Bags

- **Sets**
  - set(2,5,5,4): enumerated set  // {2,5,4}
  - set(expr: param in domain; pred)  // domain: set, bag, seq

- For set x
  - x.size: # of entries in x
  - x.add(m)  // x ← x ∪ {m}
  - x.remove(m)  // x ← x \ {m}

- **Set types**
  - Set x
  - Set<U> x  // set of entries of U

- **Bags**  // multisets
  - all the above constructs, with “set” → “bag”
  - e.g., bag(expr: param in domain; pred)
Sequences

- **Sequences**
  - \([2,3,4,2,1]\): enumerated sequence \(\text{// [head, ..., last]}\)
  - \([\text{expr: param in domain; pred}]\) \(\text{// domain: sequence}\)

- **For sequence** \(x\)
  - \(x[j]\): \(j^{th}\) entry \(\text{// x[0] is head}\)
  - \(x\).keys: \([0..x\text{.size-1}]\)
  - \(x\).append(m) \(\text{// to tail}\)
  - \(x\).remove(k) \(\text{// x[k]}\)

- \(\circ\): concatenation \(\text{// [1,2]\circ[a,b] = [1,2,a,b]}\)

- **Sequence types**
  - Seq
  - Seq<\(U\)> \(\text{// entries in U}\)

- **Tuples**: fixed-length seqs
  - Tuple<\(\cdot,\cdot\)>
  - Tuple<\(U,V\)> \(\text{// U \times V}\)
Maps

- Map
  - set of [key, value] tuples, with distinct keys
  - ```map([2,100], [3,200])``` // map with 2 entries
  - ```map(2tuple: param in domain; pred)```

- For map x
  - ```x.keys``` // sequence of keys
  - ```x[j]``` // value in [j, ·]
  - ```remove(j)``` // delete [j, ·] (if any)
  - ```x[j] ← e``` // remove(j), add [j,e]

- Map types:
  - Map
  - Map <U, V>
Set/sequence $S$ can serve as a “type” for defining vars
- $S \ x$: var $x$ can range over current values of $S$

Type $T$ can serve as a “set” for membership predicates
- $x \ in \ T$
- $T(x)$

Don’t-care value “·” in predicate $P$
- $(\text{thread in fn(.)}: \ \text{forsome}(x: \ \text{thread in fn}(x)))$
- $(\text{thread in v[.].fn(.)}: \ \text{forsome}(x,y: \ \text{thread in v[y].fn}(x)))$
- “forsome” applies to smallest predicate in $P$ enclosing ·

ongoing($S$): short for “(thread in $S$)”