Diagram 4.4-E. Constant pool relations New items abbreviated Anchor, Linkage are fully spelled CONSTANT SpecializationAnchor, CONSTANT SpecializationLinkage Anchor*, Linkage* Dynamic, *new InvokeDynamic parameter bootstrap arguments reference bootstrap method "Loadable Constant" MethodHandle existing 'API Point concept Name"* (see Diagram 4.4-G) *new concept String (see Diagram 4.4-F) string reference Fieldref. MethodType class **XMethodref** descriptor name and type "API Point Reference" Module, *new concept Package NameAndType name name descriptor Class name. Utf8

Diagram 4.4-F(a). API Point relations Note: A Linkage constant is usable anywhere its reference item may be used. invokeX getX, putX, ldc. new ClassFile withfield defaultvalue structure Xnewarray operand super class (field or instanceof interfaces method) checkcast operand (class or MethodHandle interface) refekence "API Point Reference' *new concept Linkage* "API Point *new Name"* reference *new concept parameter Fieldref, Class "Loadable XMethodref Constant" class (recursive) 'API Point Reference' (recursive, to (see Diagram 4.4-G) class only)

Diagram 4.4-F(b). "API Point References", all configurations (These are also "API Point Names", except Linkage constants.) species Linkage* reference specialized specialized plain class method method (in class) (in species) Class Class Linkage* Linkage* reference reference species member plain member Fieldref, Fieldref. XMethodref **XMethodref XMethodref** XMethodref class class class class Linkage* Linkage* reference reference Class Class Class Class

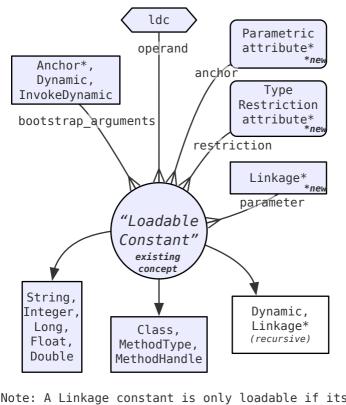


Diagram 4.4-G. "Loadable Constant" relations

Note: A Linkage constant is only loadable if its reference is loadable, i.e., a Class. Other Linkage constants (e.g., of a Fieldref) are not loadable.

non-parametric client of List<Point> (resolution states are at right; all are invariant) IMethodref Method ptr, Anchor, TR [P].get(int) NameAndType operand get(int)Obj (corresponding (CP entries) CP entry states) invokeinterface Linkage Anchor, List[Point] **Species** Class List.class List Utf8 i/u/List Class Point.class Point Utf8 pkg/Point

Diagram 4.4-H(a). Example constant pool:

```
Diagram 4.4-H(b). Example parametric interface
     public interface List<T> { ...get... }
      Linkage
     List[R.T]
                                  ClassFile
       Class
                                  this = i/u/List
        list
                                  interface = Collection[R.T]
        Utf8
                                  Parametric = Anchor R
      j/u/List
                                  (self species reference, if anv)
                                  method info
                                  name = sublist
                                  type = (int, int)List
                                  TypeRestriction = {List[R.T]}
                                  Parametric = Anchor R
       ConDy
                                  Code = none (ACC ABSTRACT)
        R.T
                                  method info
      Anchor R
                                  name = get
       BSM=...
                                  type = (int)0bject
                                  TypeRestriction = \{R.T\}
                                  Parametric = Anchor R
                                  Code = none (ACC ABSTRACT)
```

