Conventions

Shankar

March 11, 2014

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)` // $x \leftarrow x \cup \{m\}$
  - `x.remove(m)` // $x \leftarrow x \setminus \{m\}$
- 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 // [head, ..., last]
  - `[`*expr*`:` *param* `in` *domain*`;` *pred*`]` // *domain*: sequence
- For sequence `x`
  - `x[j]`: jth entry // `x[0]` is head
  - `x.keys`: `[0..x.size - 1]`
  - `x.append(m)` // to tail
  - `x.remove(k)` // `x[k]`
- $\circ$: concatenation // `[1,2]` $\circ$ `[a,b]` $=$ `[1,2,a,b]`
- Sequence types
  - `Seq`
  - `Seq<U>` // entries in `U`
- Tuples: fixed-length seqs
  - `Tuple<`$\cdot$`,`$\cdot$`>`
  - `Tuple<U,V>` // $U \times V$

Maps

- Map
  - set of [key, value] tuples, with distinct keys
  - `map([2,100], [3,200])` // map with 2 entries
  - `map(`2*tuple*`:` *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>`

Miscellaneous

- 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*
  - `(thread in fn(.)`: `forsome(x: thread in fn(x))`
  - `(thread in v[.].fn(.)`: `forsome(x,y: thread in v[y].fn(x))`
  - “`forsome`” applies to smallest predicate in *P* enclosing ·
- `ongoing(S)`: short for “`(thread in S)`”