Prototype Verification System

Lecture Notes

Daniele Parmeggiani

February 29, 2024

Contents

Chapter 1

Introduction

In this chapter we will introduce PVS (the Prototype Verification System),[1] and we will install a modern environment for using PVS. Later, we'll also see how to prove some simple properties of a hashmap using PVS.

1.1 Introduction to PVS

PVS is a powerful tool designed for formal specification and verification of software and hardware systems. It is a language coupled with a complete environment for developing formal mathematical specifications and then verifying that those meet certain properties.

PVS comes with an expressive specification language that allows you to describe your systems in a precise and formal manner. One of the core functionalities of PVS is its ability to prove theorems about your specifications. This involves using mathematical reasoning to demonstrate that certain properties hold true for your system under specified conditions.

PVS includes a set of built-in libraries that cover a wide range of mathematical theories and common programming constructs. These libraries provide a foundation for specifying and verifying a variety of systems. Notably among them, we can find NASALib,[2] a vast set of libraries developed by the Formal Methods teams at the NASA Langley Research Center, along with the Stanford Research Institute, the National Institute of Aerospace, and the larger PVS community.

PVS is often used to verify critical software systems where correctness is of utmost importance, such as avionics and aerospace software, medical devices, or other safety-critical applications.

[1]`https://pvs.csl.sri.com/`

[2]`https://github.com/nasa/pvslib`

It is also extensively employed in the verification of hardware systems, and can be used to formally specify and verify communication protocols, cryptographic and concurrent algorithms, and other complex systems.

1.2 Installing PVS

The recommended development environment for PVS is NASA's plugin for Visual Studio Code:[3]

1. download and install Visual Studio Code for your platform from `https://code.visualstudio.com/download`;
2. navigate to the extensions tab in VS Code (the blocks icon at the left edge of the screen) and search for "pvs";
3. the first item in the list should be called "PVS" and authored by Paolo Masci;
4. click the install button and follow the on-screen installation instructions.

Note that this procedure installs the PVS binaries and libraries in your system as well, at a path that you need to specify.

To confirm that everything was installed correctly, open a new VS Code workspace; i.e. create and select a folder from your file system. Then, navigate to the newly appeared PVS tab at the left edge of the screen, right-click on the folder's name and select "New PVS File." In the new file insert the following content:

```
hello: THEORY
  BEGIN

  world: CONJECTURE
    TRUE
END hello
```

Now, right-click the name of the new file on the left panel and select "Rerun All Proofs:" you should see a new tab called `hello.summary` displaying the above conjecture `world` with the status `proved`.

For further help installing this plugin, please refer to `https://github.com/nasa/vscode-pvs#installation-instructions`.

[3]`https://github.com/nasa/vscode-pvs`

1.3 Proving Properties of a Hashmap

Let us introduce the core concepts and features of the PVS proof assistant with a commonly known data structure. We will try to prove a hashmap has the following features:

1. it should be possible to insert a new (key, value) pair;
2. it should be possible, given a key, to retrieve its corresponding value; and
3. it should be possible to remove a (key, value) pair from the hashmap.

Let's name these operations *insert*, *lookup*, and *delete*. Since we'll be talking about keys, values and maps, we should let PVS know about them. Specifically, we introduce new *uninterpreted* types; i.e. types that PVS knows nothing about: whether there would be 0, 1, or infinite instances of these types, for example. PVS instead is thus only told that a key `K` is something different from a value `V`.

```
K: TYPE
V: TYPE
```

Now we define what a hashmap is, and we use this notation:

```
M: TYPE = [K -> V]
```

This tells PVS that a hashmap `M` is a function from keys to values. We will later define this function as a sequence of associations. You can also think of this `TYPE` as an array of (key, value) pairs.[4]

We could define `M` as a partial function (that is, a function defined over a subset of its domain), but efficient theorem proving strongly encourages the use of total functions. One way to denote the fact that some $k \in K$ is not to be found in the hashmap, is to identify some particular value that indicates this. Let's choose an arbitrary element of `V`:

```
null: V
```

So that, we will expect a hashmap to be empty iff $\forall k \in K, m(k) = \texttt{null}$. We can write this in PVS too:

[4]PVS also allows expressing exactly *an array of tuples*, but it's able to reason very efficiently with functions, which is why we choose this representation.

```
empty: M
empty_ax: AXIOM
    FORALL (k: K): empty(k) = null
```

We have declared a hashmap to be the *one* empty hashmap, and we have axiomatically defined `empty` as the hashmap in which all keys point to `null`. This may surprise a programmer who's used to think that there may be multiple instances of a hashmap class that all happen to be empty. From the point of view of PVS, though, all those instances are exactly the same. A program may distinguish them based on their memory address, but PVS has no notion of this, instead it only knows abstractly about the types we have defined so far. Think of it mathematically instead of programmatically: the number 0 is exactly the number 0 no matter how many times we may write it on a piece of paper.

Also, do not consider the axiom itself to be computationally expensive since it associates `null` to the entire key space. Quite the contrary: in the carrying out of a proof, PVS will know that whatever we may write next, the value of any key in `empty` will be `null`, without employing any memory to store the possible associations.

Now, let's "implement" the hashmap operations:

```
lookup: [M, K -> V]
lookup_ax: AXIOM FORALL (m: M) (k: K):
    lookup(m, k) = m(k)

insert: [M, K, V -> M]
insert_ax: AXIOM FORALL (m: M) (k: K) (v: V):
    insert(m, k, v) = m WITH [(k) := v]

delete: [M, K -> M]
delete_ax: AXIOM FORALL (m: M) (k: K):
    delete(m, k) = m WITH [(k) := null]
```

Notice some of the notations we have used so far: `m(k)` is exactly equivalent to the mathematical notation of function application, the `AXIOM` and `FORALL` keywords refer to their usual meanings, and the keyword `WITH` has been used to add an association to the function `m` above.

Also, notice the *functional* specification style: we pass the whole "state of the system" (i.e. the entire hashmap) to the functions `lookup`, `insert`, and `delete`. Consider `insert`: the input hashmap is somehow discarded in the operation and we instead "return" a novel hashmap that has the added association. As previously said, do not consider that PVS will actually allocate and deallocate memory to represent these hashmaps.

If we were programming a hashmap and we felt content of what we wrote, we may turn to writing tests. But with PVS we instead want to write theorems about our hashmaps: they are general statements that we think should be true if our specification does what it ought to. This can provide much more information than a single test case, since with one theorem we would be running an entire *class* of test cases.

A suitable challenge for our specification may be: "if I add a key `k` with a value `v` to a hashmap and then I lookup `k`, I should get back `v`." We can write it as:

```
insert_then_lookup: CONJECTURE FORALL (m: M) (k: K) (v: V):
    lookup(insert(m, k, v), k) = v
```

In your VS Code editor, you should see some links above the definition of `insert_then_lookup`. We invoke the prover by clicking "prove":

```
Starting prover session for insert_then_lookup


insert_then_lookup :

|-------
{1}   FORALL (m: M) (k: K) (v: V): lookup(insert(m, k, v), k) =
v

>>
```

This is a *sequent*: in general there will be several numbered formulas above the turnstile symbol `|-------` (antecedent), and several below (consequent). The idea is that we have to establish that the conjunction (and) of formulas in the antecedent implies the disjunction (or) of the formulas in the consequent. Therefore, a sequent is true if any antecedent is the same as any consequent, if any antecedent is false, or if any consequent is true.

The prompt indicates that PVS is waiting for us to submit a prover command, in Lisp syntax. The commands provided by PVS can be categorized as basic commands and strategies, which in turn are compositions of basic commands. Let us disregard enumerating all of the various commands available and their semantics during this introduction (they will be explored later in §???) and let us instead introduce a few commands and consider how a proof with PVS can be carried out in general.

The highest-level strategy is called `grind`. It does skolemization, heuristic instantiation, propositional simplification, if-lifting, rewriting, and applies decision procedures for linear arithmetic and equality. It can take several optional arguments to specify the formulas that can be used for automatic rewriting (below, we'll be specifying that it can use the axioms in the theory `hashmap`).

Following the execution of a command, PVS may be required to split the proof into two or more parts. The PVS prover itself maintains a proof tree; the act of proving the specification can be thought of as traversing the tree, with the goal of showing that each leaf of the tree is true.

The `grind` strategy satisfies the proof for `insert_then_lookup`:

```
>> (grind :theories ("hashmap"))

Q.E.D.
```

Following the display of `Q.E.D.`, you should see a pop-up indicating that the proof has been saved. We will mostly run our proofs in the interactive environment, but after a proof is successfully completed, it can be re-run in batch mode, without user interactions. Interactive proofs can also be suspended with `(quit)` and later resumed.

In the example shown above the proof tree had just one trivial branch. Now let's turn to another less trivial conjecture:

```
insert_then_delete: CONJECTURE FORALL (m: M), (k: K), (v: V):
    delete(insert(m, k, v), k) = m
```

Intuitively, the above states that inserting a key `k` into a hashmap `m` and then deleting `k`, yields `m` unchanged. The first proof went very smoothly, let's see about this one. The same strategy used before produces the following result:

```
Starting prover session for insert_then_delete


insert_then_delete :

|-------
{1}   FORALL (m: M), (k: K), (v: V): delete(insert(m, k, v), k)
= m

>> (grind :theories ("hashmap"))
```

```
Trying repeated skolemization, instantiation, and if-lifting

insert_then_delete :

|-------
{1}   m!1 WITH [(k!1) := null] = m!1

>>
```

The identifiers with `!` in them are Skolem constants—arbitrary representatives for quantified variables. Notice that PVS has already simplified away one of the two `WITH` expressions: if it only substituted the axioms, it could have equivalently written `m!1 WITH [(k!1) := v!1] WITH [(k!1) := null] = m!1`.

The resulting sequent is requesting us to prove that two functions (i.e. hashmaps) are equal. We try to appeal to the principle of extensionality, which states that two functions are equal if their values are the same for every point of their domains:

```
>> (apply-extensionality)


Applying extensionality

insert_then_delete :

|-------
{1}   m!1 WITH [(k!1) := null](x!1) = m!1(x!1)
[2]   m!1 WITH [(k!1) := null] = m!1

>> (delete 2)


Deleting some formulas

insert_then_delete :

|-------
[1]   m!1 WITH [(k!1) := null](x!1) = m!1(x!1)
```

```
>>
```

At any point we can invoke the `delete` command to remove an element of a sequent; here the original formula was deleted to reduce clutter, since it is equivalent to the more interesting extensional form.[5]

This sequent is asking us to show that the value associated with an arbitrary key `x!1` is the same before and after `m` was updated for `k!1`. We can do a case analysis here to consider whether `x!1` = `k!1`.

```
>> (lift-if)


Lifting IF-conditions to the top level

insert_then_delete :

|-------
{1}   IF x!1 = k!1 THEN null = m!1(x!1) ELSE TRUE ENDIF

>> (ground)


Applying propositional simplification and decision procedures

insert_then_delete :

{-1}   x!1 = k!1
|-------
{1}   null = m!1(x!1)

>>
```

Let's look closely at this sequent: it is asking to demonstrate that if $x_1 = k_1$, then $m_1(x_1) =$ `null`, but modus ponens, we also are trying to show that $m_1(k_1) =$ `null`. That is, that the hashmap's value for k_1 was `null` before we inserted k_1. But it doesn't necessarily have to be so! If the value associated with k_1 was a non-null value, say v_2, then the `insert` operation *updates* the association and the later `delete` associates k_1 to `null`. Thus, our

[5]Recall that to prove a sequent we need to prove that the conjunction of the antecedents implies the disjunction of the consequents.

conjecture (that the hashmap is unchanged following an `insert` + `delete`) is true only under the assumption that the key we add to the hashmap wasn't already in the hashmap (i.e. $m(k) =$ `null`).

If we change our conjecture to

```
insert_then_delete: CONJECTURE FORALL (m: M), (k: K), (v: V):
    lookup(m, k) = null => delete(insert(m, k, v), k) = m
```

then, we can prove it with

```
(grind :theories ("hashmap"))
(apply-extensionality)
(delete 2)
(grind :theories ("hashmap"))
```

1.3.1 Axiomatic Issues

Let's now introduce two seemingly innocuous axioms, extending our specification the same way we have done before:

```
is_in?: [M, K -> bool]
is_in_ax: AXIOM FORALL (m: M), (k: K):
    m(k) /= null

insert_is_in_ax: AXIOM FORALL (m: M), (k: K), (v: V):
    is_in?(insert(m, k, v), k)
```

The semantics of these axioms should be fairly self-explanatory: a key is in the hashmap if it is not null, and after inserting some key k in a hashmap `m`, `k` is in `m`.

But these axioms hide an inconsistency!

Exercise 1.3.1. Prove, without using PVS, that true = false, based on the axioms provided so far.

Hint: start from `is_in?(insert(empty, k, null), k)`.

In fact, it shouldn't be possible that after associating a key k with `null`, we find that the value associated with k is not `null`! We have clearly abused axiomatic definitions and PVS will happily run this impossible proof for us, given that the inconsistency is within the axioms themselves, that PVS assumes to hold. We'll see later how to avoid abusing axioms, but for now, let's try to carry out the exercise proof using PVS.

```
Starting prover session for what


what :

|-------
{1}   FORALL (k: K): is_in?(insert(empty, k, null), k)

>> (skolem!)


Skolemizing

what :

|-------
{1}   is_in?(insert(empty, k!1, null), k!1)

>> (use "insert_ax")


Using lemma insert_ax

what :

{-1}   insert(empty, k!1, null) = empty WITH [(k!1) := null]
|-------
[1]   is_in?(insert(empty, k!1, null), k!1)

>> (use "is_in_ax")


Using lemma is_in_ax

what :

{-1}   insert(empty, k!1, null)(k!1) /= null
[-2]   insert(empty, k!1, null) = empty WITH [(k!1) := null]
|-------
[1]   is_in?(insert(empty, k!1, null), k!1)
```

```
>> (use "empty_ax")


Using lemma empty_ax

what :

{-1}   empty(k!1) = null
[-2]   insert(empty, k!1, null)(k!1) /= null
[-3]   insert(empty, k!1, null) = empty WITH [(k!1) := null]
|-------
[1]   is_in?(insert(empty, k!1, null), k!1)

>> (ground)

Q.E.D.
```

The complete `.pvs` file for this example can be found in §A.1.

1.4 Exercises

Exercise 1.4.1. Encode in PVS grammar the following properties of natural numbers, without proving them. Note that in PVS to denote that a variable is a natural number, we write e.g. `FORALL (n: nat)`.

1. Let $n \in \mathbf{N}$, $n > 5 \Rightarrow n > 10$.
2. Let $x, y, z \in \mathbf{N}$. If $x = y + z$, then $x^2 = yx + zx$, $xy = y^2 + zy$, and $xz = z^2 + yz$.
3. Let $x, y, z \in \mathbf{N}$. If $x = y + z$ and the above formulae hold, then $x^2 = y^2 + z^2 + 2yz$, and $(y + z)^2 = y^2 + z^2 + 2yz$.

What happens when you try to prove these properties using `(grind)`?

Chapter 2

Fundamentals of PVS Syntax and its Prover

You can read more on the “PVS Language Reference” (Owre et al., 2020) and the “PVS Prover Guide” (Shankar et al., 2020).

A language cheat-sheet can be found in Appendix B.

Chapter 3

Concurrent Reference Counting

- shared memory is an array of objects
- objects have a reference count (and nothing else) $\Rightarrow$ cannot have reference cycles
- prove that:
  - objects reaching refs = 0 are freed (flag)
  - ($\star$) no thread tries to incref a freed object $\Rightarrow$ impossible $\Rightarrow$ necessity of GC
- introduce a GC: a thread decides to be the GC (non-determinism)
- prove ($\star$)
- objects have a “data” field: a reference to another object $\Rightarrow$ cycles
- prove that every object is eventually freed

Let us now go through a more interesting specification: one that deals with concurrent memory management. In this chapter we'll practice ...

As a basis, we observe that computer memory is an array of bytes. On a 64-bit computer, you can address memory from byte number 0, to byte number $2^{64} - 1$. This is not a simplistic assumption that we're making here: it's the abstraction that all modern computers and operating systems use. Of course, memory is more complicated than this: there is virtual memory, kernel-space memory, file-system mapped memory, and process-specific memory, to name a few.

In here, we cannot go in so deep as to take all of these different kinds of memory into consideration. Instead, we'll be specifying the memory for one single process, as if all the rest is not there.

Another assumption we'll make is that the machine memory is unbounded. This sort of makes memory management redundant, but here we're interested in specifying the behavior of a management system irrespective of the amount of memory it can dispose of.

We can be free to add this constraint later, and we will, after we dealt with the other issues that will come up shortly.

To keep things simple, we'll specify a reference counting-based, object-oriented scheme. That is, we assume that our model process has knowledge of thing called objects, that may represent different classes of data. In managing memory, though, what the classes of data are isn't important. Each object has an amount of memory associated with it and however large or small it may be, we can assume the correct amount will be allocated (because we assumed memory is infinite), and deallocated once the object is freed.

3.1 Specification

The following rules are the behavior of the memory management system we would like to specify:

Rule 1. When an object is allocated, its reference count will be 1.

Rule 2. Any object can be freed if its reference count is 0.

Rule 3. When a new reference is made to an existing object, the object's reference count must be increased by 1.

Rule 4. When an existing reference to an object is removed, the object's reference count must be decreased by 1.

The rules, that will be verified, are based on these intuitive notions:

1. when an object is allocated, the returned pointer is the only reference to that object;

2. such is the case when no other object or variable is referencing it, thus it is safe to reclaim the memory;

3-4. the stored reference count must be kept consistent with its semantics.

The rules above ensure essentially that, in principle, any allocated object can be eventually freed, when its presence is no longer necessary.

Notice though, how Rule (2) doesn't specify that the object *is* freed, only that it *can* be freed. This will become important later, but first we need to talk about why these properties alone are not sufficient.

There are important features of a memory management system that we're not guaranteeing: eventual reclamation, and use-after-free avoidance. As already stated, Rule (2) only requires the possibility of freeing; we want to enforce that an object that can be freed, will be freed.

Use-after-free is a commonly known bug of programs that don't employ a memory management system, or that use it incorrectly. What happens is that a piece of memory is freed and then used again, as the name implies, either for a read or for a write. At that point, the program generally crashes with a segmentation fault. A more subtle, but possibly worse, manifestation of this bug is when the program requests to allocate memory and the allocation routine returns some address x, the program then frees x, requests more memory to be allocated, and the allocation routine returns x again.[1] If we distinguish the two states of the object at x with A and B, we may be surprised when using the pointer x as if it was A, but instead it is B. This may not be a problem per se, but generally can be a very tricky situation in a concurrent system. In fact it could be that one thread t_1 frees A and allocates an unrelated object B, while another thread t_2 still had a pointer x that it considered to pertain to A. This of course is a bug in the overall program. In a reference count-based memory scheme, a possible cause of such a bug is that the code of t_2 didn't correctly account for a new reference to A, or that t_1 incorrectly decremented the reference count.

Let us add more rules to deal with these problems.

Rule 5. When the reference count of an object is 0, it remains 0 until the object is freed.

Rule 6. If an object is being freed, its reference count must be 0.

Rule 7. If an object has been freed, no thread tries to use it.

Rule 8. After the reference count of an object becomes 0,[2] the object is eventually freed.

```
Object: TYPE = [#
```

[1]This is not an unlikely scenario, many memory management systems allocate chunks of memory. When a call to the free routine is made, the system marks that piece of memory within a chunk as free, and when a successive call to the allocation routine for the same size is made, the previously freed piece is unmarked and returned.

This behavior is useful to improve performance by not continuously relying on calls to the kernel when allocations are frequent. Object-oriented and functional language runtimes often exhibit this usage pattern.

[2]Such is a definite, unique point in the program execution due to the combination of Rules (1), and (5).

```
    refcount: nat,
    free: bool
#]

Address: TYPE = nat

memory: [Address -> Object]
```

Exercise 3.1.1. Consider `gc_cycle_4`. What, if anything, would change if it was split into multiple functions, each mutating the global state one variable at a time? Explain your answer.

3.2 Big Data

Let's add data to our objects! How much data? A lot of data! Yes! No seriously, how much? I need to write the number of bytes to allocate for this buffer. Well, some people say it's gigabytes, some say it's exabytes, and others more reasonably say it's "whatever doesn't fit in your RAM."

Let us not succumb to the major pitfall of the trendy big data: trying to define "big." Let's take a step back. We're trying to reason on the correctness of a memory management system. In particular, of an abstract system that can use an infinite amount of space! Try to beat infinity in your "big data" system. It turns out that with infinite space we could store a single bit per object and still have the possibility of encoding all the natural numbers. In fact, we don't even need that bit! We can encode whatever information we need simply in the address of one object. Since there can be an infinite amount of objects, there must be an infinite number of addresses. Assuming that the memory allocator gives out an object at a perfectly random address, after an infinite number of attempts, we will manage to get the object at the exact address that we need.

Except of course if that object had been previously allocated. Alright, we do have this limitation, fair enough. But it can be easily solved! Let's add a pointer field to every object, such that if the pointer of object α points to the address of α, then the semantics are those previously described: the object α encodes the number α. Otherwise, if it points to another object β, then it means that α represents the number β.[3]

This poses a problem for our GC. If there exist two distinct objects α and β, such that α points to β, then β cannot be freed before α is freed.

[3]This is a ludicrous introduction. In fact, to store a pointer to an unbounded number of memory locations, it's necessary to store a pointer of infinite size. Nevertheless, what follows does hold for garbage collection in general.

