18. Hash Tables

18.1 Hash Table Concepts

18.1.1 Hash-Table Operations

The next figure lists some *defined names* that are applicable to *hash tables*. The following rules apply to *hash tables*.

-- A *hash table* can only associate one value with a given key. If an attempt is made to add a second value for a given key, the second value will replace the first. Thus, adding a value to a *hash table* is a destructive operation; the *hash table* is modified.
-- There are four kinds of *hash tables*: those whose keys are compared with **eq**, those whose keys are compared with **eql**, those whose keys are compared with **equal**, and those whose keys are compared with **equalp**.
-- *Hash tables* are created by **make-hash-table**. **gethash** is used to look up a key and find the associated value. New entries are added to *hash tables* using **setf** with **gethash**. **remhash** is used to remove an entry. For example:

```
(setq a (make-hash-table)) =>  #<HASH-TABLE EQL 0/120 32536573>
(setf (gethash 'color a) 'brown) =>  BROWN
(setf (gethash 'name a) 'fred) =>  FRED
(gethash 'color a) =>  BROWN, true
(gethash 'name a) =>  FRED, true
(gethash 'pointy a) =>  NIL, false
```

In this example, the symbols `color` and `name` are being used as keys, and the symbols `brown` and `fred` are being used as the associated values. The *hash table* has two items in it, one of which associates from `color` to `brown`, and the other of which associates from `name` to `fred`.

-- A key or a value may be any *object*.
-- The existence of an entry in the *hash table* can be determined from the *secondary value* returned by **gethash**.

```
clrhash           hash-table-p     remhash
gethash           make-hash-table  sxhash
hash-table-count  maphash
```

Figure 18-1. Hash-table defined names

18.1.2 Modifying Hash Table Keys

The function supplied as the `:test` argument to **make-hash-table** specifies the 'equivalence test' for the *hash table* it creates.

An *object* is 'visibly modified' with regard to an equivalence test if there exists some set of *objects* (or potential *objects*) which are equivalent to the *object* before the modification but are no longer equivalent afterwards.

If an *object* O1 is used as a key in a *hash table* H and is then visibly modified with regard to the equivalence test of H, then the consequences are unspecified if O1, or any *object* O2 equivalent to O1 under the equivalence test (either before or after the modification), is used as a key in further operations on H. The consequences of using O1 as a key are unspecified even if O1 is visibly modified and then later modified again in such a way as to undo the visible modification.

Following are specifications of the modifications which are visible to the equivalence tests which must be supported by *hash tables*. The modifications are described in terms of modification of components, and are defined recursively. Visible modifications of components of the *object* are visible modifications of the *object*.

18.1.2.1 Visible Modification of Objects with respect to EQ and EQL

No *standardized function* is provided that is capable of visibly modifying an *object* with regard to **eq** or **eql**.

18.1.2.2 Visible Modification of Objects with respect to EQUAL

As a consequence of the behavior for **equal**, the rules for visible modification of *objects* not explicitly mentioned in this section are inherited from those in Section 18.1.2.1 (Visible Modification of Objects with respect to EQ and EQL).

18.1.2.2.1 Visible Modification of Conses with respect to EQUAL

Any visible change to the *car* or the *cdr* of a *cons* is considered a visible modification with regard to **equal**.

18.1.2.2.2 Visible Modification of Bit Vectors and Strings with respect to EQUAL

For a *vector* of *type* **bit-vector** or of *type* **string**, any visible change to an *active element* of the *vector*, or to the *length* of the *vector* (if it is *actually adjustable* or has a *fill pointer*) is considered a visible modification with regard to **equal**.

18.1.2.3 Visible Modification of Objects with respect to EQUALP

As a consequence of the behavior for **equalp**, the rules for visible modification of *objects* not explicitly mentioned in this section are inherited from those in Section 18.1.2.2 (Visible Modification of Objects with respect to EQUAL).

18.1.2.3.1 Visible Modification of Structures with respect to EQUALP

Any visible change to a *slot* of a *structure* is considered a visible modification with regard to **equalp**.

18.1.2.3.2 Visible Modification of Arrays with respect to EQUALP

In an *array*, any visible change to an *active element*, to the *fill pointer* (if the *array* can and does have one), or to the *dimensions* (if the *array* is *actually adjustable*) is considered a visible modification with regard to **equalp**.

18.1.2.3.3 Visible Modification of Hash Tables with respect to EQUALP

In a *hash table*, any visible change to the count of entries in the *hash table*, to the keys, or to the values associated with the keys is considered a visible modification with regard to **equalp**.

Note that the visibility of modifications to the keys depends on the equivalence test of the *hash table*, not on the specification of **equalp**.

18.1.2.4 Visible Modifications by Language Extensions

Implementations that extend the language by providing additional mutator functions (or additional behavior for existing mutator functions) must document how the use of these extensions interacts with equivalence tests and *hash table* searches.

Implementations that extend the language by defining additional acceptable equivalence tests for *hash tables* (allowing additional values for the `:test` argument to **make-hash-table**) must document the visible components of these tests.