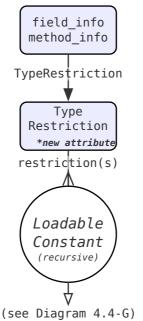
```
Diagram 4.4-H(c). Example parametric implementation
     class ArrayList<T> implements List<T> { ...get... }
      Linkage
    A...vList[R.T]
                                  ClassFile
       Class
                                  this = i/u/ArravList
     ArravList
                                  interface = i/u/List[R.T]
        IItf8
                                  Parametric = Anchor R
   i/u/ArravList
                                   (self species reference, if any)
                                  method info
      Linkage
                                  name = subList
     List[R.T]
                                  type = (int, int)List
       Class
                                  TypeRestriction = {List[R.T]}
        List
                                  Parametric = Anchor R
                                  Code = stuff using elements/A
        Utf8
      i/u/List
                                  method info
                                  name = qet
                                  type = (int)0bject
                                  TvpeRestriction = \{R,T\}
                                  Parametric = Anchor R
                                  Code = stuff using elements/A
       ConDy
                                  field info
        R.T
                                  name = elements
      Anchor R
                                  type = Object[]
       BSM=...
                                  TypeRestriction = \{R.T[]\}
                                  Parametric = Anchor R
       ConDy
       R.T[]
```

```
Diagram 4.4-H(d). Example parametric subclass
  class MyVector<T> extends ju.Vector<T> { ...get... }
  class Vector<T> { ... protected T[] elementData; ... }
      Linkage
    MvVector[T]
                                  ClassFile
       Class
                                  this = MvVector
      MvVector
                                  super = i/u/Vector[R.T]
        IItf8
                                  Parametric = Anchor R
      MvVector
                                  (self species reference, if any)
      Linkage
    Vector[R.T]
       Class
                                  method info
       Vector
                                  name = get
                                  type = (int)0bject
        Utf8
                                  TvpeRestriction = \{R,T\}
     i/u/Vector
                                  Parametric = Anchor R
                                  Code = (invariants, parametrics)
                                            (bytecodes...)
       ConDy
                                   getfield
        R.T
                                   name = elementData
      Anchor R
                                   type = Object[]
       BSM=...
                                   class = Vector[R.T]
      Fieldref
    V[T].el-Data
                                   possible type restriction⋅⋅⋅→T[]
                                   iload 1 aaload
    NameAndType
   elmData:Obj[]
```

Diagram 4.7-D(a). Parametric attribute relations ClassFile field_info method info Parametric Parametric Parametric* *new attribute anchor

Anchor*

Diagram 4.7-D(b). TypeRestriction attribute relations



```
Diagram 4.4-H(d). Example parametric subclass
  class MyVector<T> extends ju.Vector<T> { ...get... }
  class Vector<T> { ... protected T[] elementData; ... }
      Linkage
    MvVector[T]
                                  ClassFile
       Class
                                  this = MvVector
      MvVector
                                  super = i/u/Vector[R.T]
        IItf8
                                  Parametric = Anchor R
      MvVector
                                  (self species reference, if any)
      Linkage
    Vector[R.T]
       Class
                                  method info
       Vector
                                  name = get
                                  type = (int)0bject
        Utf8
                                  TvpeRestriction = \{R,T\}
     i/u/Vector
                                  Parametric = Anchor R
                                  Code = (invariants, parametrics)
                                            (bytecodes...)
       ConDy
                                   getfield
        R.T
                                   name = elementData
      Anchor R
                                   type = Object[]
       BSM=...
                                   class = Vector[R.T]
      Fieldref
    V[T].el-Data
                                   possible type restriction⋅⋅⋅→T[]
                                   iload 1 aaload
    NameAndType
   elmData:Obj[]
```

to support the "Parametric VM".

Legend:

• A rectangular box shows one or more constant types. For example, a box labeled class represents a constant class info structure.

Graph of existing and proposed relations among constant pool structures,

example, string depends directly on the utf8 which specifies its characters.

• Arrows from boxes are labeled to show which item in a given

An arrow shows how one entity depends directly on another. For

- constant pool structure provides the index for the relation indicated by the arrow.

 A circle shows, not a single constant type, but a conceptual group
- of types, which are (for some uses) interchangeable. The conceptual groups are *Loadable Constant*, *API Point Reference*, and *API Point Name*.
- Box-headed arrows into a group circle show users of all the types in the group, while arrows out of the circle indicate the group's types. Small dotted arrows show some routes through a group.
 A starred footnote of *new* indicates a proposed new constant type
- or conceptual group of types. The new types are

 SpecializationAnchor and SpecializationLinkage, or Anchor and
 Linkage for short. The new conceptual groups are API Point
- Reference and (a subset) API Point Name.
 An arrow with a blank head redirects to a different diagram.
 Recursion points are white, and stand in place of the corresponding
- Recursion points are white, and stand in place of the corresponding colored concept or constant type.
 A lozenge shaped box shows one or more instruction types
 - A lozenge shaped box shows one or more instruction types.
 A box with rounded corners shows some other structure, such as the new Parametric and TypeRestriction attributes.