Appendix A

Solutions

A.1 The simple hashmap example

```
hashmap: THEORY
BEGIN
    K: TYPE
    V: TYPE
    M: TYPE = [K -> V]

    null: V
    empty: M
    empty_ax: AXIOM
        FORALL (k: K): empty(k) = null

    lookup: [M, K -> V]
    lookup_ax: AXIOM FORALL (m: M) (k: K):
        lookup(m, k) = m(k)

    insert: [M, K, V -> M]
    insert_ax: AXIOM FORALL (m: M) (k: K) (v: V):
        insert(m, k, v) = m WITH [(k) := v]

    delete: [M, K -> M]
    delete_ax: AXIOM FORALL (m: M) (k: K):
        delete(m, k) = m WITH [(k) := null]

    is_in?: [M, K -> bool]
    is_in_ax: AXIOM FORALL (m: M), (k: K):
        m(k) /= null

    insert_is_in_ax: AXIOM FORALL (m: M), (k: K), (v:
       V):
        is_in?(insert(m, k, v), k)

    insert_then_lookup: CONJECTURE FORALL (m: M) (k: K
       ) (v: V):
        lookup(insert(m, k, v), k) = v

    insert_then_delete: CONJECTURE FORALL (m: M), (k:
       K), (v: V):
        lookup(m, k) = null => delete(insert(m, k, v),
            k) = m

    what: CONJECTURE FORALL (k: K):
        is_in?(insert(empty, k, null), k)
END hashmap
```

A.2 Exercise 1.4.1

```
chapter_1: THEORY
BEGIN

    ex_1: CONJECTURE FORALL (n: nat):
        n > 5 => n > 10

    ex_2_1?(x,y,z: nat): bool = x^2 = y*x + z*x

    ex_2_2?(x,y,z: nat): bool = x*y = y^2 + z*y

    ex_2_3?(x,y,z: nat): bool = x*z = z^2 + y*z

    ex_2: CONJECTURE FORALL (x,y,z: nat):
        x = y + z => (ex_2_1?(x,y,z) and ex_2_2?(x,y,z
           ) and ex_2_3?(x,y,z))

    ex_3: CONJECTURE FORALL (x,y,z: nat):
        (x = y + z and ex_2_1?(x,y,z) and ex_2_2?(x,y,
           z) and ex_2_3?(x,y,z)) =>
            (x^2 = y^2 + z^2 + 2*y*z and (y + z)^2 = y
              ^2 + z^2 + 2*y*z)

END chapter_1
```

A.3 Chapter 3 Pseudocode

```
extern int T;  // number of threads (T >= 1)

typedef struct {
    int refcount;
    bool allocated;
    bool freed;
} object;

// each thread sees its own registers and variables
// think of them as thread-local storage
object *registers[10] = {NULL, NULL, NULL, NULL, NULL,
    NULL, NULL, NULL, NULL, NULL};
object *variables[10] = {NULL, NULL, NULL, NULL, NULL,
    NULL, NULL, NULL, NULL, NULL};

typedef struct GC {
    object *obj;
    struct GC *next;
} GC;

GC *gc_head = NULL;
bool STW_world_stopped = false;  // stop the world
bool STW_requested = false;
int STW_count_down = T;


void alloc()
{
    do {
        object *obj = choose();
    } while (!CAS(obj->allocated, false, true));
    obj->refcount = 1;
    registers[0] = obj;
}
```

```
void free()
{
    CAS(registers[0]->freed, false, true);
}

void inc_ref()
{
    object *obj = registers[0];

    do {
        int refcount = obj->refcount;  // registers[1]

        if (obj->refcount == 0) {
            registers[0] = NULL;
            return;
        }
    } while (!CAS(obj->refcount, refcount, refcount +
       1));
}

void dec_ref_naif()
{
    object *obj = registers[0];

    do {
        int refcount = obj->refcount;

        if (obj->refcount == 0) {
            registers[0] = NULL;
            return;
        }
    } while (!CAS(obj->refcount, refcount, refcount -
       1));
    refcount--;

    if (refcount == 0) {
        free(obj);
    }
}

void dec_ref()
```

```
{
    object *obj = registers[0];

    do {
        int refcount = obj->refcount;  // registers[1]

        if (obj->refcount == 0) {
            registers[0] = NULL;
            return;
        }
    } while (!CAS(obj->refcount, refcount, refcount -
       1));
    refcount--;

    if (refcount == 0) {
        GC *gc = new GC();  // simplified as one
           atomic operation in `dec_ref_7`

        gc->obj = obj;
        gc->next = gc_head;

        gc_tail = gc_head;
        do {
            while (gc_tail->next != NULL) {
                gc_tail = gc_tail->next;
            }
        } while (!CAS(gc_tail->next, NULL, gc);
    }
}

void collect()
{
    GC *gc = gc_head;

    while (gc != NULL) {
        free(gc->obj);
        gc = gc->next;
    }

    gc_head = NULL;
}
```

```
void gc_cycle()
{
    if (!CAS(STW_requested, false, true)) {
        return;
    }

    while (STW_count_down - 1 > 0) { }  // wait for
       all threads (except this one)
    STW_world_stopped = true;

    collect();

    STW_count_down = T;
    STW_requested = false;
    STW_world_stopped = false;  // resume the world (
       must be last)
}

void STW_wait()
{
    while (STW_world_stopped) { }

    if (STW_requested) {
        do {
            count_down = STW_count_down;
        } while (!CAS(STW_count_down, count_down,
           count_down - 1));

        while (STW_requested || STW_world_stopped) { }
    }
}


int main()
{
    while (true) {
        STW_wait();  // maybe wait if the world is
           stopped, or if a stop is requested

        int action = choose();  // registers[5]
        int variable = choose();  // registers[6]

        switch (action) {
        case 0:  // alloc
            if (variables[variable] != NULL) {
                break;
            }

            alloc();

            object *obj = registers[0];
            if (obj == NULL) {
                goto error;
            }
            variables[variable] = obj;
            registers[0] = NULL;

            break;
        case 1:  // dec_ref
            if (variables[variable] == NULL) {
                break;
            }

            registers[0] = variables[variable];
            dec_ref();

            if (registers[0] == NULL) {
                goto error;
            }

            variables[variable] = NULL;
            registers[0] = NULL;
            break;
        case 2:  // inc_ref
            if (registers[variable] != NULL) {
                break;
            }

            object *obj = choose();  // registers[7]
            if (!obj->allocated) {
                break;
```

```
            }

            variables[variable] = obj;
            registers[0] = variables[variable];
            inc_ref();

            if (registers[0] == NULL) {
                variables[variable] = NULL;
            }

            registers[0] = NULL;
            break;
        case 3:  // GC
            gc_cycle();
            break;
        default:
            break;
        }
    }

    error:
    while (true) { }
}
```

A.4 Chapter 3 Solution

```
refcount[T: {x: nat | x >= 1}]: THEORY
BEGIN

    Object: TYPE = [#
        refcount: nat,
        allocated: bool,
        freed: bool
    #]

    Address: TYPE = nat
    NULL: Address = 0

    Register: TYPE = upto[10]
    Variable: TYPE = upto[10]
    Thread: TYPE = upto[T]
```

```
t, t1, t2: VAR Thread

% GC: DATATYPE
% BEGIN
%     gc_NULL: gc_NULL?
%     gc(obj: Address, next: GC): gc?
% END GC

GC: TYPE = list[Address]

State: TYPE = [#
    memory: [Address -> Object],

    registers: [Thread -> [Register -> nat]],
    variables: [Thread -> [Variable -> Address]],

    gc_head: GC,

    STW_world_stopped: bool,
    STW_requested: bool,
    STW_count_down: upto[T],

    pc: [Thread -> nat]
#]

s, s1, s2: VAR State


% operations
alloc(t, s1, s2): bool = s1`pc(t) = 28 and (
    exists (a: Address): (
        (
            % preconditions
            s1`memory(a) = (# refcount := 0,
               allocated := false, freed := false
               #)
        )
    ) => (
        % mutation (allocation)
        s2 = s1 with [
            (memory)(a) := (# refcount := 1,
               allocated := true, freed := false
               #),
            (registers)(t)(0) := a,
            (pc)(t) := 154
        ]
    )
)

free(t, s1, s2): bool = s1`pc(t) = 38 and (
    if s1`memory(s1`registers(t)(0))`freed = false
        then
        s2 = s1 with [
            (memory)(s1`registers(t)(0)) := s1`
               memory(s1`registers(t)(0)) with [
                (freed) := true
            ],
            (pc)(t) := 107
        ]
    else
        s2 = s1 with [
            (pc)(t) := 107
        ]
    endif
)

inc_ref_1(t, s1, s2): bool = s1`pc(t) = 44 and (
    s2 = s1 with [
        (registers)(t)(1) := s1`memory(s1`
           registers(t)(0))`refcount,
        (pc)(t) := 46
    ]
)

inc_ref_2(t, s1, s2): bool = s1`pc(t) = 46 and (
    if s1`registers(t)(1) = 0 then
        s2 = s1 with [
            (pc)(t) := 47
        ]
    else
        s2 = s1 with [
            (pc)(t) := 50
        ]
    endif
)

inc_ref_3(t, s1, s2): bool = s1`pc(t) = 47 and (
    s2 = s1 with [
        (registers)(t)(0) := NULL,
        (pc)(t) := 196
    ]
)

inc_ref_4(t, s1, s2): bool = s1`pc(t) = 50 and (
    if s1`memory(s1`registers(t)(0))`refcount /=
       s1`registers(t)(1) then
        s2 = s1 with [
            (pc)(t) := 44
        ]
    else
        s2 = s1 with [
            (memory)(s1`registers(t)(0)) := s1`
               memory(s1`registers(t)(0)) with [
                (refcount) := s1`registers(t)(1) +
                    1
            ],
            (pc)(t) := 196
        ]
    endif
)

dec_ref_1(t, s1, s2): bool = s1`pc(t) = 77 and (
    s2 = s1 with [
        (registers)(t)(1) := s1`memory(s1`
           registers(t)(0))`refcount,
        (pc)(t) := 79
    ]
)

dec_ref_2(t, s1, s2): bool = s1`pc(t) = 79 and (
    if s1`registers(t)(1) = 0 then
        s2 = s1 with [
            (pc)(t) := 80
        ]
    else
        s2 = s1 with [
            (pc)(t) := 83
        ]
    endif
)

dec_ref_3(t, s1, s2): bool = s1`pc(t) = 80 and (
    s2 = s1 with [
        (registers)(t)(0) := NULL,
        (pc)(t) := 175
    ]
)

dec_ref_4(t, s1, s2): bool = s1`pc(t) = 83 and (
    if s1`memory(s1`registers(t)(0))`refcount /=
       s1`registers(t)(1) then
        s2 = s1 with [
            (pc)(t) := 77
        ]
    else
        s2 = s1 with [
            (memory)(s1`registers(t)(0)) := s1`
               memory(s1`registers(t)(0)) with [
                (refcount) := s1`registers(t)(1) -
                    1
            ],
            (pc)(t) := 84
        ]
    endif
)

dec_ref_5(t, s1, s2): bool = s1`pc(t) = 84 and (
    s2 = s1 with [
        (registers)(t)(1) := s1`registers(t)(1) -
           1,
        (pc)(t) := 86
    ]
)
```

```
dec_ref_6(t, s1, s2): bool = s1`pc(t) = 86 and (
    if s1`registers(t)(1) = 0 then
        s2 = s1 with [
            (pc)(t) := 87
        ]
    else
        s2 = s1 with [
            (pc)(t) := 175
        ]
    endif
)

dec_ref_7(t, s1, s2): bool = s1`pc(t) = 87 and (
   % perform lines 87 to 97 as one atomic
   operation
    s2 = s1 with [
        (gc_head) := append(s1`gc_head, cons(s1`
           registers(t)(0), null)),
        (pc)(t) := 107
    ]
)

collect(t, s1, s2): bool = s1`pc(t) = 107 and (
    if s1`gc_head = null then
        s2 = s1 with [
            (pc)(t) := 124
        ]
    else
        s2 = s1 with [
            (gc_head) := cdr(s1`gc_head),
            (memory)(car(s1`gc_head)) := s1`memory
               (car(s1`gc_head)) with [
                (freed) := true
            ],
            (pc)(t) := 107
        ]
    endif
)

gc_cycle_1(t, s1, s2): bool = s1`pc(t) = 115 and (
    if s1`STW_requested = false then
```

```
        s2 = s1 with [
            (STW_requested) := true,
            (pc)(t) := 119
        ]
    else
        s2 = s1 with [
            (pc)(t) := 204
        ]
    endif
)

gc_cycle_2(t, s1, s2): bool = s1`pc(t) = 119 and (
    if s1`STW_count_down - 1 > 0 then
        s2 = s1 with [
            (pc)(t) := 119
        ]
    else
        s2 = s1 with [
            (pc)(t) := 120
        ]
    endif
)

gc_cycle_3(t, s1, s2): bool = s1`pc(t) = 120 and (
    s2 = s1 with [
        (STW_world_stopped) := true,
        (pc)(t) := 107
    ]
)

gc_cycle_4(t, s1, s2): bool = s1`pc(t) = 124 and (
    s2 = s1 with [
        (STW_world_stopped) := false,
        (STW_requested) := false,
        (STW_count_down) := T,
        (pc)(t) := 204
    ]
)

STW_wait_1(t, s1, s2): bool = s1`pc(t) = 131 and (
    if s1`STW_world_stopped then
```

```
        s2 = s1 with [
            (pc)(t) := 131
        ]
    else
        s2 = s1 with [
            (pc)(t) := 133
        ]
    endif
)

STW_wait_2(t, s1, s2): bool = s1`pc(t) = 133 and (
    if s1`STW_requested then
        s2 = s1 with [
            (pc)(t) := 135
        ]
    else
        s2 = s1 with [
            (pc)(t) := 148
        ]
    endif
)

STW_wait_3(t, s1, s2): bool = s1`pc(t) = 135 and (
    s2 = s1 with [
        (registers)(t)(1) := s1`STW_count_down,
        (pc)(t) := 136
    ]
)

STW_wait_4(t, s1, s2): bool = s1`pc(t) = 136 and (
    if s1`registers(t)(1) = s1`STW_count_down then
        s2 = s1 with [
            (STW_count_down) := s1`registers(t)(1)
                - 1,
            (pc)(t) := 138
        ]
    else
        s2 = s1 with [
            (pc)(t) := 135
        ]
    endif
```

```
)

STW_wait_5(t, s1, s2): bool = s1`pc(t) = 138 and (
    if s1`STW_requested or s1`STW_world_stopped
       then
        s2 = s1 with [
            (pc)(t) := 138
        ]
    else
        s2 = s1 with [
            (pc)(t) := 148
        ]
    endif
)

main_1(t, s1, s2): bool = s1`pc(t) = 146 and (
    s2 = s1 with [
        (pc)(t) := 131
    ]
)

main_2(t, s1, s2): bool = s1`pc(t) = 148 and (
    exists (action: upto[3]):
        s2 = s1 with [
            (registers)(t)(5) := action,
            (pc)(t) := 149
        ]
)

main_3(t, s1, s2): bool = s1`pc(t) = 149 and (
    exists (variable: upto[3]):
        s2 = s1 with [
            (registers)(t)(6) := variable,
            (pc)(t) := 151
        ]
)

main_4(t, s1, s2): bool = s1`pc(t) = 151 and (
    cond
        s1`registers(t)(5) = 0 -> s2 = s1 with [
            (pc)(t) := 153
        ],
        s1`registers(t)(5) = 1 -> s2 = s1 with [
            (pc)(t) := 168
        ],
        s1`registers(t)(5) = 2 -> s2 = s1 with [
            (pc)(t) := 183
        ],
        s1`registers(t)(5) = 3 -> s2 = s1 with [
            (pc)(t) := 203
        ],
        else -> s2 = s1 with [
            (pc)(t) := 146
        ]
    endcond
)

main_5(t, s1, s2): bool = s1`pc(t) = 153 and (
    if s1`variables(t)(s1`registers(t)(6)) /= NULL
        then
        s2 = s1 with [
            (pc)(t) := 146
        ]
    else
        s2 = s1 with [
            (pc)(t) := 157
        ]
    endif
)

main_6(t, s1, s2): bool = s1`pc(t) = 157 and (
    s2 = s1 with [
        (pc)(t) := 28
    ]
)

main_7(t, s1, s2): bool = s1`pc(t) = 168 and (
    if s1`registers(t)(0) = NULL then
        s2 = s1 with [
            (pc)(t) := 211
        ]
    else
        s2 = s1 with [
            (pc)(t) := 163
        ]
    endif
)

main_8(t, s1, s2): bool = s1`pc(t) = 163 and (
    s2 = s1 with [
        (variables)(t)(s1`registers(t)(6)) := s1`
           registers(t)(0),
        (registers)(t)(0) := NULL,
        (pc)(t) := 146
    ]
)

main_9(t, s1, s2): bool = s1`pc(t) = 168 and (
    if s1`variables(t)(s1`registers(t)(6)) = NULL
       then
        s2 = s1 with [
            (pc)(t) := 146
        ]
    else
        s2 = s1 with [
            (pc)(t) := 172
        ]
    endif
)

main_10(t, s1, s2): bool = s1`pc(t) = 172 and (
    s2 = s1 with [
        (registers)(t)(0) := s1`variables(t)(s1`
           registers(t)(6)),
        (pc)(t) := 173
    ]
)

main_11(t, s1, s2): bool = s1`pc(t) = 173 and (
    s2 = s1 with [
        (pc)(t) := 74
    ]
)
```

```
main_12(t, s1, s2): bool = s1`pc(t) = 175 and (
    if s1`registers(t)(0) = NULL then
        s2 = s1 with [
            (pc)(t) := 211
        ]
    else
        s2 = s1 with [
            (pc)(t) := 179
        ]
    endif
)

main_13(t, s1, s2): bool = s1`pc(t) = 179 and (
    s2 = s1 with [
        (variables)(t)(s1`registers(t)(6)) := NULL
           ,
        (registers)(t)(0) := NULL,
        (pc)(t) := 146
    ]
)

main_14(t, s1, s2): bool = s1`pc(t) = 183 and (
    if s1`variables(t)(s1`registers(t)(6)) /= NULL
        then
        s2 = s1 with [
            (pc)(t) := 146
        ]
    else
        s2 = s1 with [
            (pc)(t) := 187
        ]
    endif
)

main_15(t, s1, s2): bool = s1`pc(t) = 187 and (
    exists (a: Address):
        s2 = s1 with [
            (registers)(t)(7) := a,
            (pc)(t) := 188
        ]
)

main_16(t, s1, s2): bool = s1`pc(t) = 188 and (
    if not s1`memory(s1`registers(t)(7))`allocated
        then
        s2 = s1 with [
            (pc)(t) := 146
        ]
    else
        s2 = s1 with [
            (pc)(t) := 192
        ]
    endif
)

main_17(t, s1, s2): bool = s1`pc(t) = 192 and (
    s2 = s1 with [
        (variables)(t)(s1`registers(t)(6)) := s1`
           registers(t)(7),
        (registers)(t)(0) := s1`registers(t)(7),
        (pc)(t) := 194
    ]
)

main_18(t, s1, s2): bool = s1`pc(t) = 194 and (
    s2 = s1 with [
        (pc)(t) := 44
    ]
)

main_19(t, s1, s2): bool = s1`pc(t) = 196 and (
    if s1`registers(t)(0) = NULL then
        s2 = s1 with [
            (pc)(t) := 197
        ]
    else
        s2 = s1 with [
            (pc)(t) := 200
        ]
    endif
)
```

```
main_20(t, s1, s2): bool = s1`pc(t) = 197 and (
    s2 = s1 with [
        (variables)(t)(s1`registers(t)(6)) := NULL
           ,
        (pc)(t) := 200
    ]
)

main_21(t, s1, s2): bool = s1`pc(t) = 200 and (
    s2 = s1 with [
        (registers)(t)(0) := NULL,
        (pc)(t) := 146
    ]
)

main_22(t, s1, s2): bool = s1`pc(t) = 203 and (
    s2 = s1 with [
        (pc)(t) := 115
    ]
)

main_23(t, s1, s2): bool = s1`pc(t) = 204 and (
    s2 = s1 with [
        (pc)(t) := 146
    ]
)

error(t, s1, s2): bool = s1`pc(t) = 211 and (
    s2 = s1 with [
        (pc)(t) := 211
    ]
)


% transitions
step(t, s1, s2): bool =
    alloc(t, s1, s2)
    or free(t, s1, s2)
    or inc_ref_1(t, s1, s2)
    or inc_ref_2(t, s1, s2)
    or inc_ref_3(t, s1, s2)
    or inc_ref_4(t, s1, s2)
    or dec_ref_1(t, s1, s2)
    or dec_ref_2(t, s1, s2)
    or dec_ref_3(t, s1, s2)
    or dec_ref_4(t, s1, s2)
    or dec_ref_5(t, s1, s2)
    or dec_ref_6(t, s1, s2)
    or dec_ref_7(t, s1, s2)
    or collect(t, s1, s2)
    or gc_cycle_1(t, s1, s2)
    or gc_cycle_2(t, s1, s2)
    or gc_cycle_3(t, s1, s2)
    or gc_cycle_4(t, s1, s2)
    or STW_wait_1(t, s1, s2)
    or STW_wait_2(t, s1, s2)
    or STW_wait_3(t, s1, s2)
    or STW_wait_4(t, s1, s2)
    or STW_wait_5(t, s1, s2)
    or main_1(t, s1, s2)
    or main_2(t, s1, s2)
    or main_3(t, s1, s2)
    or main_4(t, s1, s2)
    or main_5(t, s1, s2)
    or main_6(t, s1, s2)
    or main_7(t, s1, s2)
    or main_8(t, s1, s2)
    or main_9(t, s1, s2)
    or main_10(t, s1, s2)
    or main_11(t, s1, s2)
    or main_12(t, s1, s2)
    or main_13(t, s1, s2)
    or main_14(t, s1, s2)
    or main_15(t, s1, s2)
    or main_16(t, s1, s2)
    or main_17(t, s1, s2)
    or main_18(t, s1, s2)
    or main_19(t, s1, s2)
    or main_20(t, s1, s2)
    or main_21(t, s1, s2)
    or main_22(t, s1, s2)
```

```
    or main_23(t, s1, s2)
    or error(t, s1, s2)


% state invariants
init(s): bool =
    forall (a: Address, t: Thread, r: Register, v:
        Variable): (
        s`memory(a) = (# refcount := 0, allocated
           := false, freed := false #)
        and s`registers(t)(r) = NULL
        and s`variables(t)(v) = NULL
        and s`pc(t) = 146
    ) and (
        s`gc_head = null
        and s`STW_world_stopped = false
        and s`STW_requested = false
        and s`STW_count_down = T
    )

rule_1(s): bool =
    forall (t: Thread):
        pc(s)(t) = 154 => memory(s)(registers(s)(t
           )(0)) = (# refcount := 1, allocated :=
           true, freed := false #)

% rule_3(s1, s2): bool =
%     forall (t: Thread):
%         pc(s1)(t) = 44 =>

rule_6(s): bool =
    forall (t: Thread):
        pc(s)(t) = 38 => (
            s`memory(s`registers(t)(0))`refcount =
                0
            and s`memory(s`registers(t)(0))`freed
               = false
            and s`memory(s`registers(t)(0))`
               allocated = true
        )
```

```
rule_7(s): bool =
    forall (t: Thread, v: Variable):
        variables(s)(t)(v) /= NULL => memory(s)(
           variables(s)(t)(v))`freed = false

not_in_error_state(s): bool =
    forall (t: Thread):
        pc(s)(t) /= 211

INV(s): bool =
    rule_1(s)
    and rule_6(s)
    and rule_7(s)
    and not_in_error_state(s)


% intermediate proofs
dec_ref_4_implies_dec_ref_2: LEMMA
    forall (t: Thread): forall(s1: State, s2:
       State):
        (
            INV(s1) and s1`pc(t) = 79 and s1`
               registers(t)(1) /= 0 and step(t, s1
               , s2)
        ) => (
            INV(s2) and s2`pc(t) = 83 and s2`
               registers(t)(1) /= 0
        )


% core proofs
% @QED init_implies_state_invariant proved by dp
   on Thu, 06 Jun 2024 15:45:12 GMT
init_implies_invariants: THEOREM
    forall (s: State):
        init(s) => INV(s)

step_implies_invariants: THEOREM
    forall (t: Thread, s1: State, s2: State):
        step(t, s1, s2) and INV(s1) => INV(s2)
```

```
END refcount
```

A.5 hesselink

```
hashtable7[P : ({ i:nat | i>0 }) ]: THEORY
BEGIN

% The library of finite set can be download from the
   PVS site.

IMPORTING finite_sets[nat]

% General definitions, Axims and their lemmas

range(i:nat):  TYPE = {s: nat | 1 <= s & s <= i}

size(s : setof[range(P)]) : upto(P)

sizeAX1 : AXIOM
  forall (s : setof[range(P)]):
    (forall (i:range(P)): NOT member(i,s)) => size(s)
       =0

sizeAX2 : AXIOM
  forall (s : setof[range(P)]):
    forall (p:range(P)): s(p) =>
         size(s)=1+size(remove(p,s))

sizeAX3: AXIOM
   forall (S,U : setof[range(P)]):
       disjoint?(S,U) =>size(union(S,U)) = size(S) +
          size(U)

equalSet : THEOREM
   forall (S,T : setof[range(P)]):
    (forall (n1:range(P)): S(n1)=T(n1))<=> S=T

equalSize : THEOREM
   forall (S,T : setof[range(P)]):
    (forall (n1:range(P)): S(n1)=T(n1))=> size(S)=size
       (T)
```

```
incSize : THEOREM
   forall (S,T : setof[range(P)]):
    (exists (n1:range(P)):
          S(n1) AND NOT T(n1) AND
          forall (n2:range(P)): (n1/=n2 => S(n2)=T(n2)
            )) =>
      size(S)=size(T)+1

sizeTM : THEOREM
  forall (s : setof[range(P)]): forall (i : range(P)):
    member(i,s) => size(s)=1+size(remove(i,s))

sizeTM1 : THEOREM
  forall (s : setof[range(P)]): forall (i : range(P)):
    NOT(member(i,s)) => size(s)=size(remove(i,s))

posSize : THEOREM
   forall (S : setof[range(P)]):
      ((exists (n1:range(P)): S(n1)) <=> size(S)>=1)

more1Size : THEOREM
   forall (S : setof[range(P)]):
      ((exists (n1,n2:range(P)): S(n1) AND S(n2) AND
         n1/=n2) => size(S)>1)

zeroSize : THEOREM
   forall (S : setof[range(P)]):
     ((forall (n1:range(P)): NOT S(n1)) => size(S)=0)


oneSize : THEOREM
  forall (S : setof[range(P)]):
    (exists (n1:range(P)):
          S(n1) AND
          forall (n2:range(P)): (n1/=n2 => NOT S(n2)))
             =>
      size(S)=1

Size_diff_subset: THEOREM
   forall (S,U : setof[range(P)]):
      ( forall (n : range(P)): (U(n) =>  S(n))) =>
      size(difference(S,U))=size(S)-size(U)

largeEqualSize : THEOREM
   forall (S,U : setof[range(P)]):
      ( forall (n : range(P)): (U(n) =>  S(n))) =>
      size(S)>=size(U)

largerSize : THEOREM
   forall (S,U : setof[range(P)]):
        size(union(S,U))<=size(S)+size(U)

% Datatypes for interface

Address : TYPE+

Value : Type+

null : Value

nonNull: AXIOM
    EXISTS (v :Value ): v/=null

EValue : DATATYPE BEGIN
           del : delp
           old(val1 : Value) : oldp
           nor(val2 : Value) : norp
         END EValue

nullValue : AXIOM
    forall (v :Value ): v/=null => nor(v)/=nor(null)

val(e : EValue) : Value =
  IF delp(e) THEN null
  ELSIF oldp(e) THEN val1(e)
  ELSE val2(e)
  ENDIF

oldEValue : AXIOM
    forall (v :EValue ): oldp(v) => old(val(v))=v
```

```
oldEValue1: AXIOM
    forall (v:EValue): old(val(v))=old(null) => val(v)
       =null
norEValue : AXIOM
    forall (v :EValue ): norp(v) => nor(val(v))=v

ADR : [ Value -> Address ]

nullAddress: Address=ADR(null)

Ax1 : AXIOM
  forall ( v : Value ):
      v=null <=> ADR(v)=nullAddress

% Hashtable structure

Hashtable_ : TYPE =
  [# size : nat,
     occ : nat,
     dels : nat,
     bound : nat,
     table_ : [ below(size) -> EValue ]
  #]

Hashtable : DATATYPE BEGIN
               bot : bot?
               i(i_ : Hashtable_) : i?
            END Hashtable

% Hash function

KEY : [ Address, nat, nat -> nat ]

Ax2 : AXIOM
  forall ( a : Address, l,n : nat ):  % l: length of
     the array (hash table), n: distance
     KEY(a,l,n)<l

Ax3 : AXIOM
  forall ( a : Address, i,j,n : nat ):
     i/=j AND i<n AND j<n => KEY(a,n,i)/=KEY(a,n,j)
```

```
KEYs(a:Address, l:nat, n:nat): setof[nat]=
       {x:nat| EXISTS (i:{s:nat|s<n}): x=KEY(a,l,i)}

equal_finite_set : THEOREM
   forall (S,T : finite_set):
    (forall (n1:nat): S(n1)=T(n1))<=> S=T

KEYs_1 : THEOREM
  forall ( a : Address, l,n : nat ):
     FORALL (k:nat): member(k,KEYs(a,l,n))=>k<l

KEYs_2 : THEOREM
  forall ( a : Address,l : nat):
     FORALL (m:below(l+1)): card(KEYs(a,l,m))=m

KEYs_3 : THEOREM
  forall ( a : Address,l : nat):
     KEYs(a,l,l)={s:nat|s<l}

KEYsurjective : THEOREM
  forall ( a : Address, l,k : nat ):
     k<l => exists (n:below(l)): k=KEY(a,l,n)

lessCardinal: THEOREM
 FORALL (E1, E2: finite_set[nat]):
  (EXISTS (f:[(E1)->(E2)]): injective?(f))=> card(E1)
     <=card(E2)

% the state vector

state : TYPE =
  [#

% ghost hashtable, whose item equals the corresponding
    item of current hashtable
% alsomost everywhere except not being taged to be "
   old" or "done".
     Y: [nat -> EValue],

% global variables
```

```
     H : [ range(2*P) -> nat ],
     busy : [ range(2*P) -> nat ],
     prot : [ range(2*P) -> nat ],
     next : [ range(2*P) -> upto(2*P) ],
     currInd : range(2*P),
     Heap : [ nat -> Hashtable ],
     H_index : nat,   % Index of position of new
        hashtable in Heap

% private variables to each process:
     index : [ range(P) -> range(2*P) ],

% local variables of procedures, private to each
   process:
%find
     a_find : [ range(P) -> Address ],
     r_find : [ range(P) -> EValue ],
     n_find : [ range(P) -> nat ],
     l_find : [ range(P) -> nat ],
     h_find : [ range(P) -> nat ],
     cnt_find: [ range(P) -> nat ],
%delete
     a_delete : [ range(P) -> Address ],
     r_delete : [ range(P) -> EValue ],
     n_delete : [ range(P) -> nat],
     k_delete : [ range(P) -> nat ],
     l_delete : [ range(P) -> nat],
     h_delete : [ range(P) -> nat ],
     suc_delete : [ range(P) -> bool ],
     cnt_delete : [ range(P) -> nat ],
%insert
     a_insert : [ range(P) -> Address ],
     v_insert : [ range(P) -> Value ],
     r_insert : [ range(P) -> EValue ],
     n_insert : [ range(P) -> nat ],
     k_insert : [ range(P) -> nat ],
     l_insert : [ range(P) -> nat ],
     h_insert : [ range(P) -> nat ],
     suc_insert : [ range(P) -> bool ],
     cnt_insert : [ range(P) -> nat ],
%assign
```

```
     a_assign : [ range(P) -> Address ],
     v_assign : [ range(P) -> Value ],
     r_assign : [ range(P) -> EValue ],
     n_assign : [ range(P) -> nat ],
     k_assign : [ range(P) -> nat ],
     l_assign : [ range(P) -> nat ],
     h_assign : [ range(P) -> nat ],
     suc_assign : [ range(P) -> bool ],
     cnt_assign : [ range(P) -> nat ],
%getAccess
     return_getAccess : [ range(P) -> nat ],
%releaseAccess
     i_releaseAccess : [ range(P) -> range(2*P) ],
     h_releaseAccess : [ range(P) -> nat ],
     return_releaseAccess : [ range(P) -> nat ],
%newTable
     h_newTable : [ range(P) -> nat ],
     i_newTable : [ range(P) -> range(2*P) ],
     return_newTable : [ range(P) -> nat ],
%refresh
     return_refresh : [ range(P) -> nat ],
%migrate
     h_migrate : [ range(P) -> nat ],
     i_migrate : [ range(P) -> upto(2*P) ],
%movecontents
     from_moveContents : [ range(P) -> nat ],
     to_moveContents : [ range(P) -> nat ],
     i_moveContents : [ range(P) -> nat ],
     v_moveContents: [ range(P) -> EValue ],
     toBeMoved_moveContents : [ range(P) -> setof[nat]
        ],
%moveElement
     v_moveElement : [ range(P) -> Value ],
     to_moveElement : [ range(P) -> nat ],
     a_moveElement : [ range(P) -> Address ],
     k_moveElement : [ range(P) -> nat ],
     m_moveElement : [ range(P) -> nat ],
     n_moveElement : [range(P) -> nat ],
     w_moveElement : [ range(P) -> EValue ],
     b_moveElement : [ range(P) -> bool ],
% global variable of the specification:
     X : [Address -> Value ],
% local variables of specification:
% find
     rS_find : [ range(P) -> Value ],
% delete
     sucS_delete : [ range(P) -> bool ],
% insert
     sucS_insert : [ range(P) -> bool ],
% assign

% local program counters
     pc : [ range(P) -> nat ]

% History variables
  #]

% Program steps

s,s1,s2 : VAR state
p : VAR range(P)

% state corresponding to not being in any procedure of
% the hash table has number 0, entering a procedure
   can
% be done with the enter transition.

enter(p,s1,s2) : bool =
    ( pc(s1)(p)=0 AND % call getAccess
        s2=s1 WITH [ (pc)(p) := 59,
                     (return_getAccess)(p) := 1
                     ]
    ) OR
    ( pc(s1)(p)=1 AND
     ( % call find
       (exists (a : Address): a/=nullAddress AND
         s2=s1 WITH [ (pc)(p) := 5,
                      (a_find)(p) := a
                    ]
       ) OR % call delete
       (exists (a : Address): a/=nullAddress AND
         s2=s1 WITH [ (pc)(p) := 15,
                      (a_delete)(p) := a
                    ]
       ) OR % call insert
       (exists (v : Value): v/=null AND
         s2=s1 WITH [ (pc)(p) := 27,
                      (v_insert)(p) := v,
                      (a_insert)(p) := ADR(v)
                                          ]
       ) OR % call assign
       (exists (v : Value): v/=null AND
         s2=s1 WITH [ (pc)(p) := 43,
                      (v_assign)(p) := v,
                      (a_assign)(p) := ADR(v)
                    ]
       ) OR % call releaseAccess
       ( s2=s1 WITH [ (pc)(p) := 67,
                      (return_releaseAccess)(p) := 0,
                      (i_releaseAccess)(p) := index(s1
                         )(p)
                    ]
      ))
    )

% find

find5(p,s1,s2) : bool =
     pc(s1)(p)=5 AND
     s2 = s1 WITH [ (pc)(p) := 6,
                    (h_find)(p) := H(s1)(index(s1)(p))
                       ,
                    (n_find)(p) := 0,
                    (cnt_find)(p) :=0
                  ]

find6(p,s1,s2) : bool =
     i?(Heap(s1)(h_find(s1)(p))) AND
     pc(s1)(p)=6 AND
     s2 = s1 WITH [ (pc)(p) := 7,
                    (l_find)(p) := size(i_(Heap(s1)(
                       h_find(s1)(p))))
                   ]

find7(p,s1,s2) : bool =
     i?(Heap(s1)(h_find(s1)(p))) AND
     l_find(s1)(p)=size(i_(Heap(s1)(h_find(s1)(p))))
        AND
     pc(s1)(p)=7 AND (
     if r=nor(null) OR a_find(s1)(p)=ADR(val(r))
     then s2 = s1 WITH [ (pc)(p) := 8,
                         (r_find)(p) := r,
                         (rS_find)(p) := X(s1)(a_find(
                            s1)(p)),
                         (cnt_find)(p) := cnt_find(s1)
                            (p)+1
                       ]
     else s2 = s1 WITH [ (pc)(p) := 8,
                         (r_find)(p) := r
                      ]
    endif
    )WHERE r=table_(i_(Heap(s1)(h_find(s1)(p))))
                             (KEY(a_find(s1)(p),l_find
                                (s1)(p),n_find(s1)(p))
                                )

find8(p,s1,s2) : bool =
     pc(s1)(p)=8 AND
       if r_find(s1)(p)=old(null)
       then s2 = s1 WITH [ (pc)(p) := 94,
                           (return_refresh)(p) := 10 ]
       else s2 = s1 WITH [ (pc)(p) := 13,
                           (n_find)(p) := n_find(s1)(p
                              )+1
                         ]
       endif

find10(p,s1,s2) : bool =
     pc(s1)(p)=10 AND
     s2 = s1 WITH [ (pc)(p) := 11,
                    (h_find)(p) := H(s1)(index(s1)(p))
                       ,
                    (n_find)(p) := 0
```

```
                ]

find11(p,s1,s2) : bool =
     i?(Heap(s1)(h_find(s1)(p))) AND
     pc(s1)(p)=11 AND
     s2 = s1 WITH [ (pc)(p) := 13,
                    (l_find)(p) := size(i_(Heap(s1)(
                       h_find(s1)(p))))
                ]

find13(p,s1,s2) : bool =
     pc(s1)(p)=13 AND
     if r_find(s1)(p) = nor(null) OR a_find(s1)(p)=ADR
        (val(r_find(s1)(p)))
     then s2=s1 WITH [ (pc)(p) := 14 ]
     else s2=s1 WITH [ (pc)(p) := 7 ]
     endif

find14(p,s1,s2) : bool =
     pc(s1)(p)=14 AND
     s2=s1 WITH [ (pc)(p) := 1 ]

% Delete

delete15(p,s1,s2) : bool =
     pc(s1)(p)=15 AND
     s2 = s1 WITH [ (pc)(p) := 16,
                    (cnt_delete)(p) :=0,
                    (h_delete)(p) := H(s1)(index(s1)(p
                       )),
                    (suc_delete)(p) := FALSE
                ]

delete16(p,s1,s2) : bool =
     i?(Heap(s1)(h_delete(s1)(p))) AND
     pc(s1)(p)=16 AND
     s2 = s1 WITH [ (pc)(p) := 17,
                    (l_delete)(p) := size(i_(Heap(s1)(
                       h_delete(s1)(p)))),
                    (n_delete)(p) :=0
               ]
```

```
delete17(p,s1,s2) : bool =
     i?(Heap(s1)(h_delete(s1)(p))) AND
     l_delete(s1)(p) = size(i_(Heap(s1)(h_delete(s1)(p
        )))) AND
     pc(s1)(p)=17 AND (
     IF r=nor(null) THEN
        s2 = s1 WITH [ (pc)(p) := 18,
                       (k_delete)(p) :=
                             KEY(a_delete(s1)(p),
                                l_delete(s1)(p),
                                n_delete(s1)(p)),
                       (r_delete)(p) := r,
                       (cnt_delete)(p) := cnt_delete(
                          s1)(p)+1,
                       (sucS_delete)(p):= X(s1)(
                          a_delete(s1)(p))/=null,
                       (X)(a_delete(s1)(p)):=
                          if X(s1)(a_delete(s1)(p))/=
                             null
                          then null
                          else X(s1)(a_delete(s1)(p))
                          endif
                  ]
     ELSE
        s2 = s1 WITH [ (pc)(p) := 18,
                       (k_delete)(p) :=
                             KEY(a_delete(s1)(p),
                                l_delete(s1)(p),
                                n_delete(s1)(p)),
                       (r_delete)(p) := r
                     ]
     ENDIF
     ) WHERE r= table_(i_(Heap(s1)(h_delete(s1)(p))))
                         (KEY(a_delete(s1)(p),l_delete
                            (s1)(p),n_delete(s1)(p)))


delete18(p,s1,s2) : bool =
     i?(Heap(s1)(h_delete(s1)(p))) AND
     k_delete(s1)(p)=KEY(a_delete(s1)(p),l_delete(s1)(
        p),n_delete(s1)(p)) AND
     l_delete(s1)(p)=size(i_(Heap(s1)(h_delete(s1)(p))
        )) AND
     pc(s1)(p)=18 AND
     IF oldp(r_delete(s1)(p))
     THEN s2 = s1 WITH [ (pc)(p) := 94,
                         (return_refresh)(p) := 20
                       ]
     ELSE IF a_delete(s1)(p) = ADR(val(r_delete(s1)(p)
        ))
     THEN IF table_(i_(Heap(s1)(h_delete(s1)(p))))
                          (k_delete(s1)(p))=r_delete(
                             s1)(p)
          THEN s2 = s1 WITH [ (pc)(p) := 25,
                              (suc_delete)(p):= TRUE,
                              (cnt_delete)(p) :=
                                 cnt_delete(s1)(p)+1,
                              (sucS_delete)(p):=(X(s1)
                                 (a_delete(s1)(p))/=
                                 null),
                              (X)(a_delete(s1)(p)):=
                                  IF X(s1)(a_delete(s1
                                     )(p))/=null
                                  THEN null
                                  ELSE X(s1)(a_delete(
                                     s1)(p))
                                  ENDIF,
                             (Y) := (Y(s1)) WITH
                                [ (k_delete(s1)(p)) :=
                                    del ],

                             (Heap)(h_delete(s1)(p))
                                := i(h) WHERE
                               h = i_(Heap(s1)(
                                  h_delete(s1)(p)))
                                  WITH
                                [ (table_)(k_delete(s1
                                   )(p)) := del
                                ]
                             ]
          ELSE s2 = s1 WITH [ (pc)(p) := 17
```

```
                            ]
          ENDIF
     ELSE s2 = s1 WITH [ (pc)(p) := IF r_delete(s1)(p)
        =nor(null)
                                    THEN 25 ELSE 17
                                       ENDIF,
                         (n_delete)(p) := n_delete(s1)
                            (p)+1
                       ]
     ENDIF
     ENDIF

delete20(p,s1,s2) : bool =
     pc(s1)(p)=20 AND
     s2 = s1 WITH [ (pc)(p) := 21,
                    (h_delete)(p) :=H(s1)(index(s1)(p)
                       )
                  ]

delete21(p,s1,s2) : bool =
     i?(Heap(s1)(h_delete(s1)(p))) AND
     pc(s1)(p)=21 AND
     s2 = s1 WITH [ (pc)(p) := 17,
                    (l_delete)(p) := size(i_(Heap(s1)(
                       h_delete(s1)(p)))),
                    (n_delete)(p) := 0
                  ]
delete25(p,s1,s2) : bool =
     i?(Heap(s1)(h_delete(s1)(p))) AND
     pc(s1)(p)=25 AND
     IF suc_delete(s1)(p)
     THEN s2 = s1 WITH [ (pc)(p) := 26,
                         (Heap)(h_delete(s1)(p)) := i(
                            h) WHERE
                           h = i_(Heap(s1)(h_delete(s1
                              )(p))) WITH
                           [ (dels) := dels(i_(Heap(s1
                              )
                                            (h_delete
                                               (s1)(p
                                               ))))+1
                              ]
                          ]
     ELSE s2 = s1 WITH [ (pc)(p) := 26 ]
     ENDIF

delete26(p,s1,s2) : bool =
     pc(s1)(p)=26 AND
     s2 = s1 WITH [ (pc)(p) := 1 ]


% Insert

insert27(p,s1,s2) : bool =
     pc(s1)(p)=27 AND
     s2 = s1  WITH [ (pc)(p) := 28,
                     (cnt_insert)(p) :=0,
                     (h_insert)(p) := H(s1)(index(s1)(
                        p))
                   ]

insert28(p,s1,s2) : bool =
     i?(Heap(s1)(h_insert(s1)(p))) AND
     pc(s1)(p)=28 AND
     IF occ(i_(Heap(s1)(h_insert(s1)(p)))) >
                  bound(i_(Heap(s1)(h_insert(s1)(p))))
     THEN s2 = s1 WITH [ (pc)(p) := 77,
                         (return_newTable)(p) := 30
                       ]
     ELSE s2 = s1 WITH [ (pc)(p) := 31 ]
     ENDIF

insert30(p,s1,s2) : bool =
     pc(s1)(p)=30 AND
     s2 = s1 WITH [ (pc)(p) := 31,
                    (h_insert)(p) := H(s1)(index(s1)(p
                       ))
                  ]

insert31(p,s1,s2) : bool =
     i?(Heap(s1)(h_insert(s1)(p))) AND
     pc(s1)(p)=31 AND
     s2 = s1 WITH [ (pc)(p) := 32,
                    (n_insert)(p) := 0,
                    (suc_insert)(p) :=FALSE,
                    (l_insert)(p) := size(i_(Heap(s1)(
                       h_insert(s1)(p))))
                  ]

insert32(p,s1,s2) : bool =
     pc(s1)(p)=32 AND
     s2 = s1 WITH [ (pc)(p) := 33,
                    (k_insert)(p) :=
                         KEY(a_insert(s1)(p),l_insert(
                            s1)(p),n_insert(s1)(p))
                  ]

insert33(p,s1,s2) : bool =
     i?(Heap(s1)(h_insert(s1)(p))) AND
     l_insert(s1)(p)=size(i_(Heap(s1)(h_insert(s1)(p))
        )) AND
     k_insert(s1)(p) = KEY(a_insert(s1)(p),l_insert(s1
        )(p),n_insert(s1)(p)) AND
     pc(s1)(p)=33 AND (
     IF a_insert(s1)(p)=ADR(val(r)) THEN
        s2 = s1 WITH [ (pc)(p) := 35,
                       (r_insert)(p) :=
                         table_(i_(Heap(s1)(h_insert(
                            s1)(p))))(k_insert(s1)(p))
                            ,
                       (cnt_insert)(p) := cnt_insert(
                          s1)(p)+1,
                       (sucS_insert)(p):= X(s1)(
                          a_insert(s1)(p))=null,
                       (X)(a_insert(s1)(p)):=
                          if X(s1)(a_insert(s1)(p))=
                             null
                          then v_insert(s1)(p)
                          else X(s1)(a_insert(s1)(p))
                          endif
                  ]
     ELSE
        s2 = s1 WITH [ (pc)(p) := 35,
                       (r_insert)(p) :=
                         table_(i_(Heap(s1)(h_insert(
                            s1)(p))))(k_insert(s1)(p))
                      ]
     ENDIF
     ) WHERE r= table_(i_(Heap(s1)(h_insert(s1)(p))))(
        k_insert(s1)(p))

insert35(p,s1,s2) : bool =
     i?(Heap(s1)(h_insert(s1)(p))) AND
     l_insert(s1)(p)=size(i_(Heap(s1)(h_insert(s1)(p))
        )) AND
     k_insert(s1)(p) = KEY(a_insert(s1)(p),l_insert(s1
        )(p),n_insert(s1)(p)) AND
     pc(s1)(p)=35 AND
     IF oldp(r_insert(s1)(p))
     THEN s2 = s1 WITH [ (pc)(p) := 94,
                         (return_refresh)(p) := 36
                       ]
     ELSE IF r_insert(s1)(p)=nor(null)
     THEN IF table_(i_(Heap(s1)(h_insert(s1)(p))))
                                                 (k_insert
                                                   (s1)(p
                                                   ))=nor
                                                   (null)
          THEN s2 = s1 WITH
              [ (pc)(p) := 41,
                (cnt_insert)(p) := cnt_insert(s1)(p)
                      +1,
                (sucS_insert)(p) := (X(s1)(a_insert(s1
                      )(p))=null),
                (X)(a_insert(s1)(p)):=
                  IF X(s1)(a_insert(s1)(p))=null
                  THEN v_insert(s1)(p)
                  ELSE X(s1)(a_insert(s1)(p))
                  ENDIF,
                (suc_insert)(p) :=TRUE,

                (Y) := (Y(s1)) WITH
                         [ (k_insert(s1)(p)) := nor(
                            v_insert(s1)(p)) ],
                (Heap)(h_insert(s1)(p)) := i(h) WHERE
                     h = i_(Heap(s1)(h_insert(s1)(p)))
                         WITH
                         [ (table_)(k_insert(s1)(p))
                            := nor(v_insert(s1)(p)) ]
              ]
          ELSE s2 = s1 WITH
              [ (pc)(p) := 32
              ]
          ENDIF
     ELSE s2 = s1 WITH
              [ (pc)(p) := IF a_insert(s1)(p)=ADR(val(
                 r_insert(s1)(p)))
                           THEN 41
                           ELSE 32
                           ENDIF,
                (n_insert)(p) := n_insert(s1)(p)+1
              ]
     ENDIF
     ENDIF

insert36(p,s1,s2) : bool =
     pc(s1)(p)=36 AND
     s2 = s1 WITH [ (pc)(p) := 37,
                    (h_insert)(p) := H(s1)(index(s1)(p
                       ))
                  ]

insert37(p,s1,s2) : bool =
     i?(Heap(s1)(h_insert(s1)(p))) AND
     pc(s1)(p)=37 AND
     s2 = s1 WITH [ (pc)(p) := IF (a_insert(s1)(p)=ADR
        (val(r_insert(s1)(p)))
                          OR suc_insert(s1)(p))
                              THEN 41
                              ELSE 32
                              ENDIF,
                    (n_insert)(p) := 0,
                    (l_insert)(p) := size(i_(Heap(s1)(
                       h_insert(s1)(p))))
                    ]

% insert40(p,s1,s2) : bool = s1=s2


insert41(p,s1,s2) : bool =
     i?(Heap(s1)(h_insert(s1)(p))) AND
     pc(s1)(p)=41 AND
     IF suc_insert(s1)(p)
     THEN s2 = s1 WITH [ (pc)(p) := 42,
                         (Heap)(h_insert(s1)(p)) := i(
                            h) WHERE
                           h = i_(Heap(s1)(h_insert(s1
                              )(p))) WITH
                           [ (occ) := occ(i_(Heap(s1)
                                            (h_insert
                                               (s1)(p
                                               ))))+1
                            ]
                        ]
     ELSE s2 = s1 WITH [ (pc)(p) := 42 ]
     ENDIF


insert42(p,s1,s2) : bool =
     pc(s1)(p)=42 AND
     s2 = s1 WITH [ (pc)(p) := 1 ]

% Assign


assign43(p,s1,s2) : bool =
     pc(s1)(p)=43 AND
     s2 = s1  WITH [ (pc)(p) := 44,
                     (cnt_assign)(p) :=0,
                     (h_assign)(p) := H(s1)(index(s1)(
                        p))
                   ]

assign44(p,s1,s2) : bool =
     i?(Heap(s1)(h_assign(s1)(p))) AND
```

```
      pc(s1)(p)=44 AND
      IF occ(i_(Heap(s1)(h_assign(s1)(p)))) >
                    bound(i_(Heap(s1)(h_assign(s1)(p))))
      THEN s2 = s1 WITH [ (pc)(p) := 77,
                            (return_newTable)(p) := 46
                          ]
      ELSE s2 = s1 WITH [ (pc)(p) := 47 ]
      ENDIF

assign46(p,s1,s2) : bool =
      pc(s1)(p)=46 AND
      s2 = s1 WITH [ (pc)(p) := 47,
                      (h_assign)(p) := H(s1)(index(s1)(p
                         ))
                    ]

assign47(p,s1,s2) : bool =
      i?(Heap(s1)(h_assign(s1)(p))) AND
      pc(s1)(p)=47 AND
      s2 = s1 WITH [ (pc)(p) := 48,
                      (n_assign)(p) := 0,
                      (suc_assign)(p) := FALSE,
                      (l_assign)(p) := size(i_(Heap(s1)(
                         h_assign(s1)(p))))
                    ]

assign48(p,s1,s2) : bool =
      pc(s1)(p)=48 AND
      s2 = s1 WITH [ (pc)(p) := 49,
                      (k_assign)(p) :=
                           KEY(a_assign(s1)(p),l_assign(
                              s1)(p),n_assign(s1)(p))
                    ]

assign49(p,s1,s2) : bool =
      i?(Heap(s1)(h_assign(s1)(p))) AND
      l_assign(s1)(p)=size(i_(Heap(s1)(h_assign(s1)(p))
         )) AND
      k_assign(s1)(p) = KEY(a_assign(s1)(p),l_assign(s1
         )(p),n_assign(s1)(p)) AND
      pc(s1)(p)=49 AND
      s2 = s1 WITH [ (pc)(p) := 50,
                     (r_assign)(p) :=
                          table_(i_(Heap(s1)(h_assign(
                             s1)(p))))
                                                                    (
                                                                       k_assign
                                                                       (
                                                                       s1
                                                                       )
                                                                       (
                                                                       p
                                                                       )
                                                                       )

                   ]


 assign50(p,s1,s2) : bool =
      i?(Heap(s1)(h_assign(s1)(p))) AND
      l_assign(s1)(p)=size(i_(Heap(s1)(h_assign(s1)(p))
         )) AND
      k_assign(s1)(p) = KEY(a_assign(s1)(p),l_assign(s1
         )(p),n_assign(s1)(p)) AND
      pc(s1)(p)=50 AND
      IF oldp(r_assign(s1)(p))
      THEN s2 = s1 WITH [ (pc)(p) := 94,
                          (return_refresh)(p) := 51
                        ]
      ELSE IF (r_assign(s1)(p)=nor(null) OR a_assign(s1
         )(p)=ADR(val(r_assign(s1)(p))))
      THEN IF table_(i_(Heap(s1)(h_assign(s1)(p))))
                          (k_assign(s1)(p))=r_assign(s1
                             )(p)
           THEN s2 = s1 WITH [ (pc)(p) := 57,
                               (cnt_assign)(p) :=
                                  cnt_assign(s1)(p)+1,
                               (X)(a_assign(s1)(p)):=
                                  v_assign(s1)(p),
                               (suc_assign)(p) := TRUE,

                               (Y) := (Y(s1)) WITH
```

```
                                [ (k_assign(s1)(p)) :=
                                    nor(v_assign(s1)(p
                                   ))],

                              (Heap)(h_assign(s1)(p))
                                 := i(h) WHERE
                                h = i_(Heap(s1)(
                                   h_assign(s1)(p)))
                                   WITH
                                    [ (table_)(
                                       k_assign(s1)(p)
                                       ) :=
                                                    nor
                                                       (
                                                       v_assign
                                                       (
                                                       s1
                                                       )
                                                       (
                                                       p
                                                       )
                                                       )

                                                       ]

                            ]
          ELSE s2 = s1 WITH [ (pc)(p) := 48
                            ]
          ENDIF
     ELSE s2 = s1 WITH [ (pc)(p) := 48,
                         (n_assign)(p) := n_assign(s1)
                            (p)+1 ]
     ENDIF
     ENDIF

assign51(p,s1,s2) : bool =
     pc(s1)(p)=51 AND
     s2 = s1 WITH [ (pc)(p) := 52,
                    (h_assign)(p) := H(s1)(index(s1)(p
                       ))
                  ]
```

```
assign52(p,s1,s2) : bool =
     i?(Heap(s1)(h_assign(s1)(p))) AND
     pc(s1)(p)=52 AND
     s2 = s1 WITH [ (pc)(p) := 48,
                    (n_assign)(p) := 0,
                    (l_assign)(p) := size(i_(Heap(s1)(
                       h_assign(s1)(p))))
                  ]

assign56(p,s1,s2) : bool = s2=s1

assign57(p,s1,s2) : bool =
     i?(Heap(s1)(h_assign(s1)(p))) AND
     pc(s1)(p)=57 AND
     IF r_assign(s1)(p)=nor(null)
     THEN s2 = s1 WITH [ (pc)(p) := 1,
                         (Heap)(h_assign(s1)(p)) := i(
                           h) WHERE
                           h = i_(Heap(s1)(h_assign(s1
                              )(p))) WITH
                              [ (occ) := occ(i_(Heap(
                                 s1)
                                             (h_assign
                                                (s1)(p
                                                ))))+1
                                                 ]
                       ]
     ELSE s2 = s1 WITH [ (pc)(p) := 1 ]
     ENDIF


% GetAccess

getAccess59(p,s1,s2) : bool =
     pc(s1)(p)=59 AND
     s2 = s1 WITH [ (pc)(p) := 60,
                    (index)(p) := currInd(s1)
                  ]

getAccess60(p,s1,s2) : bool =
     pc(s1)(p)=60 AND
     s2 = s1 WITH [ (pc)(p) := 61,
                    (prot)(index(s1)(p)) := prot(s1)(
                       index(s1)(p))+1
                  ]

getAccess61(p,s1,s2) : bool =
     pc(s1)(p)=61 AND
     IF index(s1)(p)=currInd(s1)
     THEN s2 = s1 WITH [ (pc)(p) := 62 ]
     ELSE s2 = s1 WITH [ (pc)(p) := 65 ]
     ENDIF

getAccess62(p,s1,s2) : bool =
     pc(s1)(p)=62 AND
     s2 = s1 WITH [ (pc)(p) := 63,
                    (busy)(index(s1)(p)) := busy(s1)(
                       index(s1)(p))+1
                  ]

getAccess63(p,s1,s2) : bool =
     pc(s1)(p)=63 AND
     IF index(s1)(p)=currInd(s1)
     THEN s2 = s1 WITH [ (pc)(p) := return_getAccess(
        s1)(p) ]
     ELSE s2 = s1 WITH [ (pc)(p) := 67,
                         (return_releaseAccess)(p) :=
                            59,
                         (i_releaseAccess)(p) := index
                            (s1)(p)
                       ]
     ENDIF

getAccess65(p,s1,s2) : bool =
     prot(s1)(index(s1)(p))>0 AND
     pc(s1)(p)=65 AND
     s2 = s1 WITH [ (pc)(p) := 59,
                    (prot)(index(s1)(p)) := prot(s1)(
                       index(s1)(p))-1
                  ]
getAccess66(p,s1,s2) : bool = s2=s1
```

```
% ReleaseAccess

releaseAccess67(p,s1,s2) : bool =
     pc(s1)(p)=67 AND
     s2 = s1 WITH [ (pc)(p) := 68,
                    (h_releaseAccess)(p) := H(s1)(
                       i_releaseAccess(s1)(p))
                  ]

releaseAccess68(p,s1,s2) : bool =
     busy(s1)(i_releaseAccess(s1)(p))>0 AND
     pc(s1)(p)=68 AND
     s2 = s1 WITH [ (pc)(p) := 69,
                    (busy)(i_releaseAccess(s1)(p)) :=
                           busy(s1)(i_releaseAccess(s1
                              )(p))-1
                  ]

releaseAccess69(p,s1,s2) : bool =
     pc(s1)(p)=69 AND
     IF  h_releaseAccess(s1)(p)/=0 AND  busy(s1)(
        i_releaseAccess(s1)(p))=0
     THEN s2 = s1 WITH [ (pc)(p) := 70 ]
     ELSE s2 = s1 WITH [ (pc)(p) := 72 ]
     ENDIF

releaseAccess70(p,s1,s2) : bool =
     pc(s1)(p)=70 AND
     IF H(s1)(i_releaseAccess(s1)(p))=h_releaseAccess(
        s1)(p)
     THEN s2 = s1 WITH [ (pc)(p) := 71,
                         (H)(i_releaseAccess(s1)(p))
                            :=0]
     ELSE s2 = s1 WITH [ (pc)(p) := 72 ]
     ENDIF

releaseAccess71(p,s1,s2) : bool =
     pc(s1)(p)=71 AND
     s2 = s1 WITH [ (pc)(p) := 72,
                    (Heap)(h_releaseAccess(s1)(p)):=
                        bot
                   ]

releaseAccess72(p,s1,s2) : bool =
     prot(s1)(i_releaseAccess(s1)(p))>0 AND
     pc(s1)(p)=72 AND
     s2 = s1 WITH [ (pc)(p) := return_releaseAccess(s1
        )(p),
                   (prot)(i_releaseAccess(s1)(p)) :=
                       prot(s1)(i_releaseAccess(s1)(p
                          ))-1
                   ]

% newTable

newTable77(p,s1,s2) : bool =
     pc(s1)(p)=77 AND
     IF next(s1)(index(s1)(p))=0
     THEN s2 = s1 WITH [ (pc)(p) := 78 ]
     ELSE s2 = s1 WITH [ (pc)(p) := 94,
                         (return_refresh)(p) :=
                            return_newTable(s1)(p)
                       ]
     ENDIF

newTable78(p,s1,s2) : bool =
     pc(s1)(p)=78 AND
     exists (l:range(2*P)):
     ( IF prot(s1)(l)=0
       THEN s2 = s1 WITH [ (pc)(p) := 81,
                           (prot)(l) := 1,
                           (i_newTable)(p) := l
                         ]
       ELSE s2 = s1 WITH [ (pc)(p) := 77,
                           (i_newTable)(p) := l
                         ]
       ENDIF
     )

newTable81(p,s1,s2) : bool =
     pc(s1)(p)=81 AND
     s2 = s1 WITH [ (pc)(p) := 82,
                    (busy)(i_newTable(s1)(p)) := 1
                  ]

newTable82(p,s1,s2) : bool =
     pc(s1)(p)=82 AND  i?(Heap(s1)(H(s1)(index(s1)(p))
        )) AND
     exists (h0:Hashtable_):
     ( forall (j:below(size(h0))): (table_(h0)(j)=nor(
        null)))
      AND
       occ(h0)=0
      AND
       dels(h0)=0
      AND
       bound(h0)+2*P<size(h0)
      AND
       bound(i_(Heap(s1)(H(s1)(index(s1)(p)))))+2*P-
          dels(i_(Heap(s1)(H(s1)(index(s1)(p)))))<
          bound(h0)
      AND
        s2 = s1 WITH [ (pc)(p) := 83,
                       (Heap)(H_index(s1)) := i(h0),
                       (H)(i_newTable(s1)(p)) :=
                          H_index(s1),
                       H_index := H_index(s1)+1
                     ]


newTable83(p,s1,s2) : bool =
     pc(s1)(p)=83 AND
     s2 = s1 WITH [ (pc)(p) := 84,
                    (next)(i_newTable(s1)(p)) := 0
                  ]

newTable84(p,s1,s2) : bool =
     pc(s1)(p)=84 AND
     IF next(s1)(index(s1)(p))=0
     THEN s2 = s1 WITH [ (pc)(p) := 77,
                         (next)(index(s1)(p)) :=
                            i_newTable(s1)(p)
```

```
                        ]
      ELSE s2 = s1 WITH [ (pc)(p) := 67,
                          (i_releaseAccess)(p) :=
                             i_newTable(s1)(p),
                          (return_releaseAccess)(p) :=
                             77
                        ]
      ENDIF

% Refresh

refresh90(p,s1,s2) : bool =
      pc(s1)(p)=90 AND
      if index(s1)(p)/=currInd(s1)
      then s2=s1 WITH [ (pc)(p) := 67,
                        (return_releaseAccess)(p) :=
                           59,
                        (i_releaseAccess)(p) := index(
                           s1)(p),
                        (return_getAccess)(p) :=
                           return_refresh(s1)(p)
                      ]
      else s2=s1 WITH [ (pc)(p) := (return_refresh)(s1)
         (p) ]
      endif

% Migrate

migrate94(p,s1,s2) : bool =
      pc(s1)(p)=94 AND
      s2 = s1 WITH [ (pc)(p) := 95,
                     (i_migrate)(p) := next(s1)(index(
                        s1)(p))
                   ]

migrate95(p,s1,s2) : bool =
      1<=i_migrate(s1)(p) AND
      pc(s1)(p)=95 AND
      s2 = s1 WITH [ (pc)(p) := 97,
                          (prot)(i_migrate(s1)(p)) :=
                                  prot(s1)(i_migrate(
                                     s1)(p))+1
                   ]

migrate97(p,s1,s2) : bool =
      pc(s1)(p)=97 AND
      IF index(s1)(p) /= currInd(s1)
      THEN s2 = s1 WITH [ (pc)(p) := 98 ]
      ELSE s2 = s1 WITH [ (pc)(p) := 99 ]
      ENDIF

migrate98(p,s1,s2) : bool =
      1<=i_migrate(s1)(p) AND
      prot(s1)(i_migrate(s1)(p))>0 AND
      pc(s1)(p)=98 AND
      s2 = s1 WITH [ (pc)(p) := 90,
                     (prot)(i_migrate(s1)(p)) := prot(
                        s1)(i_migrate(s1)(p))-1
                   ]

migrate99(p,s1,s2) : bool =
      1<=i_migrate(s1)(p) AND
      pc(s1)(p)=99 AND
      s2 = s1 WITH [ (pc)(p) := 100,
                     (busy)(i_migrate(s1)(p)) := busy(
                        s1)(i_migrate(s1)(p))+1
                   ]

migrate100(p,s1,s2) : bool =
      1<=i_migrate(s1)(p) AND
      pc(s1)(p)=100 AND
      s2 = s1 WITH [ (pc)(p) := 101,
                     (h_migrate)(p) := H(s1)(i_migrate(
                        s1)(p))
                   ]

migrate101(p,s1,s2) : bool =
      pc(s1)(p)=101 AND
      i_migrate(s1)(p)>0 AND
      IF index(s1)(p)=currInd(s1)
      THEN  s2 = s1 WITH [ (pc)(p) := 102 ]
      ELSE s2 = s1 WITH [ (pc)(p) := 67,
                          (i_releaseAccess)(p) :=
                             i_migrate(s1)(p),
                          (return_releaseAccess)(p) :=
                             90
                        ]
      ENDIF

migrate102(p,s1,s2) : bool =
      i?(Heap(s1)(H(s1)(index(s1)(p)))) AND
      pc(s1)(p)=102 AND
      s2 = s1 WITH [ (pc)(p) := 110,
                     (from_moveContents)(p) := H(s1)(
                        index(s1)(p)),
                     (to_moveContents)(p) := h_migrate(
                        s1)(p),
                     (toBeMoved_moveContents)(p) := { j
                        :nat |
                         j<size(i_(Heap(s1)(H(s1)(index
                            (s1)(p)))))}
                   ]

migrate103(p,s1,s2) : bool =
      1<=i_migrate(s1)(p) AND
      pc(s1)(p)=103 AND i?(Heap(s1)(H(s1)(i_migrate(s1)
         (p)))) AND
      IF index(s1)(p)=currInd(s1)
      THEN  s2 = s1 WITH [ (pc)(p) := 104,
                           currInd := i_migrate(s1)(p),

                           (Y) :=
            LAMBDA (x:nat):
               IF x<size(i_(Heap(s1)(H(s1)(i_migrate(s1
                  )(p)))))
               THEN h(x) ELSE nor(null) ENDIF
               WHERE h = table_(i_(Heap(s1)(H(s1)(
                  i_migrate(s1)(p)))))
                         ]
      ELSE  s2 = s1 WITH [ (pc)(p) := 67,
                           (return_releaseAccess)(p) :=
                              90,
                           (i_releaseAccess)(p) :=
                              i_migrate(s1)(p)
                         ]
      ENDIF

migrate104(p,s1,s2) : bool =
      busy(s1)(index(s1)(p))>0 AND
      pc(s1)(p)=104 AND
      s2 = s1 WITH [ (pc)(p) := 105,
                     (busy)(index(s1)(p)) := busy(s1)(
                        index(s1)(p))-1
                   ]

migrate105(p,s1,s2) : bool =
      prot(s1)(index(s1)(p))>0 AND
      i_migrate(s1)(p)>0 AND
      pc(s1)(p)=105 AND
      s2 = s1 WITH [ (pc)(p) := 67,
                     (prot)(index(s1)(p)) := prot(s1)(
                        index(s1)(p))-1,
                     (return_releaseAccess)(p) := 90,
                     (i_releaseAccess)(p) := i_migrate(
                        s1)(p)
                   ]

% MoveContents


movecontents110(p,s1,s2) : bool =
      pc(s1)(p)=110 AND
      IF currInd(s1)=index(s1)(p) AND
         toBeMoved_moveContents(s1)(p)/=emptyset
      THEN s2 = s1 WITH [ (pc)(p) := 111 ]
      ELSE s2 = s1 WITH [ (pc)(p) := 103 ]
      ENDIF

movecontents111(p,s1,s2) : bool =
      i?(Heap(s1)(from_moveContents(s1)(p))) AND
      pc(s1)(p)=111 AND
      exists (l:below(size(i_(Heap(s1)(
         from_moveContents(s1)(p)))))):
```

```
      (  member(l,toBeMoved_moveContents(s1)(p))
      AND
        IF table_(i_(Heap(s1)(from_moveContents(s1)(p))
           ))(l)/=old(null)
        THEN s2 = s1 WITH [ (pc)(p) := 114,
                            (v_moveContents)(p) :=
                                     table_(i_(Heap(s1)
                                            (
                                               from_moveContents
                                               (s1)(p))
                                               ))(l),
                            (i_moveContents)(p) := l ]
        ELSE s2 = s1 WITH [ (pc)(p) := 118,
                            (v_moveContents)(p) :=
                                     table_(i_(Heap(s1)
                                            (
                                               from_moveContents
                                               (s1)(p))
                                               ))(l),
                            (i_moveContents)(p) := l ]
        ENDIF
      )

movecontents114(p,s1,s2) : bool =
      i?(Heap(s1)(from_moveContents(s1)(p))) AND
      i_moveContents(s1)(p)<size(i_(Heap(s1)
                                (from_moveContents(s1)(
                                   p)))) AND
      pc(s1)(p)=114 AND
      IF v_moveContents(s1)(p)=table_(i_(Heap(s1)
                              (from_moveContents(s1)(p)))
                                )(i_moveContents(s1)(p))
      THEN s2 = s1 WITH [ (pc)(p) := 116,
                          (Heap)(from_moveContents(s1)(
                            p)) := i(h) WHERE
                            h=i_(Heap(s1)(
                               from_moveContents(s1)(p
                               ))) WITH
                          [ (table_)(i_moveContents(s1)
                            (p)) :=
                                      old(val(
                                               v_moveContents
                                               (s1)(p))) ]
                          ]
      ELSE s2 = s1 WITH [ (pc)(p) := 110
                          ]
      ENDIF

movecontents116(p,s1,s2) : bool =
      pc(s1)(p)=116 AND
      IF val(v_moveContents(s1)(p))/=null
      THEN s2 = s1 WITH [ (pc)(p) := 120,
                          (v_moveElement)(p) := val(
                            v_moveContents(s1)(p)),
                          (to_moveElement)(p) :=
                            to_moveContents(s1)(p)
                          ]
      ELSE s2 = s1 WITH [ (pc)(p) := 117 ]
      ENDIF

movecontents117(p,s1,s2) : bool =
      i?(Heap(s1)(from_moveContents(s1)(p))) AND
      i_moveContents(s1)(p)<size(i_(Heap(s1)(
         from_moveContents(s1)(p)))) AND
      pc(s1)(p)=117 AND
      s2 = s1 WITH [ (pc)(p) := 118,
                     (Heap)(from_moveContents(s1)(p))
                        := i(h) WHERE
                      h=i_(Heap(s1)(from_moveContents(
                         s1)(p))) WITH
                     [ (table_)(i_moveContents(s1)(p))
                        := old(null) ]
                   ]

movecontents118(p,s1,s2) : bool =
      pc(s1)(p)=118 AND
      s2 = s1 WITH [ (pc)(p) := 110,
                     (toBeMoved_moveContents)(p) :=
                        remove(i_moveContents(s1)(p),
                                      toBeMoved_moveContents
                                      (s1)(p))
                   ]
```

```
% MoveElement

moveelement120(p,s1,s2) : bool =
     i?(Heap(s1)(to_moveElement(s1)(p))) AND
     pc(s1)(p)=120 AND
     s2 = s1 WITH [ (pc)(p) := 121,
                    (n_moveElement)(p) := 0,
                    (b_moveElement)(p) := FALSE,
                    (a_moveElement)(p) := ADR(
                       v_moveElement(s1)(p)),
                    (m_moveElement)(p) := size(i_(Heap
                       (s1)
                                                  (
                                                    to_moveElement
                                                    (s1
                                                    )(p
                                                    )))
                                                    )
                  ]

moveelement121(p,s1,s2) : bool =
     i?(Heap(s1)(to_moveElement(s1)(p))) AND
     m_moveElement(s1)(p)=size(i_(Heap(s1)(
        to_moveElement(s1)(p)))) AND
     pc(s1)(p)=121 AND
     s2 = s1 WITH [ (pc)(p) := 123,
                    (k_moveElement)(p) := KEY(
                       a_moveElement(s1)(p),
                              m_moveElement(s1)(p),
                                 n_moveElement(s1)(p))
                                 ,
                    (w_moveElement)(p) := table_(i_(
                       Heap(s1)
                         (to_moveElement(s1)(p))))(KEY(
                           a_moveElement(s1)(p),
                                m_moveElement(s1)(p),
                                   n_moveElement(s1)(p
                                   )))
                  ]
```

```
moveelement123(p,s1,s2) : bool =
     i?(Heap(s1)(to_moveElement(s1)(p))) AND
     k_moveElement(s1)(p)<size(i_(Heap(s1)(
        to_moveElement(s1)(p)))) AND
     pc(s1)(p)=123 AND
     IF w_moveElement(s1)(p)=nor(null)
     THEN IF table_(i_(Heap(s1)(to_moveElement(s1)(p))
        ))
                      (k_moveElement(s1)(p))=nor(null
                        )
          THEN s2 = s1 WITH [ (pc)(p) := 125,
                              (b_moveElement)(p) :=
                                 TRUE,
                              (Heap)(to_moveElement(s1
                                 )(p)) := i(h) WHERE
                               h=i_(Heap(s1)(
                                  to_moveElement(s1)(p
                                  ))) WITH
                              [ (table_)(k_moveElement
                                 (s1)(p)) :=
                                         nor(
                                            v_moveElement
                                            (s1)(p)) ]
                            ]
          ELSE s2 = s1 WITH [ (pc)(p) := 125,
                              (b_moveElement)(p) :=
                                 FALSE
                            ]
          ENDIF
     ELSE s2 = s1 WITH [ (pc)(p) := 125,
                         (n_moveElement)(p) :=
                            n_moveElement(s1)(p)+1
 ]
     ENDIF

moveelement125(p,s1,s2) : bool =
     pc(s1)(p)=125 AND
     IF b_moveElement(s1)(p) OR
             a_moveElement(s1)(p)=ADR(val(
                w_moveElement(s1)(p)))
               OR currInd(s1)/=index(s1)(p)
     THEN s2 = s1 WITH [ (pc)(p) := 126 ]
     ELSE s2 = s1 WITH [ (pc)(p) := 121 ]
     ENDIF

moveelement126(p,s1,s2) : bool =
     i?(Heap(s1)(to_moveElement(s1)(p))) AND
     pc(s1)(p)=126 AND
     IF b_moveElement(s1)(p)
     THEN s2 = s1 WITH [ (pc)(p) := 117,
                         (Heap)(to_moveElement(s1)(p))
                             := i(h) WHERE
                           h=i_(Heap(s1)(to_moveElement
                             (s1)(p))) WITH
                              [ occ := occ(i_(Heap(s1)
                                              (
                                                to_moveElement
                                                (s1)(p
                                                ))))+1
                                                 ]
                       ]
     ELSE s2 = s1 WITH [ (pc)(p) := 117 ]
     ENDIF


% Step

step(p,s1,s2) : bool =
         find5(p,s1,s2) OR
         find6(p,s1,s2) OR
         find7(p,s1,s2) OR
         find8(p,s1,s2) OR
         find10(p,s1,s2) OR
         find11(p,s1,s2) OR
         find13(p,s1,s2) OR

         delete15(p,s1,s2) OR
         delete16(p,s1,s2) OR
         delete17(p,s1,s2) OR
         delete18(p,s1,s2) OR
         delete20(p,s1,s2) OR
```

```
delete21(p,s1,s2) OR
enter(p,s1,s2) OR

insert27(p,s1,s2) OR
insert28(p,s1,s2) OR
insert30(p,s1,s2) OR
insert31(p,s1,s2) OR
insert32(p,s1,s2) OR
insert33(p,s1,s2) OR
insert35(p,s1,s2) OR
insert36(p,s1,s2) OR
insert37(p,s1,s2) OR
find14(p,s1,s2) OR
insert41(p,s1,s2) OR

assign43(p,s1,s2) OR
assign44(p,s1,s2) OR
assign46(p,s1,s2) OR
assign47(p,s1,s2) OR
assign48(p,s1,s2) OR
assign49(p,s1,s2) OR
assign50(p,s1,s2) OR
assign51(p,s1,s2) OR
assign52(p,s1,s2) OR
assign56(p,s1,s2) OR
assign57(p,s1,s2) OR

getAccess59(p,s1,s2) OR
getAccess60(p,s1,s2) OR
getAccess61(p,s1,s2) OR
getAccess62(p,s1,s2) OR
getAccess63(p,s1,s2) OR
getAccess65(p,s1,s2) OR
getAccess66(p,s1,s2) OR

releaseAccess67(p,s1,s2) OR
releaseAccess68(p,s1,s2) OR
releaseAccess69(p,s1,s2) OR
releaseAccess70(p,s1,s2) OR
releaseAccess72(p,s1,s2) OR
```

```
         releaseAccess71(p,s1,s2) OR
         delete25(p,s1,s2) OR
         newTable77(p,s1,s2) OR
         newTable78(p,s1,s2) OR
         newTable81(p,s1,s2) OR
         newTable82(p,s1,s2) OR
         newTable83(p,s1,s2) OR
         newTable84(p,s1,s2) OR

         refresh90(p,s1,s2) OR

         migrate94(p,s1,s2) OR
         migrate95(p,s1,s2) OR
         migrate97(p,s1,s2) OR
         migrate98(p,s1,s2) OR
         migrate99(p,s1,s2) OR
         migrate100(p,s1,s2) OR
         migrate101(p,s1,s2) OR
         migrate102(p,s1,s2) OR
         migrate103(p,s1,s2) OR
         migrate104(p,s1,s2) OR
         migrate105(p,s1,s2) OR

         movecontents110(p,s1,s2) OR
         movecontents111(p,s1,s2) OR
         delete26(p,s1,s2) OR
         movecontents114(p,s1,s2) OR
         insert42(p,s1,s2) OR
         movecontents116(p,s1,s2) OR
         movecontents117(p,s1,s2) OR
         movecontents118(p,s1,s2) OR

         moveelement120(p,s1,s2) OR
         moveelement121(p,s1,s2) OR
         moveelement123(p,s1,s2) OR
         moveelement125(p,s1,s2) OR
         moveelement126(p,s1,s2)

% subtypes of state for type_checking

state1: TYPE={s:state|  i?(Heap(s)(H(s)(currInd(s))))
```

```
   AND
        (forall (p:range(P)):
           pc(s)(p)/=0 and (pc(s)(p)<59 or (pc(s)(p)
              >65
                    and not (i_releaseAccess(s)(p)=
                       index(s)(p)
                             and (pc(s)(p)>=67 and pc(
                                s)(p)<=72))))=>
                    i?(Heap(s)(H(s)(index(s)(p)))))}

state2: TYPE={s:state| next(s)(currInd(s))/=0 =>
                       i?(Heap(s)(H(s)(next(s)(currInd
                          (s)))))}

state3: TYPE={s:state2| i?(Heap(s)(H(s)(currInd(s))))}

state4: TYPE={s:state| forall (p:range(P)):
       pc(s)(p)>=95 => i_migrate(s)(p)/=0}

state5: TYPE={s:state| i?(Heap(s)(H(s)(currInd(s))))
   AND forall (p:range(P)):
        pc(s)(p)>=83 and pc(s)(p)<=84 => i?(Heap(s)(H(
           s)(i_newTable(s)(p))))}

state6: TYPE={s:state4|  forall (p:range(P)):
      pc(s)(p)>=95 and index(s)(p)=currInd(s) OR pc(s)
         (p)>=102 AND pc(s)(p)<=103
          OR pc(s)(p)>=110 => i?(Heap(s)(H(s)(
             i_migrate(s)(p))))}

state7: TYPE={s:state1|  forall (p:range(P)):
         pc(s)(p)>=114 =>
             i_moveContents(s)(p)<size(i_(Heap(s)(H(s)
                (index(s)(p)))))}

state8: TYPE={s:state7| forall (p:range(P)):
       (pc(s)(p)>=95 =>  i_migrate(s)(p)/=0) AND
       (pc(s)(p)>=95 and index(s)(p)=currInd(s) OR pc(
          s)(p)>=102 AND pc(s)(p)<=103
           OR pc(s)(p)>=110 => i?(Heap(s)(H(s)(
              i_migrate(s)(p)))))}
```

```
state9: TYPE={s:state6|
          forall (p:range(P)):
           (pc(s)(p)>=123 =>
               k_moveElement(s)(p)<size(i_(Heap(s)(H(s
                  )(i_migrate(s)(p))))))
            AND
           (pc(s)(p)>=120  => to_moveElement(s)(p)=H(s
              )(i_migrate(s)(p)))}

state10: TYPE={s:state1|
  forall (p:range(P)):
      (pc(s)(p)=18 => H(s)(index(s)(p))=h_delete(s)(p)
          and
                      k_delete(s)(p)<size(i_(Heap(s)(H
                         (s)(index(s)(p))))))
        AND
      (pc(s)(p)=35 => H(s)(index(s)(p))=h_insert(s)(p)
          and
                      k_insert(s)(p)<size(i_(Heap(s)(H
                         (s)(index(s)(p))))))
        AND
      (pc(s)(p)=50 => H(s)(index(s)(p))=h_assign(s)(p)
          and
                      k_assign(s)(p)<size(i_(Heap(s)(H
                         (s)(index(s)(p))))))
      }

state11: TYPE={s:state1|
  forall (p:range(P)):
         (pc(s)(p)>=6 AND pc(s)(p)<=13 AND pc(s)(p)
            /=10 =>
               h_find(s)(p)=H(s)(index(s)(p))) AND
         (pc(s)(p)>=16 AND pc(s)(p)<=25 AND pc(s)(p)
            /=20 =>
               h_delete(s)(p)=H(s)(index(s)(p))) AND
         (pc(s)(p)>=28 AND pc(s)(p)<=41 AND pc(s)(p)
            /=30 and pc(s)(p)/=36 =>
               h_insert(s)(p)=H(s)(index(s)(p))) AND
         (pc(s)(p)>=44 AND pc(s)(p)<=57 AND pc(s)(p)
            /=46 and pc(s)(p)/=51 =>
```

```
               h_assign(s)(p)=H(s)(index(s)(p)))}

state12: TYPE={s:state11|
  forall (p:range(P)):
        (pc(s)(p)=7 OR pc(s)(p)=8 OR pc(s)(p)=13 =>
                l_find(s)(p)=size(i_(Heap(s)(h_find(s
                   )(p))))) AND
        (pc(s)(p)>=17 AND pc(s)(p)<=18 =>
               l_delete(s)(p)=size(i_(Heap(s)(
                  h_delete(s)(p))))) AND
        (pc(s)(p)>=32 AND pc(s)(p)<=35 =>
                l_insert(s)(p)=size(i_(Heap(s)(
                   h_insert(s)(p))))) AND
        (pc(s)(p)>=48 AND pc(s)(p)<=50 =>
               l_assign(s)(p)=size(i_(Heap(s)(
                  h_assign(s)(p)))))}

state13: TYPE={s:state7|
      (forall (i,j:range(2*P)): i/=j and i?(Heap(s)(H(
         s)(i)))=>H(s)(i)/= H(s)(j)) AND
      (forall (p:range(P)):
        (pc(s)(p)>=95  => i_migrate(s)(p)=next(s)(
           index(s)(p))) AND
        (pc(s)(p)>=95  => i_migrate(s)(p)/=0) AND
        ( pc(s)(p)>=95 and index(s)(p)=currInd(s) OR
            pc(s)(p)>=102 AND pc(s)(p)<=103 OR
            pc(s)(p)>=110 => i?(Heap(s)(H(s)(i_migrate
               (s)(p))))) AND
        (pc(s)(p)/=0 AND (pc(s)(p)<59 or (pc(s)(p)>65
                              and not (i_releaseAccess(
                                 s)(p)=index(s)(p)
                                       and (pc(s)(p)
                                          >=67 and pc(s
                                          )(p)<=72))))
            AND H(s)(index(s)(p))/=H(s)(currInd(s))=>
                  forall (k: below(size(i_(Heap(s)(H(s
                     )(index(s)(p))))))):
                           table_(i_(Heap(s)(H(s)(
                              index(s)(p)))))(k)=old(
                              null)))}
```

```
state14: TYPE={s:state2| next(s)(currInd(s))/=0}

state15: TYPE={s:state3| (exists (k: below(size(i_(
   Heap(s)(H(s)(currInd(s))))))):
            oldp(table_(i_(Heap(s)(H(s)(currInd(s)))))
               (k)))
               => next(s)(currInd(s))/=0}

state16: TYPE={s:state1|
     forall (p:range(P)):
       pc(s)(p)>=110 => H(s)(index(s)(p))=
          from_moveContents(s)(p)}

% auxiliary Definitions

nbSet1(s): finite_set=
       { i : below(H_index(s)) | Heap(s)(i)/=bot }

nbSet2(s): finite_set=
       {i:range(2*P)| H(s)(i)/=0 OR
              (EXISTS (p:range(P)): pc(s)(p)=71 and
                 i_releaseAccess(s)(p)=i)}

deSet1(s:{x:state| i?(Heap(x)(H(x)(currInd(x))))}):
   finite_set=
   {k:below(size(i_(Heap(s)(H(s)(currInd(s))))))|Y(s)(
      k)=del}

deSet2(s:state):setof[range(P)]= {p:range(P)|
           index(s)(p)=currInd(s) and pc(s)(p)=25 and
              suc_delete(s)(p)}

deSet3(s:state14):finite_set=
   {k:below(size(i_(Heap(s)(H(s)(next(s)(currInd(s))))
      )))|
          table_(i_(Heap(s)(H(s)(next(s)(currInd(s)))
             ))(k)=del}

ocSet1(s:state):setof[range(P)]= {p:range(P)|
   index(s)(p)/=currInd(s) OR
   index(s)(p)=currInd(s) AND (pc(s)(p)>=30 and pc(s)(
      p)<=41
                or pc(s)(p)>=46 and pc(s)(p)<=57
                or pc(s)(p)>=59 and pc(s)(p)<=65 and
                   return_getAccess(s)(p)>=30
                or pc(s)(p)>=67 and pc(s)(p)<=72 and
                   (return_releaseAccess(s)(p)=59 and
                       return_getAccess(s)(p)>=30
                      or return_releaseAccess(s)(p)=90
                         and return_refresh(s)(p)
                        >=30)
                or (pc(s)(p)=90 or pc(s)(p)>=104 and
                   pc(s)(p)<=105)
                      and return_refresh(s)(p)>=30)
                }

ocSet2(s:state):setof[range(P)]= {p:range(P)|
           pc(s)(p)>=125 and b_moveElement(s)(p)
                      and to_moveElement(s)(p)=H(s)(
                         currInd(s))}

ocSet3(s:state):setof[range(P)]= {p:range(P)|
           index(s)(p)=currInd(s) and pc(s)(p)=41 and
              suc_insert(s)(p) OR
           index(s)(p)=currInd(s) and pc(s)(p)=57 and
              r_assign(s)(p)=nor(null)}

ocSet4(s:{x:state|i?(Heap(x)(H(x)(currInd(x))))}):
   finite_set=
    {x:below(size(i_(Heap(s)(H(s)(currInd(s))))))| val
       ((Y(s))(x))/=null}

ocSet5(s:state14):finite_set=
    {x:below(size(i_(Heap(s)(H(s)(next(s)(currInd(s)))
       ))))|
           val(table_(i_(Heap(s)(H(s)(next(s)(currInd(
              s))))))(x))/=null}

ocSet6(s:state14):finite_set=
   {k:below(size(i_(Heap(s)(H(s)(next(s)(currInd(s))))
      )))|
          table_(i_(Heap(s)(H(s)(next(s)(currInd(s))))
           ))(k)/=nor(null)}

ocSet7(s:state14):setof[range(P)]= {p:range(P)|
           pc(s)(p)>=125 and b_moveElement(s)(p)
                     and to_moveElement(s)(p)=H(s)(
                        next(s)(currInd(s)))}

G_ghostset(s:{x:state| i?(Heap(x)(H(x)(currInd(x))))})
   :finite_set=
   {k:below(size(i_(Heap(s)(H(s)(currInd(s))))))|Y(s)(
      k)/=nor(null)}

G_prot_alter(s:state,p:range(P),i:range(2*P),j:range
   (2)):bool = COND
    j=1 -> index(s)(p)=i AND pc(s)(p)/=0 AND pc(s)(p)
       /=59 AND pc(s)(p)/=60,
    j=2 -> (index(s)(p)=i AND (pc(s)(p)=104 or pc(s)(p
       )=105))
           OR
           (i_releaseAccess(s)(p)=i AND index(s)(p)/=i
               AND
                pc(s)(p)>=67 AND pc(s)(p)<=72)
           OR
           (i_newTable(s)(p)=i AND pc(s)(p)>=81 AND pc
              (s)(p)<=84)
           OR
           (i_migrate(s)(p)=i AND pc(s)(p)>=97)
    ENDCOND

prSet1(s:state,i:range(2*P)): setof[range(P)]=
    {p:range(P)| G_prot_alter(s,p,i,1)}

prSet2(s:state,i:range(2*P)): setof[range(P)]=
    {p:range(P)| G_prot_alter(s,p,i,2)}

G_protsize(s:state,i:range(2*P)): nat=
   size(prSet1(s,i))+size(prSet2(s,i))
     + IF currInd(s)=i THEN 1 ELSE 0 ENDIF
     + IF next(s)(currInd(s))=i THEN 1 ELSE 0 ENDIF

G_protsize1(s:state,i:range(2*P)): nat=
   size(prSet1(s,i))+size(prSet2(s,i))

prSet3(s:state,i:range(2*P)): setof[range(P)]=
    {p:range(P)|  index(s)(p)=i and (pc(s)(p)>=61 and
       pc(s)(p)<=65
                          or pc(s)(p)=104 or pc(s)(p)
                             =105) OR
                  i_newTable(s)(p)=i and (pc(s)(p)=81
                     or pc(s)(p)=82) OR
                  i_releaseAccess(s)(p)=i and pc(s)(p)
                     =72 OR
                  i_migrate(s)(p)=i and (pc(s)(p)=97
                     or pc(s)(p)=98)}

prSet4(s:state,i:range(2*P)): setof[range(P)]=
    {p:range(P)|  index(s)(p)=i and (pc(s)(p)>=61 and
       pc(s)(p)<=65) OR
                  i_migrate(s)(p)=i and (pc(s)(p)=97
                     or pc(s)(p)=98)}

G_busy_alter(s:state,p:range(P),i:range(2*P),j:range
   (2)):bool = COND
    j=1 -> index(s)(p)=i AND (pc(s)(p)/=0 and pc(s)(p)
       <59 OR pc(s)(p)>62
           and pc(s)(p)/=65 and pc(s)(p)<=68 OR pc(s)(
              p)>68 and
            pc(s)(p)<=72 and return_releaseAccess(s)(p
              )>59 OR pc(s)(p)>72),
    j=2 -> (index(s)(p)=i AND pc(s)(p)=104)
           OR
           (i_releaseAccess(s)(p)=i AND index(s)(p)/=i
               AND
                pc(s)(p)>=67 AND pc(s)(p)<=68)
           OR
           (i_newTable(s)(p)=i AND pc(s)(p)>81 AND pc(
              s)(p)<=84)
           OR
           (i_migrate(s)(p)=i AND pc(s)(p)>=100)
    ENDCOND

buSet1(s:state,i:range(2*P)): setof[range(P)]=
    {p:range(P)| G_busy_alter(s,p,i,1)}

buSet2(s:state,i:range(2*P)): setof[range(P)]=
    {p:range(P)| G_busy_alter(s,p,i,2)}

G_busysize(s:state,i:range(2*P)): nat=
   size(buSet1(s,i))+size(buSet2(s,i))
     + IF currInd(s)=i THEN 1 ELSE 0 ENDIF
     + IF next(s)(currInd(s))=i THEN 1 ELSE 0 ENDIF

G_busysize1(s:state,i:range(2*P)): nat=
   size(buSet1(s,i))+size(buSet2(s,i))

%invariants on main correctness properties

Co1(s:state) : bool=
  forall (p:range(P)):
        pc(s)(p)=14 => val(r_find(s)(p))=rS_find(s)(p)

Co2(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)=25 or pc(s)(p)=26 => suc_delete(s)(p)=
        sucS_delete(s)(p)

Co3(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)=41 OR pc(s)(p)=42 => suc_insert(s)(p)=
        sucS_insert(s)(p)

Cn1(s:state) : bool=
  forall (p:range(P)):
      pc(s)(p)=14 => cnt_find(s)(p)=1

Cn2(s:state) : bool=
  forall (p:range(P)):
      pc(s)(p)=25 OR pc(s)(p)=26 => cnt_delete(s)(p)=1

Cn3(s:state) : bool=
  forall (p:range(P)):
      pc(s)(p)=41 OR pc(s)(p)=42 => cnt_insert(s)(p)=1
```

```
Cn4(s:state) : bool=
  forall (p:range(P)):
      pc(s)(p)=57 => cnt_assign(s)(p)=1

No1(s:state) : bool=
   card({ k : below(H_index(s)) | Heap(s)(k)/=bot })
      <=2*P

No2(s:state) : bool=
   card(nbSet1(s))=card(nbSet2(s))


%invariants about Heap

He1 (s:state):bool=
   NOT(i?(Heap(s)(0)))

He2_a (s:state):bool=
   forall (i:range(2*P)): H(s)(i)/=0 => Heap(s)(H(s)(i
      ))/=bot

He2_b (s:state):bool=
   forall (i:range(2*P)): i?(Heap(s)(H(s)(i))) => H(s)
      (i)/=0

He3_4(s):bool =
  i?(Heap(s)(H(s)(currInd(s)))) AND
  (forall (p:range(P)):
           pc(s)(p)/=0 and (pc(s)(p)<59 or (pc(s)(p)
              >65
                    and not (i_releaseAccess(s)(p)=
                       index(s)(p)
                             and (pc(s)(p)>=67 and pc(
                                s)(p)<=72)))=>
                    i?(Heap(s)(H(s)(index(s)(p)))))

He5(s):bool =
  forall (i:range(2*P)):
          i?(Heap(s)(H(s)(i)))=>
                     size(i_(Heap(s)(H(s)(i))))>= P
```

```
He6(s):bool =
  next(s)(currInd(s))/=0 => i?(Heap(s)(H(s)(next(s)(
     currInd(s)))))

Ha1 (s:state):bool=
% the range of null pointer is range(2*P).
   H_index(s)>0

Ha2 (s:state):bool=
% H_index is increasing, no pointer can beyond H_index
   forall (i:range(2*P)): H(s)(i)< H_index(s)

Ha3 (s:state):bool=
   forall (i,j:range(2*P)): i/=j and i?(Heap(s)(H(s)(i
      )))=>H(s)(i)/= H(s)(j)

Ha4(s):bool =
  forall (p:range(P)): index(s)(p)/=currInd(s)=>H(s)(
     index(s)(p))/=H(s)(currInd(s))

%invariants on counters for calling the specification

Cn5(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)>=6 and pc(s)(p)<=7  => cnt_find(s)(p)= 0

Cn6(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)>=8 and pc(s)(p)<=13 OR
     pc(s)(p)>=59 and pc(s)(p)<=65 and
        return_getAccess(s)(p)=10 OR
     pc(s)(p)>=67 and pc(s)(p)<=72 and
              (return_releaseAccess(s)(p)=59 and
                 return_getAccess(s)(p)=10
               or return_releaseAccess(s)(p)=90 and
                  return_refresh(s)(p)=10) OR
     pc(s)(p)>=90 and return_refresh(s)(p)=10
      =>
     cnt_find(s)(p)=
         IF r_find(s)(p)=nor(null) OR a_find(s)(p)=ADR
            (val(r_find(s)(p)))
         THEN 1  ELSE 0 ENDIF

Cn7(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)>=16 and pc(s)(p)<=21 and pc(s)(p)/=18 OR
     pc(s)(p)>=59 and pc(s)(p)<=65 and
        return_getAccess(s)(p)=20 OR
     pc(s)(p)>=67 and pc(s)(p)<=72 and
              (return_releaseAccess(s)(p)=59 and
                 return_getAccess(s)(p)=20
               or return_releaseAccess(s)(p)=90 and
                  return_refresh(s)(p)=20) OR
     pc(s)(p)>=90 and return_refresh(s)(p)=20
      =>
     cnt_delete(s)(p)= 0

Cn8(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)=18 =>
         cnt_delete(s)(p)= IF r_delete(s)(p)=nor(null)
             THEN 1  ELSE 0 ENDIF

Cn9(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)>=28 AND pc(s)(p)<=33 OR
     pc(s)(p)>=59 and pc(s)(p)<=65 and
        return_getAccess(s)(p)=30 OR
     pc(s)(p)>=67 and pc(s)(p)<=72 and
              (return_releaseAccess(s)(p)=59 and
                 return_getAccess(s)(p)=30
               or return_releaseAccess(s)(p)=77 and
                  return_newTable(s)(p)=30
               or return_releaseAccess(s)(p)=90 and
                  return_refresh(s)(p)=30) OR
     pc(s)(p)>=77 and pc(s)(p)<=84 and return_newTable
        (s)(p)=30 OR
     pc(s)(p)>=90 and return_refresh(s)(p)=30=>
        cnt_insert(s)(p)= 0

Cn10(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)>=35 and pc(s)(p)<=37 OR
     pc(s)(p)>=59 and pc(s)(p)<=65 and
        return_getAccess(s)(p)=36 OR
     pc(s)(p)>=67 and pc(s)(p)<=72 and
              (return_releaseAccess(s)(p)=59 and
                 return_getAccess(s)(p)=36
               or return_releaseAccess(s)(p)=90 and
                  return_refresh(s)(p)=36) OR
     pc(s)(p)>=90 and return_refresh(s)(p)=36
      =>
     cnt_insert(s)(p)=
         IF a_insert(s)(p)=ADR(val(r_insert(s)(p))) OR
             suc_insert(s)(p)
         THEN 1  ELSE 0 ENDIF

Cn11(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)>=44 and pc(s)(p)<=52 OR
     pc(s)(p)>=59 and pc(s)(p)<=65 and
          (return_getAccess(s)(p)=46 or
             return_getAccess(s)(p)=51) OR
     pc(s)(p)>=67 and pc(s)(p)<=72 and
          (return_releaseAccess(s)(p)=59 and (
             return_getAccess(s)(p)=46
               or return_getAccess(s)(p)=51) or
           return_releaseAccess(s)(p)=77 and
              return_newTable(s)(p)=46 or
           return_releaseAccess(s)(p)=90 and (
              return_refresh(s)(p)=46
               or return_refresh(s)(p)=51)) OR
     pc(s)(p)>=77 and pc(s)(p)<=84 and return_newTable
        (s)(p)=46 OR
     pc(s)(p)>=90 and (return_refresh(s)(p)=46 or
        return_refresh(s)(p)=51)
  => cnt_assign(s)(p)= 0

%invariants about old hashtable, current hashtable and
    the auxiliary hashtable Y

Cu1(s:state1):bool =
  forall (p:range(P)):
       pc(s)(p)/=0 AND (pc(s)(p)<59 or (pc(s)(p)>65
                                 and not (i_releaseAccess(
                                    s)(p)=index(s)(p)
                                          and (pc(s)(p)
                                             >=67 and pc(s
                                             )(p)<=72))))
             AND H(s)(index(s)(p))/=H(s)(currInd(s))=>
                  forall (k: below(size(i_(Heap(s)(H(s
                     )(index(s)(p))))))):
                            table_(i_(Heap(s)(H(s)(
                               index(s)(p)))))(k)=old(
                               null)

Cu2(s:state1):bool=
   card(G_ghostset(s))<size(i_(Heap(s)(H(s)(currInd(s)
     ))))

Cu3(s:state1):bool=
   bound(i_(Heap(s)(H(s)(currInd(s)))))+P*2 <
               size(i_(Heap(s)(H(s)(currInd(s)))))

Cu4(s:state1):bool=
   dels(i_(Heap(s)(H(s)(currInd(s)))))+size(deSet2(s))
     =card(deSet1(s))

Cu6(s:state1):bool=
   occ(i_(Heap(s)(H(s)(currInd(s)))))+size(ocSet1(s))+
     size(ocSet2(s))<=
          bound(i_(Heap(s)(H(s)(currInd(s)))))+P*2

Cu7(s:state1):bool=
   card(G_ghostset(s))=occ(i_(Heap(s)(H(s)(currInd(s))
     )))+
      size(ocSet2(s))+size(ocSet3(s))

Cu8(s:state1):bool =
     (exists (k: below(size(i_(Heap(s)(H(s)(currInd(
        s))))))):
                       oldp(table_(i_(Heap(s)(H(s)(
                          currInd(s)))))(k)))
       => next(s)(currInd(s))/=0
```

```
Cu9_10(s:state1):bool=
  forall (a:Address):
     (forall (n:below(l)):
        ( (NOT(oldp(r1)) => r1=Y(s)(n)) AND
          (oldp(r1) and val(r1)/=null =>val(Y(s)(n))=
             val(r1))
        )WHERE r1= table_(i_(Heap(s)(h)))(n)
     )WHERE h=H(s)(currInd(s)),l=size(i_(Heap(s)(h)))

Cu11_12(s:state1):bool =
  forall (a:Address):
       (forall (n:below(size(i_(Heap(s)(h))))):
          (
            (
             (forall (m: below(n)):
               (r1/=nor(null) AND (r1=del OR ADR(val(
                  r1))/=a))
                          WHERE
               r1=(Y(s))(KEY(a,size(i_(Heap(s)(h))),m)
                  ))
             AND
             (r2=nor(null) OR (r2/=del AND ADR(val(r2)
                )=a))
             => X(s)(a)=val(r2)
            )
            AND
            (
             X(s)(a)=val(r2) and X(s)(a)/=null =>
             (forall (m: below(n)):
               (r1/=nor(null) AND (r1=del OR ADR(val(
                  r1))/=a))
                          WHERE
               r1=(Y(s))(KEY(a,size(i_(Heap(s)(h))),m)
                  ))
             and
             (r2=nor(null) OR (r2/=del AND ADR(val(r2)
                )=a))
            )
        ) WHERE r2=(Y(s))(KEY(a,size(i_(Heap(s)(h))),n
           ))
```

```
      ) WHERE h=H(s)(currInd(s))

Cu13_14(s:state1):bool =
  (forall (a:Address):
       (X(s)(a)/=null => FORALL (i:below(l)):
                            X(s)(a)=val(Y(s)(KEY(a,l
                               ,i)))=>
                            not ((exists (j:below(l)
                               ):
                                j/=i AND ADR(val(Y(s
                                   )(KEY(a,l,j))))=a
                                   )))
       AND
       (X(s)(a)=null => not(exists (i:below(l)):
                          val(Y(s)(KEY(a,l,i)))/=
                             null AND
                          ADR(val(Y(s)(KEY(a,l,i))))
                             =a))
  )where l=size(i_(Heap(s)(H(s)(currInd(s)))))

Cu15(s:state1):bool =
  (forall (a:Address):
       (X(s)(a)/=null =>
                 exists (i:below(l)): X(s)(a)=val(Y(s)
                    (KEY(a,l,i)))
       )where l=size(i_(Heap(s)(H(s)(currInd(s))))))

Cu16(s:state1) : bool=
   EXISTS (f: [{x:below(size(i_(Heap(s)(H(s)(currInd(s
      ))))))|
                  val((Y(s))(x))/=null}->
               {v:Value|v/=null and
                   (EXISTS (k:below(size(i_(Heap(s)(H(
                      s)(currInd(s))))))):
                         v=val((Y(s))(k)))}]):
                            bijective?(f)

%invariants about next and next(currInd)

Ne1(s):bool =
  currInd(s)/=next(s)(currInd(s))
```

```
Ne2(s):bool =
  next(s)(currInd(s))/=0 => next(s)(next(s)(currInd(s)
     ))=0

Ne3(s):bool =
  forall (p:range(P)):
         (pc(s)(p)/=0 and (pc(s)(p)<59 or pc(s)(p)>=62
             and pc(s)(p)/=65))=>
                index(s)(p)/=next(s)(currInd(s))

Ne4(s):bool =
  forall (p:range(P)):
         (pc(s)(p)/=0 and (pc(s)(p)<59 or pc(s)(p)>=62
             and pc(s)(p)/=65))=>
                index(s)(p)/=next(s)(index(s)(p))

Ne5(s):bool =
  forall (p:range(P)):
     (pc(s)(p)/=0 and (pc(s)(p)<59 or pc(s)(p)>=62 and
         pc(s)(p)/=65))
           and next(s)(index(s)(p))=0 => index(s)(p)=
              currInd(s)

Ne6(s:state3):bool=
    next(s)(currInd(s))/=0=> card(ocSet6(s))<=card(
       G_ghostset(s))-
              dels(i_(Heap(s)(H(s)(currInd(s)))))-
                 size(deSet2(s))

Ne7(s:state3):bool=
   next(s)(currInd(s))/=0=> bound(i_(Heap(s)(H(s)(
      currInd(s)))))
           +P*2-dels(i_(Heap(s)(H(s)(currInd(s)))))<=
                bound(i_(Heap(s)(H(s)(next(s)(currInd(
                  s))))))

Ne8(s:state2):bool=
    next(s)(currInd(s))/=0=>
           bound(i_(Heap(s)(H(s)(next(s)(currInd(s))))
              ))+P*2 <
```

```
                  size(i_(Heap(s)(H(s)(next(s)(currInd(
                     s))))))

Ne9(s:state2):bool=
   next(s)(currInd(s))/=0=>
        dels(i_(Heap(s)(H(s)(next(s)(currInd(s))))))=
           card(deSet3(s))

Ne9a(s:state2):bool=
   next(s)(currInd(s))/=0=>
        dels(i_(Heap(s)(H(s)(next(s)(currInd(s))))))=
           0

Ne10(s:state2):bool =
     next(s)(currInd(s))/=0 =>
           (NOT (EXISTS (k:below(l)): table_(i_(Heap(s
              )(h)))(k)=del OR
                  table_(i_(Heap(s)(h)))(k)=old(null))
           )where h=H(s)(next(s)(currInd(s))), l=size(
              i_(Heap(s)(h)))

Ne11(s:state3):bool =
      next(s)(currInd(s))/=0 =>
        (forall (k:below(size(i_(Heap(s)(h))))):
                 not(oldp(table_(i_(Heap(s)(h)))(k)))
        )WHERE h=H(s)(next(s)(currInd(s)))

Ne12_13(s:state15):bool =
     forall (k: below(size(i_(Heap(s)(H(s)(currInd(s))
       ))))):
        table_(i_(Heap(s)(H(s)(currInd(s)))))(k)=old(
           null) =>
        forall (n:below(size(i_(Heap(s)(H(s)(next(s)(
           currInd(s)))))))):
         (  ( (forall (m: below(n)):
               (r1/=nor(null) AND (r1=del OR ADR(val(
                  r1))/=a))WHERE
                         r1=table_(i_(Heap(s)(h)))(KEY
                            (a,l,m)))
               AND
               (r2=nor(null) OR (r2/=del AND ADR(val(
                r2))=a))
              => X(s)(a)=val(r2))
           AND
            ( X(s)(a)=val(r2) and X(s)(a)/=null=>
               (forall (m: below(n)):
                 (r1/=nor(null) AND (r1=del OR ADR(val
                    (r1))/=a)) WHERE
                           r1=table_(i_(Heap(s)(h)))(
                              KEY(a,l,m)))
               AND
               (r2=nor(null) OR (r2/=del AND ADR(val(
                  r2))=a)))
         )WHERE h=H(s)(next(s)(currInd(s))), a=ADR(val
            (Y(s)(k))),
               l=size(i_(Heap(s)(h))),
               r2=table_(i_(Heap(s)(h)))(KEY(a,l,n))

Ne14(s:state3):bool =
   forall (a:Address):
     next(s)(currInd(s))/=0 AND a/=nullAddress=>
        (FORALL (n:below(l)):
            (X(s)(a)=val(r2) and X(s)(a)/=null=>
               (forall (m: below(n)):
                 (r1/=nor(null) AND (r1=del OR ADR(val
                    (r1))/=a)) WHERE
                           r1=table_(i_(Heap(s)(h)))(
                              KEY(a,l,m)))
                 AND
               (r2=nor(null) OR (r2/=del AND ADR(val(
                  r2))=a))
            )WHERE r2=table_(i_(Heap(s)(h)))(KEY(a,l,n
               ))
        )WHERE h=H(s)(next(s)(currInd(s))), l=size(i_(
           Heap(s)(h)))

Ne15_16(s:state15):bool =
     forall (k: below(size(i_(Heap(s)(H(s)(currInd(s))
        ))))):
      table_(i_(Heap(s)(H(s)(currInd(s)))))(k)=old(
        null) =>
       ((X(s)(a)/=null =>
               FORALL (i:below(l)):
                  X(s)(a)=val(table_(i_(Heap(s)(h)))(
                     KEY(a,l,i)))=>
                       not ((exists (j:below(l)):
                   j/=i AND ADR(val(table_(i_(Heap(s)(
                      h)))(KEY(a,l,j))))=a)))
       AND
       (X(s)(a)=null => not(exists (i:below(l)):
                            val(table_(i_(Heap(s)(h)))
                              (KEY(a,l,i)))/=null AND
                            ADR(val(table_(i_(Heap(s)(
                               h)))(KEY(a,l,i))))=a))
      )where a=ADR(val(Y(s)(k))), h=H(s)(next(s)(
         currInd(s))),
             l=size(i_(Heap(s)(h)))

Ne17_18(s:state3):bool =
   forall (a:Address):
       next(s)(currInd(s))/=0 AND a/=nullAddress=>
          (FORALL (i:below(l1)):
              ADR(val(table_(i_(Heap(s)(h1)))(i)))=a
                 =>
                exists (j:below(l2)):
                  val(Y(s)(j))=val(table_(i_(Heap(s)(
                      h1)))(i)) AND
                              oldp(table_(i_(Heap(s)(
                                 h2)))(j)) AND
                                  X(s)(a)=val(table_(
                                     i_(Heap(s)(h1)))(
                                     i))
                                         AND X(s)(a)/=
                                            null
           )where h1=H(s)(next(s)(currInd(s))), h2=H(s
              )(currInd(s)),
                  l1=size(i_(Heap(s)(h1))),l2=size(i_(
                     Heap(s)(h2)))

Ne19(s:state3):bool =
   forall (a:Address):
     next(s)(currInd(s))/=0 AND a/=nullAddress=>
     (FORALL (i:below(l)):
            ADR(val(table_(i_(Heap(s)(h)))(KEY(a,l,i))
               ))=a=>
                   not ((exists (j:below(l)):  j/=i
                      AND
                      ADR(val(table_(i_(Heap(s)(h)))(
                         KEY(a,l,j))))=a))
     )WHERE h=H(s)(next(s)(currInd(s))), l=size(i_(
        Heap(s)(h)))

Ne20(s:state15):bool =
     forall (k: below(size(i_(Heap(s)(H(s)(currInd(s))
        ))))):
       table_(i_(Heap(s)(H(s)(currInd(s)))))(k)=old(
          null)
         and X(s)(ADR(val(Y(s)(k))))/=null =>
          (exists (i:below(l)): X(s)(a)=val(table_(i_(
             Heap(s)(h)))(KEY(a,l,i)))
           )where a=ADR(val(Y(s)(k))), h=H(s)(next(s)(
              currInd(s))),
                  l=size(i_(Heap(s)(h)))

Ne22(s:state2):bool=
   next(s)(currInd(s))/=0=> card(ocSet6(s))=
       occ(i_(Heap(s)(H(s)(next(s)(currInd(s))))))+
          size(ocSet7(s))

Ne23(s:state3):bool=
 forall (p: range(P) ) :
    next(s)(currInd(s))/=0 =>
        occ(i_(Heap(s)(H(s)(next(s)(currInd(s))))))<=
          bound(i_(Heap(s)(H(s)(next(s)(currInd(s)))))
            )

Ne24(s:state3) : bool=
   next(s)(currInd(s))/=0 => card(ocSet5(s))<=  card(
      ocSet4(s))

Ne25(s:state2) : bool=
   next(s)(currInd(s))/=0=>
   (EXISTS (f: [{x:below(size(h1))| val(table_(h1)(x))
      /=null} ->
```

```
                {v:Value|v/=null and
                    (EXISTS (k:below(size(h1))):
                          v=val(table_(h1)(k)))}]):
                             bijective?(f)
   )WHERE h1=i_(Heap(s)(H(s)(next(s)(currInd(s)))))

Ne26(s:state3) : bool=
   next(s)(currInd(s))/=0=>
   (EXISTS (f: [{v:Value| v/=null and
                   (EXISTS (k:below(size(h1))): v=val(
                      table_(h1)(k)))} ->
                 {v:Value| v/=null and
                   (EXISTS (k:below(size(h2))): v=val(Y
                      (s)(k)))}]):injective?(f)
   )WHERE h1=i_(Heap(s)(H(s)(next(s)(currInd(s))))),
          h2=i_(Heap(s)(H(s)(currInd(s))))

Ne27(s:state3) : bool=
   next(s)(currInd(s))/=0 and
       ((EXISTS (k: below(size(h1))): val(table_(h1)(k
          )) /= null)
        WHERE h1=i_(Heap(s)(H(s)(next(s)(currInd(s))))
           ))
     =>
    ((EXISTS (f: [{x:below(size(h1))| val(table_(h1)(x
       ))/=null} ->
                 {x:below(size(h2))| val((Y(s))(x))/=
                   null}]):injective?(f))
     WHERE h1=i_(Heap(s)(H(s)(next(s)(currInd(s))))),
           h2=i_(Heap(s)(H(s)(currInd(s)))))

%invariants about find

fi1_(s): bool=
   forall (p:range(P)):
          a_insert(s)(p)=ADR(v_insert(s)(p)) AND
             a_assign(s)(p)=ADR(v_assign(s)(p))
          AND
          v_assign(s)(p)/=null and v_insert(s)(p)/=
             null
          AND
          a_find(s)(p)/=nullAddress and a_delete(s)(p)
             /=nullAddress

fi2(s:state):bool =
  forall (p:range(P)):
         pc(s)(p)=6 or pc(s)(p)=11 =>n_find(s)(p)=0

fi3(s:state11):bool =
  forall (p:range(P)):
         pc(s)(p)=7 OR pc(s)(p)=8 OR pc(s)(p)=13 =>
                 l_find(s)(p)=size(i_(Heap(s)(h_find(s
                    )(p))))

fi4(s:state):bool =
  forall (p:range(P)):
         (pc(s)(p)>=6 AND pc(s)(p)<=13 AND pc(s)(p)
            /=10 =>
              h_find(s)(p)=H(s)(index(s)(p)))

fi5_(s:state1): bool=
   forall (p:range(P)):
      (pc(s)(p)=7 and h_find(s)(p)=H(s)(currInd(s)) =>
              n_find(s)(p)<size(i_(Heap(s)(H(s)(
                 currInd(s)))))) AND
      ((pc(s)(p)=17 or pc(s)(p)=18) and h_delete(s)(p)
         =H(s)(currInd(s))=>
              n_delete(s)(p)<size(i_(Heap(s)(H(s)(
                 currInd(s)))))) AND
      ((pc(s)(p)=33 or pc(s)(p)=35) and h_insert(s)(p)
         =H(s)(currInd(s))=>
              n_insert(s)(p)<size(i_(Heap(s)(H(s)(
                 currInd(s)))))) AND
      (pc(s)(p)=50 and h_assign(s)(p)=H(s)(currInd(s))
         =>
              n_assign(s)(p)<size(i_(Heap(s)(H(s)(
                 currInd(s))))))

fi6(s:state1):bool =
  forall (p:range(P)):
      pc(s)(p)=8 AND h_find(s)(p)=H(s)(currInd(s)) AND
       ((r1/=nor(null) AND r1/=old(null) AND (r1=del
           OR ADR(val(r1))/=a_find(s)(p)))
               WHERE r1=r_find(s)(p))
           =>
        (r2/=nor(null) AND (r2=del OR ADR(val(r2))/=a))
                WHERE l=size(i_(Heap(s)(H(s)(currInd(s)
                  )))),n=n_find(s)(p),
                     a=a_find(s)(p), r2=Y(s)(KEY(a,l,n
                        ))

fi7(s:state1):bool =
  forall (p:range(P)):
         pc(s)(p)=13 AND h_find(s)(p)=H(s)(currInd(s))
            AND
         ((r1/=nor(null) AND (r1=old(null) OR r1=del
            OR ADR(val(r1))/=a_find(s)(p)))
               WHERE r1=r_find(s)(p))
            =>
           (forall (m:below(n_find(s)(p))):
              (r2/=nor(null) AND (r2=del OR ADR(val(r2
                 ))/=a))
                  WHERE l=size(i_(Heap(s)(H(s)(currInd(
                     s))))) ,n=n_find(s)(p),
                        a=a_find(s)(p), r2=Y(s)(KEY(a,l
                          ,m)))

fi8(s:state1):bool =
  forall (p:range(P)):
      (pc(s)(p)=7 or pc(s)(p)=8) AND h_find(s)(p)=H(s)
         (currInd(s)) =>
             (forall (m:below(n_find(s)(p))):
                 (r1/=nor(null) AND (r1=del OR ADR(val
                    (r1))/=a))
               WHERE l=size(i_(Heap(s)(H(s)(currInd(s)
                  )))),n=n_find(s)(p),
                     a=a_find(s)(p), r1=Y(s)(KEY(a,l,m
                        )))

fi9(s:state12):bool =
  forall (p:range(P)):
     pc(s)(p)=7 AND ((r=nor(null) OR (r/=old(null) AND
         r/=del AND ADR(val(r))=a))
```

```
          WHERE h=Heap(s)(h_find(s)(p)), a=a_find(s)(p)
             ,
                              l=l_find(s)(p), n=n_find
                                 (s)(p),
                              r=table_(i_(h))(KEY(a,l,
                                 n)))
             =>
        ((X(s)(a)=val(r)) WHERE h=Heap(s)(h_find(s)(p))
           , a=a_find(s)(p),
                              l=l_find(s)(p), n=n_find
                                 (s)(p),
                              r=table_(i_(h))(KEY(a,l,
                                 n)))

fi10(s:state) : bool=
  forall (p:range(P)):
         (  (pc(s)(p)<=1 or pc(s)(p)>7) AND
                    (r=nor(null) OR (r/=old(null) AND
                       r/=del AND a=ADR(val(r))))
            =>val(r)=rS_find(s)(p)
          )WHERE r=r_find(s)(p), a=a_find(s)(p)

fi11_(s): bool=
   forall (p:range(P)): (pc(s)(p)=8 and oldp(r_find(s)
      (p)) OR
       pc(s)(p)=18 and oldp(r_delete(s)(p)) OR pc(s)(p
          )=35 and oldp(r_insert(s)(p))
           OR pc(s)(p)=50 and oldp(r_assign(s)(p)))
              AND index(s)(p)=currInd(s) =>
                     next(s)(currInd(s))/=0

%invariants about delete

de2(s:state11):bool =
  forall (p:range(P)):
         pc(s)(p)>=17 AND pc(s)(p)<=18 =>
                l_delete(s)(p)=size(i_(Heap(s)(
                   h_delete(s)(p))))

de3(s:state):bool =
  forall (p:range(P)):
        (pc(s)(p)>=16 AND pc(s)(p)<=25 AND pc(s)(p)
           /=20 =>
              h_delete(s)(p)=H(s)(index(s)(p)))

de4_(s:state) : bool=
  forall (p:range(P)):
      (pc(s)(p)=18 =>
            k_delete(s)(p)=KEY(a_delete(s)(p),l_delete
               (s)(p),n_delete(s)(p)))
        AND
      (pc(s)(p)=33 OR pc(s)(p)=35 =>
            k_insert(s)(p)=KEY(a_insert(s)(p),l_insert
               (s)(p),n_insert(s)(p)))
        AND
      (pc(s)(p)=49 OR pc(s)(p)=50 =>
            k_assign(s)(p)=KEY(a_assign(s)(p),l_assign
               (s)(p),n_assign(s)(p)))

de5(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)>=16 and pc(s)(p)<=21 OR
     pc(s)(p)>=59 and pc(s)(p)<=65 and
        return_getAccess(s)(p)=20 OR
     pc(s)(p)>=67 and pc(s)(p)<=72 and
              (return_releaseAccess(s)(p)=59 and
                 return_getAccess(s)(p)=20
               or return_releaseAccess(s)(p)=90 and
                  return_refresh(s)(p)=20) OR
     pc(s)(p)>=90 and return_refresh(s)(p)=20
      => not(suc_delete(s)(p))

de6(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)>=18 and pc(s)(p)<=21 OR
     pc(s)(p)>=59 and pc(s)(p)<=65 and
        return_getAccess(s)(p)=20 OR
     pc(s)(p)>=67 and pc(s)(p)<=72 and
              (return_releaseAccess(s)(p)=59 and
                 return_getAccess(s)(p)=20
               or return_releaseAccess(s)(p)=90 and
                  return_refresh(s)(p)=20) OR
     pc(s)(p)>=90 and return_refresh(s)(p)=20
       =>
     (sucS_delete(s)(p)=>r_delete(s)(p)/=nor(null))

de7_(s:state10):bool =
  forall (p:range(P)):
      (pc(s)(p)=18 and  not(oldp(table_(i_(Heap(s)(
         h_delete(s)(p))))
               (k_delete(s)(p))))=>h_delete(s)(p)=H(s)
                  (currInd(s)))
      AND
      (pc(s)(p)=35 and  not(oldp(table_(i_(Heap(s)(
         h_insert(s)(p))))
               (k_insert(s)(p))))=>h_insert(s)(p)=H(s)
                  (currInd(s)))
      AND
      (pc(s)(p)=50 and  not(oldp(table_(i_(Heap(s)(
         h_assign(s)(p))))
               (k_assign(s)(p))))=>h_assign(s)(p)=H(s)
                  (currInd(s)))

de9(s:state11):bool =
  forall (p:range(P)):
      pc(s)(p)=18 and h_delete(s)(p)=H(s)(currInd(s))
         and
           (val(r_delete(s)(p))/=null or r_delete(s)(p
              )=del)=>
          (r1/=nor(null) and (r1=del OR ADR(val(r1))=
             ADR(val(r_delete(s)(p))))
          )WHERE h=Heap(s)(h_delete(s)(p)), a=a_delete
             (s)(p),
                l=size(i_(h)), n=n_delete(s)(p), r1=Y(
                   s)(KEY(a,l,n))

de10(s:state1):bool =
  forall (p:range(P)):
    (pc(s)(p)=17 OR pc(s)(p)=18) AND h_delete(s)(p)=H(
       s)(currInd(s))
        =>
      (forall (m:below(n_delete(s)(p))):
             (r1/=nor(null) AND (r1=del OR ADR(val(r1)
                )/=a))
        WHERE l=size(i_(Heap(s)(H(s)(currInd(s))))),n=
           n_delete(s)(p),
                a=a_delete(s)(p), r1=Y(s)(KEY(a,l,m)))

de11(s:state12):bool =
  forall (p:range(P)):
     (pc(s)(p)>=17 and pc(s)(p)<=18) AND ((r=nor(null)
         OR
                          (r/=old(null) AND r/=del AND
                             ADR(val(r))=a))
              WHERE h=Heap(s)(h_delete(s)(p)), a=
                 a_delete(s)(p),
                    l=l_delete(s)(p), n=n_delete(s)(p),
                       r=table_(i_(h))(KEY(a,l,n)))
              =>
     ((X(s)(a)=val(r))
      WHERE h=Heap(s)(h_delete(s)(p)), a=a_delete(s)(p
         ), l=l_delete(s)(p),
            n=n_delete(s)(p), r=table_(i_(h))(KEY(a,l,
              n)))

de13_(s:state1):bool =
  forall (p:range(P)):
      (pc(s)(p)=18 => k_delete(s)(p)<size(i_(Heap(s)(H
         (s)(index(s)(p))))))
        AND
      (pc(s)(p)=35 => k_insert(s)(p)<size(i_(Heap(s)(H
         (s)(index(s)(p))))))
        AND
      (pc(s)(p)=50 => k_assign(s)(p)<size(i_(Heap(s)(H
         (s)(index(s)(p))))))

%invariants about insert

in2(s:state11):bool =
  forall (p:range(P)):
         pc(s)(p)>=32 AND pc(s)(p)<=35 =>
                 l_insert(s)(p)=size(i_(Heap(s)(
                    h_insert(s)(p))))
```

```
in3(s:state):bool =
  forall (p:range(P)):
         (pc(s)(p)>=28 AND pc(s)(p)<=41 AND pc(s)(p)
            /=30 and pc(s)(p)/=36 =>
               h_insert(s)(p)=H(s)(index(s)(p)))

in5(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)>=32 and pc(s)(p)<=37 OR
     pc(s)(p)>=59 and pc(s)(p)<=65 and
        return_getAccess(s)(p)=36 OR
     pc(s)(p)>=67 and pc(s)(p)<=72 and
              (return_releaseAccess(s)(p)=59 and
                 return_getAccess(s)(p)=36
               or return_releaseAccess(s)(p)=90 and
                  return_refresh(s)(p)=36) OR
     pc(s)(p)>=90 and return_refresh(s)(p)=36
      => not(suc_insert(s)(p))

in6(s:state) : bool=
  forall (p:range(P)):
     pc(s)(p)>=35 and pc(s)(p)<=37 OR
     pc(s)(p)>=59 and pc(s)(p)<=65 and
        return_getAccess(s)(p)=36 OR
     pc(s)(p)>=67 and pc(s)(p)<=72 and
              (return_releaseAccess(s)(p)=59 and
                 return_getAccess(s)(p)=36
               or return_releaseAccess(s)(p)=90 and
                  return_refresh(s)(p)=36) OR
     pc(s)(p)>=90 and return_refresh(s)(p)=36
       =>
    (sucS_insert(s)(p)=>(ADR(val(r_insert(s)(p)))/=
       a_insert(s)(p)))

in9(s:state11):bool =
  forall (p:range(P)):
      pc(s)(p)=35 and h_insert(s)(p)=H(s)(currInd(s))
         and
            (val(r_insert(s)(p))/=null or r_insert(s)(p
              )=del)=>
           (r1/=nor(null) and (r1=del OR ADR(val(r1))=
             ADR(val(r_insert(s)(p))))
          )WHERE h=Heap(s)(h_insert(s)(p)), a=a_insert
             (s)(p),
                 l=size(i_(h)), n=n_insert(s)(p), r1=Y
                    (s)(KEY(a,l,n))

in10(s:state1):bool =
  forall (p:range(P)):
    (pc(s)(p)=32 OR pc(s)(p)=33 or pc(s)(p)=35) AND
       h_insert(s)(p)=H(s)(currInd(s))
        =>
      (forall (m:below(n_insert(s)(p))):
             (r1/=nor(null) AND (r1=del OR ADR(val(r1)
               )/=a))
       WHERE l=size(i_(Heap(s)(H(s)(currInd(s))))),n=
          n_insert(s)(p),
                a=a_insert(s)(p), r1=Y(s)(KEY(a,l,m)))

in11(s:state12):bool =
  forall (p:range(P)):
     (pc(s)(p)=33 or pc(s)(p)=35) AND
        ((r=nor(null) OR (r/=old(null) AND r/=del AND
           ADR(val(r))=a))
        WHERE h=Heap(s)(h_insert(s)(p)), a=a_insert(s)
           (p), l=l_insert(s)(p),
              n=n_insert(s)(p), r=table_(i_(h))(KEY(a,
                 l,n)))
             =>
     ((X(s)(a)=val(r) )
      WHERE h=Heap(s)(h_insert(s)(p)), a=a_insert(s)(p
         ), l=l_insert(s)(p),
            n=n_insert(s)(p), r=table_(i_(h))(KEY(a,l,
              n)))

%invariants about assign

as2(s:state11):bool =
  forall (p:range(P)):
         pc(s)(p)>=48 AND pc(s)(p)<=50 =>
                l_assign(s)(p)=size(i_(Heap(s)(
                   h_assign(s)(p))))
```

```
as3(s:state):bool =
  forall (p:range(P)):
         (pc(s)(p)>=44 AND pc(s)(p)<=57 AND pc(s)(p)
            /=46 and pc(s)(p)/=51 =>
               h_assign(s)(p)=H(s)(index(s)(p)))

as7(s:state11):bool =
  forall (p:range(P)):
      pc(s)(p)=50 and h_assign(s)(p)=H(s)(currInd(s))
         and
           (val(r_assign(s)(p))/=null or r_assign(s)(p
              )=del)=>
          (r1/=nor(null) and (r1=del OR ADR(val(r1))=
             ADR(val(r_assign(s)(p))))
          )WHERE h=Heap(s)(h_assign(s)(p)), a=a_assign
             (s)(p),
                 l=size(i_(h)), n=n_assign(s)(p), r1=Y
                    (s)(KEY(a,l,n))

as8(s:state1):bool =
  forall (p:range(P)):
      (pc(s)(p)=48 or pc(s)(p)=49 or pc(s)(p)=50) AND
           h_assign(s)(p)=H(s)(currInd(s)) =>
             (forall (m:below(n_assign(s)(p))):
                 (r1/=nor(null) AND (r1=del OR ADR(val
                    (r1))/=a))
               WHERE l=size(i_(Heap(s)(H(s)(currInd(s)
                  )))),n=n_assign(s)(p),
                     a=a_assign(s)(p), r1=Y(s)(KEY(a,l
                        ,m)))

as9(s:state12):bool =
  forall (p:range(P)):
     pc(s)(p)=50 AND ((r=nor(null) OR (r/=old(null)
        AND r/=del AND ADR(val(r))=a))
         WHERE h=Heap(s)(h_assign(s)(p)), a=a_assign(s
            )(p), l=l_assign(s)(p),
               n=n_assign(s)(p), r=table_(i_(h))(KEY(a
                  ,l,n)))
            =>
     ((X(s)(a)=val(r))
      WHERE h=Heap(s)(h_assign(s)(p)), a=a_assign(s)(p
         ), l=l_assign(s)(p),
            n=n_assign(s)(p), r=table_(i_(h))(KEY(a,l,
               n)))

%invariants about releaseAccess

rA1 (s:state):bool=
   forall (p:range(P)): h_releaseAccess(s)(p)< H_index
      (s)

rA2(s:state):bool=
   forall (p:range(P)): pc(s)(p)>=70 and pc(s)(p)<=71
      =>h_releaseAccess(s)(p)/=0

rA3(s): bool=
  forall (p:range(P)):
      pc(s)(p)=71=>  i?(Heap(s)(h_releaseAccess(s)(p))
         )

rA4 (s:state):bool=
   forall (p:range(P)): pc(s)(p)=71 => H(s)(
      i_releaseAccess(s)(p))=0

rA5(s:state):bool=
   forall (p:range(P)):
         pc(s)(p)=71 => NOT(EXISTS (i:range(2*P)):
            h_releaseAccess(s)(p)=H(s)(i))

rA6_7(s:state):bool =
  forall (p:range(P)):
      (pc(s)(p)=70 =>
             H(s)(i_releaseAccess(s)(p))/=H(s)(currInd
                (s)) AND
             (not (exists (r:range(P)): pc(s)(r)/=0
                and (pc(s)(r)<59 or (pc(s)(r)>65
                    and not (i_releaseAccess(s)(r)=
                       index(s)(r) and (pc(s)(r)>=67
                                   and pc(s)(r)<=72))
                                      ))
```

```
                     and H(s)(i_releaseAccess(s)(p))=H(
                        s)(index(s)(r)))))

rA8(s:state):bool =
  forall (p:range(P)):
     pc(s)(p)=70 => i_releaseAccess(s)(p)/=next(s)(
        currInd(s))

rA9(s:state):bool =
  forall (p:range(P)):
      pc(s)(p)>=68 and pc(s)(p)<=72 and  (
         h_releaseAccess(s)(p)=0 OR
           h_releaseAccess(s)(p)/= H(s)(
              i_releaseAccess(s)(p)))
              => H(s)(i_releaseAccess(s)(p))=0

rA10(s:state):bool =
  forall (p:range(P)):
     pc(s)(p)>=67 AND pc(s)(p)<=72 AND
          (return_releaseAccess(s)(p)=0 OR
             return_releaseAccess(s)(p)=59)
              => i_releaseAccess(s)(p)=index(s)(p)

rA11(s:state):bool =
 forall (p:range(P)):
        pc(s)(p)>=67 AND pc(s)(p)<=72 AND
                (return_releaseAccess(s)(p)=77 OR
                   return_releaseAccess(s)(p)=90)
                 =>i_releaseAccess(s)(p)/=index(s)(p)

rA12(s): bool=
  forall (p:range(P)):
      pc(s)(p)>=67 AND pc(s)(p)<=72 and
         return_releaseAccess(s)(p)=77 =>
            next(s)(index(s)(p))/=0

rA13(s:state):bool =
  forall (p:range(P)):
     pc(s)(p)=71 => not (exists (r:range(P)):
            p/=r AND pc(s)(r)=71 AND h_releaseAccess(s
               )(p)=h_releaseAccess(s)(r))
```

```
rA14(s:state):bool =
  forall (p:range(P)):
     pc(s)(p)=71 => not (exists (r:range(P)):
            p/=r AND pc(s)(r)=71 AND i_releaseAccess(s
              )(p)=i_releaseAccess(s)(r))

%invariants about newTable

nT1(s:state) : bool=
  forall (p:range(P)):
        pc(s)(p)>=81 and pc(s)(p)<=82 => Heap(s)(H(s)(
           i_newTable(s)(p)))=bot

nT2(s:state):bool =
  forall (p:range(P)):
      pc(s)(p)>=83 and pc(s)(p)<=84 => i?(Heap(s)(H(s)
         (i_newTable(s)(p))))

nT3(s):bool =
  forall (p:range(P)):
     pc(s)(p)=84 => next(s)(i_newTable(s)(p))=0

nT4(s:state5):bool =
  forall (p:range(P)):
      pc(s)(p)>=83 and pc(s)(p)<=84 =>
        (dels(h)=0) WHERE h=i_(Heap(s)(H(s)(i_newTable
           (s)(p))))

nT5(s:state5):bool =
  forall (p:range(P)):
      pc(s)(p)>=83 and pc(s)(p)<=84 =>
        (occ(h)=0
        )WHERE h=i_(Heap(s)(H(s)(i_newTable(s)(p))))

nT6(s:state5):bool =
  forall (p:range(P)):
      pc(s)(p)>=83 and pc(s)(p)<=84 =>
          (bound(h)+2*P<size(h)
          )WHERE h=i_(Heap(s)(H(s)(i_newTable(s)(p))))
```

```
nT7(s:state5):bool =
  forall (p:range(P)):
      (pc(s)(p)>=83 and pc(s)(p)<=84) and index(s)(p)=
         currInd(s)
        =>  (bound(h1)+2*P-dels(h1)<bound(h2)
            )WHERE h1=i_(Heap(s)(H(s)(currInd(s)))),
                   h2=i_(Heap(s)(H(s)(i_newTable(s)(p)
                     )))

nT8(s:state5):bool =
  forall (p:range(P)):
      pc(s)(p)>=83 and pc(s)(p)<=84  =>
          (forall (n:below(size(h))): table_(h)(n)=nor
             (null)
          )WHERE h=i_(Heap(s)(H(s)(i_newTable(s)(p))))

nT9_10(s): bool=
   forall (p:range(P)):
     pc(s)(p)>=81 and pc(s)(p)<=84  => i_newTable(s)(p
        )/=currInd(s) AND
         (not exists (r:range(P)):
              pc(s)(r)/=0 and (pc(s)(r)<59 or pc(s)(r)
                 >=62 and pc(s)(r)/=65)
                   and i_newTable(s)(p)=index(s)(r)
                     )

nT11(s): bool=
   forall (p:range(P)):
     pc(s)(p)>=81 and pc(s)(p)<=84  => i_newTable(s)(p
        )/=next(s)(currInd(s))

nT12_13(s:state):bool =
  forall (p:range(P)):
      (pc(s)(p)>=81 and pc(s)(p)<=84  =>
         (not (exists (r:range(P)): pc(s)(r)/=0 and (
            pc(s)(r)<59 or (pc(s)(r)>65
                    and not (i_releaseAccess(s)(r)=
                       index(s)(r) and (pc(s)(r)>=67
                                    and pc(s)(r)<=72))
                                       ))
                    and H(s)(i_newTable(s)(p))=H(s)(
                    index(s)(r))))
          and
          H(s)(i_newTable(s)(p))/=H(s)(currInd(s)))

nT14(s): bool=
   forall (p:range(P)):
       pc(s)(p)>=81 and pc(s)(p)<=84 =>
             not (exists (r:range(P)):
                   pc(s)(r)>=67 and pc(s)(r)<=72 and
                                  i_newTable(s)(p)=
                                     i_releaseAccess(s
                                     )(r))

nT15(s:state):bool =
  forall (p:range(P)):
      pc(s)(p)>=83 AND pc(s)(p)<=84  =>
             (not (exists (r:range(P)): pc(s)(r)>=67
                and pc(s)(r)<=72 AND
                             H(s)(i_newTable(s)(p))=H(
                                s)(i_releaseAccess(s)(
                                r))))

nT16(s): bool=
   forall (p:range(P)):
       pc(s)(p)>=81 and pc(s)(p)<=84 =>
             not (exists (r:range(P)):
                   pc(s)(r)>=81 and pc(s)(r)<=84 and p
                     /=r and
                                  i_newTable(s)(p)=
                                     i_newTable(s)(r))

nT17(s): bool=
   forall (p:range(P)):
       pc(s)(p)>=81 and pc(s)(p)<=84 =>
               not (exists (r:range(P)): pc(s)(r)>=95
                  AND pc(s)(r) <= 99 AND
                     index(s)(r) = currInd(s) and
                        i_newTable(s)(p)=i_migrate(s)(
                        r))

nT18(s): bool=
  forall (p:range(P)):
     pc(s)(p)>=81 and pc(s)(p)<=84 =>
        not(exists (r:range(P)): pc(s)(r)>=99 and
           i_newTable(s)(p)=i_migrate(s)(r))

%invariants about migrate

mi1_a(s:state):bool =
  forall (p:range(P)):
      pc(s)(p)=98 => index(s)(p)/=currInd(s)

mi1_b(s):bool =
  forall (p:range(P)):
     pc(s)(p)=104 OR pc(s)(p)=105 =>index(s)(p)/=
        currInd(s)

mi2(s): bool=
   forall (p:range(P)):
       pc(s)(p)>=95 => i_migrate(s)(p)/=index(s)(p)

mi3(s): bool=
   forall (p:range(P)):
       pc(s)(p)=94 =>next(s)(index(s)(p))>0

mi4(s): bool=
   forall (p:range(P)):
       pc(s)(p)>=95 => i_migrate(s)(p)/=0

mi5(s): bool=
   forall (p:range(P)):
        pc(s)(p)>=95  => i_migrate(s)(p)=next(s)(index
           (s)(p))

mi6_a(s): bool=
   forall (p:range(P)):
       pc(s)(p)>=102 AND pc(s)(p)<=103 OR pc(s)(p)
          >=110
           =>
       (forall (r:range(P)): pc(s)(r)=70 =>
            i_releaseAccess(s)(r)/=i_migrate(s)(p))
```

```
mi6_b(s): bool=
   forall (p:range(P)):
      (pc(s)(p)>=95 AND pc(s)(p)<=103 OR pc(s)(p)
         >=110) AND index(s)(p)=currInd(s)
            =>
         (forall (r:range(P)): pc(s)(r)=70 =>
                         i_releaseAccess(s)(r)/=
                            i_migrate(s)(p))

mi7_a(s:state4):bool =
  forall (p:range(P)):
         pc(s)(p)>=99=>i_migrate(s)(p)/=next(s)(
            i_migrate(s)(p))

mi7_b(s:state4):bool =
  forall (p:range(P)):
     pc(s)(p)>=95 AND pc(s)(p)<=97 AND index(s)(p)=
        currInd(s)
              => i_migrate(s)(p)/=next(s)(i_migrate(s
                 )(p))

mi8(s:state4):bool =
  forall (p:range(P)):
  (pc(s)(p)>=95 and pc(s)(p)<=97 OR pc(s)(p)>=99 AND
     pc(s)(p)<=103 OR pc(s)(p)>=110)
           AND index(s)(p)=currInd(s) =>next(s)(
              i_migrate(s)(p))=0

mi9_10(s:state4): bool=
   forall (p:range(P)):
       (pc(s)(p)>=95 AND pc(s)(p)<=103 OR pc(s)(p)
          >=110) AND index(s)(p)=currInd(s)
           =>
         (not exists (r:range(P)):
              pc(s)(r)/=0 and (pc(s)(r)<59 or pc(s)(r)
                 >=62 and pc(s)(r)/=65) and
                         H(s)(i_migrate(s)(p))=H(s)(
                            index(s)(r)))
          AND H(s)(i_migrate(s)(p))/=H(s)(currInd(s))

mi11(s:state4):bool =
  forall (p:range(P)):
     pc(s)(p)=101 and index(s)(p)=currInd(s) OR pc(s)(
        p)=102 =>
           h_migrate(s)(p)=H(s)(i_migrate(s)(p))

mi12_a(s:state4):bool =
  forall (p:range(P)):
      pc(s)(p)>=95 AND index(s)(p)=currInd(s)=>
                 i?(Heap(s)(H(s)(i_migrate(s)(p))))

mi12_b(s:state4):bool =
  forall (p:range(P)):
      pc(s)(p)>=102 AND pc(s)(p)<=103 OR pc(s)(p)>=110
          =>
                 i?(Heap(s)(H(s)(i_migrate(s)(p))))

mi13(s:state1):bool =
  forall (p:range(P)):
      pc(s)(p)=103 AND index(s)(p)=currInd(s)=>
          forall (k: below(size(i_(Heap(s)(H(s)(index(
             s)(p))))))):
                table_(i_(Heap(s)(H(s)(index(s)(p)))))
                   (k)=old(null)

mi14_15(s:state6):bool =
forall (p:range(P)):
  pc(s)(p)=103 and index(s)(p)=currInd(s)=>
     (forall (a:Address):
       (forall (n:below(size(i_(Heap(s)(h))))):
          (
            ((forall (m: below(n)):
               (r1/=nor(null) AND (r1=del OR ADR(val(
                  r1))/=a))
                WHERE
               r1=table_(i_(Heap(s)(h)))(KEY(a,size(i_
                  (Heap(s)(h))),m)))
             AND
            (r2=nor(null) OR (r2/=del AND ADR(val(r2))
               =a))
             => X(s)(a)=val(r2))
           AND
```

```
              (X(s)(a)=val(r2) and X(s)(a)/=null=>
               (forall (m: below(n)):
                 (r1/=nor(null) AND (r1=del OR ADR(val(
                    r1))/=a))
                  WHERE
                 r1=table_(i_(Heap(s)(h)))(KEY(a,size(i_
                    (Heap(s)(h))),m)))
               AND
              (r2=nor(null) OR (r2/=del AND ADR(val(r2))
                 =a)))
           )WHERE r2=table_(i_(Heap(s)(h)))(KEY(a,size(
              i_(Heap(s)(h))),n))
         ) WHERE h=H(s)(i_migrate(s)(p)))

mi16(s:state6):bool =
forall (p:range(P)):
  pc(s)(p)=103 and
        index(s)(p)=currInd(s)=>(forall (k:below(size(
           i_(Heap(s)(h))))):
           not(oldp(table_(i_(Heap(s)(h)))(k)))
          ) WHERE h=H(s)(i_migrate(s)(p))

mi17_18(s:state6):bool =
forall (p:range(P)):
  pc(s)(p)=103 and index(s)(p)=currInd(s)=>
  (forall (a:Address):
       (X(s)(a)/=null =>
              FORALL (i:below(l)):
                 X(s)(a)=val(table_(i_(Heap(s)(h)))(
                    KEY(a,l,i)))=>
                      not ((exists (j:below(l)):
                    j/=i AND ADR(val(table_(i_(Heap(s)(
                       h)))(KEY(a,l,j))))=a)))
       AND
       (X(s)(a)=null => not(exists (i:below(l)):
                           val(table_(i_(Heap(s)(h)))
                              (KEY(a,l,i)))/=null AND
                           ADR(val(table_(i_(Heap(s)(
                              h)))(KEY(a,l,i))))=a))
  ) WHERE h=H(s)(i_migrate(s)(p)),l=size(i_(Heap(s)(h)
    ))
```

```
mi19(s:state6):bool =
forall (p:range(P)):
  pc(s)(p)=103 and index(s)(p)=currInd(s)=>
  (forall (a:Address):
       (X(s)(a)/=null =>
         exists (i:below(l)): X(s)(a)=val(table_(i_(
            Heap(s)(h)))(KEY(a,l,i))))
       )where  h=H(s)(i_migrate(s)(p)),l=size(i_(Heap(
          s)(h)))

mi20(s:state13):bool =
  forall (p:range(P)):
      (pc(s)(p)=117 and X(s)(a)/=null and
       val(table_(i_(Heap(s)(H(s)(index(s)(p)))))(
          i_moveContents(s)(p)))/=null
      or pc(s)(p)>=126 and X(s)(a)/=null and index(s)(
         p)=currInd(s)
      or pc(s)(p)=125 and X(s)(a)/=null and index(s)(p
         )=currInd(s)
         and (b_moveElement(s)(p) OR (val(
            w_moveElement(s)(p)) /= null AND
             a_moveElement(s)(p) = ADR(val(
                w_moveElement(s)(p)))))
      =>
      ((exists (i:below(l)): X(s)(a)=val(table_(i_(
         Heap(s)(h)))(KEY(a,l,i))))
       WHERE h=H(s)(next(s)(currInd(s))), l=size(i_(
          Heap(s)(h))))
     )where a=ADR(val(Y(s)(i_moveContents(s)(p))))

%invariants about moveContents

mc1(s:state4):bool =
  forall (p:range(P)):
     pc(s)(p)=103 OR pc(s)(p)>=110=> to_moveContents(s
        )(p)=H(s)(i_migrate(s)(p))

mC2(s:state):bool =
  forall (p:range(P)):
      pc(s)(p)>=110 => H(s)(index(s)(p))=
         from_moveContents(s)(p)

mC3(s:state1) : bool=
  forall (p:range(P),x:nat):
    pc(s)(p) > 102 and member(x,
       toBeMoved_moveContents(s)(p))=>
           x < size(i_(Heap(s)(H(s)(index(s)(p)))))

mC4(s:state16) : bool=
  forall (p:range(P)):
       pc(s)(p) = 111 =>
       (EXISTS (l: below(size(i_(Heap(s)(
          from_moveContents(s)(p)))))):
          member(l, toBeMoved_moveContents(s)(p)))

mC5(s:state):bool =
  forall (p:range(P)):
      pc(s)(p)>=114 AND pc(s)(p)/=118 =>
         v_moveContents(s)(p)/=old(null)

mC6(s:state1):bool =
  forall (p:range(P)):
         pc(s)(p)>=114 =>
             i_moveContents(s)(p)<size(i_(Heap(s)(H(s)
                (index(s)(p)))))

mC7(s:state7):bool =
  forall (p:range(P)):
      pc(s)(p)=118 =>
        table_(i_(Heap(s)(H(s)(index(s)(p)))))(
           i_moveContents(s)(p))=old(null)

mC8(s:state1):bool =
  forall (p:range(P)):
      pc(s)(p)>=110 =>
          forall (k: below(size(i_(Heap(s)(H(s)(index(
             s)(p))))))):
             (not(member(k, toBeMoved_moveContents(s)(
                p)))=>
                       table_(i_(Heap(s)(H(s)(index(s
                          )(p)))))(k)=old(null))
```

```
mC9(s:state1):bool =
  forall (p:range(P)):
      pc(s)(p)>=110 and index(s)(p)=currInd(s) AND
             toBeMoved_moveContents(s)(p) = emptyset=>
               forall (k: below(size(i_(Heap(s)(H(s)(
                  index(s)(p))))))):
                  table_(i_(Heap(s)(H(s)(index(s)(p)))
                     ))(k)=old(null)

mC10(s:state8):bool =
  forall (p:range(P)):
     pc(s)(p)>=116 and val(v_moveContents(s)(p))/=null
        and
        table_(i_(Heap(s)(H(s)(index(s)(p)))))(
           i_moveContents(s)(p))=old(null)
         => (table_(h)(KEY(a,l,0))/=nor(null)
            )WHERE h=i_(Heap(s)(H(s)(i_migrate(s)(p)))
              ),
                  a=ADR(val(v_moveContents(s)(p))), l
                    =size(h)

mC11(s:state7):bool =
  forall (p:range(P)):
    pc(s)(p)>=116 and ((k1/=old(null))
      WHERE k1=table_(i_(Heap(s)(H(s)(index(s)(p)))))(
         i_moveContents(s)(p)))
       =>
    ((val(v_moveContents(s)(p))=val(k1) AND oldp(k1))
      WHERE k1=table_(i_(Heap(s)(H(s)(index(s)(p)))))(
         i_moveContents(s)(p)))

mC12(s:state):bool =
  forall (p:range(P)):
    pc(s)(p)>=116 and index(s)(p)=currInd(s) and  val(
       v_moveContents(s)(p))/=null =>
         val(v_moveContents(s)(p))=val(Y(s)(
            i_moveContents(s)(p)))

%invariants about moveElement
```

```
mE1(s:state):bool =
  forall (p:range(P)):
     pc(s)(p)>=120 => val(v_moveContents(s)(p))=
        v_moveElement(s)(p)

mE2(s:state):bool =
  forall (p:range(P)):
      pc(s)(p)>=120 => v_moveElement(s)(p)/=null

mE3(s:state4):bool =
  forall (p:range(P)):
      pc(s)(p)>=120  => to_moveElement(s)(p)=H(s)(
         i_migrate(s)(p))

mE4(s:state):bool =
  forall (p:range(P)):
      pc(s)(p)>=121 => a_moveElement(s)(p)=ADR(val(
         v_moveContents(s)(p)))

mE5(s:state9) : bool=
  forall (p:range(P)):
      pc(s)(p)>=121 =>
           m_moveElement(s)(p)=size(i_(Heap(s)(
              to_moveElement(s)(p))))

mE6(s:state):bool =
  forall (p:range(P)):
      pc(s)(p)=121 or pc(s)(p)=123 => not(
         b_moveElement(s)(p))

mE7(s:state9):bool =
  forall (p:range(P)):
       pc(s)(p)=123 =>(k_moveElement(s)(p)=KEY(a,l,n)
       )WHERE h=i_(Heap(s)(to_moveElement(s)(p))),
              a=a_moveElement(s)(p),l=size(h),n=
                 n_moveElement(s)(p)

mE8(s:state6):bool =
  forall (p:range(P)):
            pc(s)(p)>=123 =>
               k_moveElement(s)(p)<size(i_(Heap(s)(H(s
                )(i_migrate(s)(p)))))

mE9(s:state9):bool =
  forall (p:range(P)):
       pc(s)(p)=120 and
           ((table_(h)(KEY(ADR(v_moveElement(s)(p)),
              size(h),0))=nor(null))
            WHERE h=i_(Heap(s)(to_moveElement(s)(p))))
           => index(s)(p)=currInd(s)

mE10(s:state9):bool =
  forall (p:range(P)):
       (pc(s)(p)=121 or pc(s)(p)=123) and ((table_(h)(
          KEY(a,l,n))=nor(null)
              )WHERE h=i_(Heap(s)(to_moveElement(s)(p)
                 )),
                     a=a_moveElement(s)(p),l=size(h),n
                        =n_moveElement(s)(p))
           => index(s)(p)=currInd(s)

mE11(s:state9):bool =
  forall (p:range(P)):
     (pc(s)(p)=121 OR pc(s)(p)=123) and ((table_(h)(
        KEY(a,l,n))=nor(null)
            )WHERE h=i_(Heap(s)(to_moveElement(s)(p)))
              ,
                   a=a_moveElement(s)(p),l=size(h),n=
                      n_moveElement(s)(p))
       => not(exists (r:range(P)): pc(s)(r)=103 AND
          index(s)(r)=currInd(s))

mE12(s:state3):bool =
  forall (p:range(P)):
     (pc(s)(p)=121 or pc(s)(p)=123) and next(s)(
        currInd(s))/=0 and
        to_moveElement(s)(p)=H(s)(next(s)(currInd(s)))
           =>
           n_moveElement(s)(p)<size(i_(Heap(s)(H(s)(
              next(s)(currInd(s))))))

mE13(s:state9):bool =
  forall (p:range(P)):
        (pc(s)(p)=123 or pc(s)(p)=125) and
           w_moveElement(s)(p)/=nor(null)
           =>
        (ADR(val(w_moveElement(s)(p)))= ADR(val(table_
           (h)(k_moveElement(s)(p))))
         or table_(h)(k_moveElement(s)(p))=del
         or table_(h)(k_moveElement(s)(p))=old(null)
        )WHERE h=i_(Heap(s)(to_moveElement(s)(p)))

mE14(s:state9):bool =
  forall (p:range(P)):
      pc(s)(p)>=123 and w_moveElement(s)(p)/=nor(null)
           =>(table_(h)(k_moveElement(s)(p))/=nor(null
              )
             )WHERE h=i_(Heap(s)(H(s)(i_migrate(s)(p))
                ))

mE15(s:state6):bool =
  forall (p:range(P)):
      pc(s)(p)=117 and val(v_moveContents(s)(p))/=null
          OR
        (pc(s)(p)=121 or pc(s)(p)=123) and
           n_moveElement(s)(p)>0 OR pc(s)(p)>=125
           =>(table_(h)(KEY(a,l,0))/=nor(null)
             )WHERE h=i_(Heap(s)(H(s)(i_migrate(s)(p))
                )),
                    a=ADR(val(v_moveContents(s)(p))),
                       l=size(h)

mE16(s:state9):bool =
  forall (p:range(P)):
      pc(s)(p)=121 OR pc(s)(p)=123 OR (pc(s)(p)=125
         and not (b_moveElement(s)(p) OR
                    (val(w_moveElement(s)(p)) /= null
                       AND
                             a_moveElement(s)(p) = ADR
                                (val(w_moveElement(s)(
                               p))))))
             =>
             forall (m:below(n_moveElement(s)(p))):
                  (r1/=nor(null) AND (r1=del OR r1=old(
                     null) OR ADR(val(r1))/=a)
                  )WHERE l=size(i_(Heap(s)(
                      to_moveElement(s)(p)))),
                      a=a_moveElement(s)(p),
                      r1=table_(i_(Heap(s)(
                         to_moveElement(s)(p))))(KEY(a,
                         l,m))

%invariants concerning prot

pr1(s:state): bool =
   forall (i:range(2*P)): G_protsize(s,i)=prot(s)(i)

pr2_3(s:state):bool=
  (forall(p:range(P)):
      pc(s)(p)/=0 and (pc(s)(p)<59 or pc(s)(p)>=62 and
          pc(s)(p)/=65) =>
           prot(s)(index(s)(p))>0)
  AND prot(s)(currInd(s))>0

pr4(s:state):bool=
  next(s)(currInd(s))/=0=>prot(s)(next(s)(currInd(s)))
     >0

pr5(s:state) : bool=
  forall (m:range(2*P)):
        prot(s)(m)=0 => Heap(s)(H(s)(m))=bot

pr6(s:state) : bool=
  forall (m:range(2*P)):
        prot(s)(m)-size(prSet3(s,m))<=0 and busy(s)(m)
           =0
          => Heap(s)(H(s)(m))=bot

pr7(s:state):bool=
  forall(p:range(P)):  pc(s)(p)>=67 and pc(s)(p)<=72
     =>
        prot(s)(i_releaseAccess(s)(p))>0

pr8(s:state):bool=
  forall(p:range(P)):  pc(s)(p)>=81 AND pc(s)(p)<=84=>
        prot(s)(i_newTable(s)(p))>0

pr9(s:state4):bool=
  forall(p:range(P)):  pc(s)(p)>=97=> prot(s)(
     i_migrate(s)(p))>0

pr10(s:state) : bool=
  forall (p:range(P)):
        pc(s)(p)>=81 and pc(s)(p)<=82=>
            prot(s)(i_newTable(s)(p))-size(prSet4(s,
               i_newTable(s)(p)))=1

%invariants concerning busy

bu1(s:state): bool =
   forall (i:range(2*P)): G_busysize(s,i)=busy(s)(i)

bu2_3(s:state):bool=
  (forall(p:range(P)):  pc(s)(p)/=0 and (pc(s)(p)<59
     or (pc(s)(p)>65
                       and not (i_releaseAccess(s)(p)=
                          index(s)(p) and (pc(s)(p)
                          >=67
                                and pc(s)(p)<=72))))
                       =>busy(s)(index(s)(p))>0)
  AND busy(s)(currInd(s))>0

bu4(s:state):bool=
  next(s)(currInd(s))/=0=>busy(s)(next(s)(currInd(s)))
     >0

bu5(s:state):bool=
  forall(p:range(P)):  pc(s)(p)=81 => busy(s)(
     i_newTable(s)(p))=0

bu6(s:state4):bool=
  forall(p:range(P)):  pc(s)(p)>=100=> busy(s)(
     i_migrate(s)(p))>0

%other invariants
```

```
Ot1(s) : bool =
      X(s)(nullAddress)=null

Ot2(s) : bool =
   forall (a : Address):
       X(s)(a)/=null => ADR(X(s)(a))=a

Ot3(s) : bool = forall (p : range(P)):
           (return_getAccess(s)(p)=10 OR
            return_getAccess(s)(p)=20 OR
            return_getAccess(s)(p)=30 OR
            return_getAccess(s)(p)=36 OR
            return_getAccess(s)(p)=46 OR
            return_getAccess(s)(p)=51 OR
            return_getAccess(s)(p)=1) AND
           (
            return_releaseAccess(s)(p)=59 OR
            return_releaseAccess(s)(p)=77 OR
            return_releaseAccess(s)(p)=90 OR
            return_releaseAccess(s)(p)=0) AND
           (
            return_refresh(s)(p)=10 OR
            return_refresh(s)(p)=20 OR
            return_refresh(s)(p)=30 OR
            return_refresh(s)(p)=36 OR
            return_refresh(s)(p)=46 OR
            return_refresh(s)(p)=51) AND
           (
            return_newTable(s)(p)=30 OR
            return_newTable(s)(p)=46)

Ot4(u:state):bool=
  forall (q:range(P)):
    pc(u)(q)=0 OR pc(u)(q)=1 OR pc(u)(q)=5 OR pc(u)(q)
       =6 OR pc(u)(q)=7 OR
    pc(u)(q)=8 OR pc(u)(q)=10 OR pc(u)(q)=11 OR pc(u)(
       q)=13 OR pc(u)(q)=14 OR
    pc(u)(q)=15 OR pc(u)(q)=16 OR pc(u)(q)=17 OR pc(u)
       (q)=18 OR pc(u)(q)=20 OR
    pc(u)(q)=21 OR pc(u)(q)=25 OR pc(u)(q)=26 OR pc(u)
```

```
      (q)=27 OR pc(u)(q)=28 OR
    pc(u)(q)=30 OR pc(u)(q)=31 OR pc(u)(q)=32 OR pc(u)
      (q)=33 OR pc(u)(q)=35 OR
    pc(u)(q)=36 OR pc(u)(q)=37 OR pc(u)(q)=41 OR pc(u)
      (q)=42 OR
    pc(u)(q)=43 OR pc(u)(q)=44 OR pc(u)(q)=46 OR pc(u)
      (q)=47 OR pc(u)(q)=48 OR
    pc(u)(q)=49 OR pc(u)(q)=50 OR pc(u)(q)=51 OR pc(u)
      (q)=52 OR
    pc(u)(q)=57 OR pc(u)(q)=59 OR pc(u)(q)=60 OR pc(u)
      (q)=61 OR pc(u)(q)=62 OR
    pc(u)(q)=63 OR pc(u)(q)=65 OR pc(u)(q)=67 OR pc(u)
      (q)=68 OR
    pc(u)(q)=69 OR pc(u)(q)=70 OR pc(u)(q)=72 OR pc(u)
      (q)=77 OR
    pc(u)(q)=78 OR pc(u)(q)=81 OR pc(u)(q)=82 OR pc(u)
      (q)=83 OR pc(u)(q)=84 OR
    pc(u)(q)=90 OR pc(u)(q)=94 OR pc(u)(q)=95 OR pc(u)
      (q)=97 OR pc(u)(q)=98 OR
    pc(u)(q)=99 OR pc(u)(q)=100 OR pc(u)(q)=101 OR pc(
      u)(q)=102 OR pc(u)(q)=103 OR
    pc(u)(q)=104 OR pc(u)(q)=105 OR pc(u)(q)=110 OR pc
      (u)(q)=111 OR pc(u)(q)=114 OR
    pc(u)(q)=116 OR pc(u)(q)=117 OR pc(u)(q)=118 OR pc
      (u)(q)=120 OR pc(u)(q)=121 OR
    pc(u)(q)=123 OR pc(u)(q)=125 OR pc(u)(q)=126 OR pc
      (u)(q)=71

%invariants for type_checking assumptions

G_FT(s:state):bool=
    He3_4(s) and He6(s) and Cu8(s) and mi4(s)
         and nT2(s) and mi12_a(s) and mi12_b(s) and
            mC6(s)
         and mE8(s) and mE3(s) and de13_(s) and fi4(s)
         and de3(s) and in3(s) and as3(s) and fi3(s)
         and de2(s) and in2(s) and as2(s) and mi5(s)
         and Cu1(s) and Ha3(s)

% Definitions for trivial invariants
```

```
G_T0(s) : bool =true
G_T1(s) : bool =true
G_T2(s) : bool =true

G_try1 :THEOREM % proven, trivial theorem
   forall (v:EValue): norp(v)=> not oldp(v)

G_try2 :THEOREM % proven, need axiom oldEValue and
   norEValue
   forall (v:EValue): val(v)=null => v=old(null) OR v=
      del OR v=nor(null)

% proofs for the stabilities

IV_Co1: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND Co1(u) AND fi10(u) AND Ot3(u)
           AND fi1_(u)
           => Co1(v)

IV_Co2: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND Co2(u) AND G_FT(u) AND de5(u) AND
         Ot3(u)
         and de6(u) and fi1_(u) and de11(u)
      => Co2(v)

IV_Co3: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND Co3(u) AND G_FT(u) AND in5(u) AND
         Ot3(u)
         and in6(u) and fi1_(u) and in11(u)
      => Co3(v)

IV_Cn1: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v)  AND Cn1(u) AND Cn6(u) AND Ot3(u)
          => Cn1(v)

IV_Cn2: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v)  AND Cn2(u) AND Cn8(u) AND Ot3(u)
           AND fi1_(u)
          => Cn2(v)

IV_Cn3: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v)  AND Cn3(u) AND Cn10(u) AND Ot3(u)
            AND fi1_(u)
            AND in5(u)
          => Cn3(v)

IV_Cn4: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v)  AND Cn4(u) AND Cn11(u) AND Ot3(u)
          => Cn4(v)

IV_No1 : THEOREM %%% proven
  forall (u: state) :
     No2(u)=> No1(u)

IV_No2 : THEOREM %%% proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND No2(u) AND nT1(u) AND He2_a(u)
        AND rA2(u)
          AND Ot3(u) AND Ha2(u) AND Ha1(u) AND rA1(u)
          AND rA14(u) AND rA3(u) AND nT14(u)
          AND rA4(u)
      => No2(v)

IV_He1: THEOREM %proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) and He1(u) and Ha1(u)
          => He1(v)

IV_He2_a: THEOREM %proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND He2_a(u) AND Ha3(u) AND rA5(u)
          => He2_a(v)

IV_He2_b: THEOREM %proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) and He2_b(u) and Ha1(u) and He1(u)
            AND rA2(u)
          => He2_b(v)

IV_He3_4: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v)  AND He3_4(u) AND Ot3(u) AND rA6_7
           (u)
             and mi4(u) and mi12_b(u) and  rA11(u)
             and rA5(u)
        => He3_4(v)

IV_He5: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v)  AND He5(u) AND He1(u)
        => He5(v)

IV_He6: THEOREM %$$proven
   forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND He6(u) AND G_FT(u) AND rA8(u)
         AND Ha3(u) AND mi8(u) AND nT2(u) AND rA5(u)
      => He6(v)

IV_Ha1: THEOREM % proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND Ha1(u) => Ha1(v)

IV_Ha2: THEOREM % proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND Ha2(u) and Ha1(u)=> Ha2(v)

IV_Ha3: THEOREM %proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND Ha3(u) AND Ha2(u) AND Ha1(u)
           AND He2_a(u)
           AND He1(u)
          => Ha3(v)

IV_Ha4: THEOREM %$$proven
   forall (u: state) :
       Ha3(u) and He3_4(u)=> Ha4(u)
```

```
IV_Cn5: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND Cn5(u) AND Cn6(u) AND Ot3(u)
      => Cn5(v)

IV_Cn6: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND Cn6(u) AND Cn5(u) AND Ot3(u)
      => Cn6(v)

IV_Cn7: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND Cn7(u) AND Cn8(u) AND Ot3(u) AND
        fi1_(u)
      => Cn7(v)

IV_Cn8: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND Cn8(u) and Cn7(u) AND Ot3(u)
      => Cn8(v)

IV_Cn9: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND Cn9(u) and Cn10(u) AND Ot3(u) AND
         fi1_(u)
        AND in5(u)
      => Cn9(v)

IV_Cn10: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND Cn10(u) AND Cn9(u) AND Ot3(u) AND
         in5(u)
      => Cn10(v)

IV_Cn11: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND Cn11(u) AND Ot3(u)
      => Cn11(v)

IV_Cu1: THEOREM %$$$ proven
```

```
  forall (u,v : state1, q : range(P) ) :
        step(q,u,v)  AND Cu1(u) AND G_FT(u) AND Ot3(u)
            AND Ha4(u)
             and rA6_7(u) and nT12_13(u) and Ha2(u)
             and He3_4(u) and mi9_10(u) and  rA11(u)
             and nT9_10(u) and mi13(u) and rA5(u)
        => Cu1(v)

IV_Cu2: THEOREM %$$$ proven
   forall (u: state1) :
     Cu6(u) AND Cu7(u) AND Cu3(u) and He3_4(u)
     => Cu2(u)

IV_Cu3: THEOREM %$$$ proven
   forall (u,v : state1, q : range(P) ) :
     step(q,u,v) AND Cu3(u) AND G_FT(u) AND rA6_7(u)
        AND G_T0(u)
         AND nT12_13(u) AND mi5(u) AND  mi4(u)
         AND Ne8(u) and rA5(u)
     => Cu3(v)

IV_Cu4: THEOREM %$$$ proven
   forall (u,v : state1, q : range(P) ) :
     step(q,u,v) AND Cu4(u) AND G_FT(u) AND fi1_(u)
        AND rA6_7(u)
         AND Ha2(u) AND nT12_13(u) AND Ne9(u) and
            Cu9_10(u)
         AND de7_(u) AND He3_4(u) AND mi5(u) AND mi4(u
            )
         AND Ot3(u) and Ha4(u) and de3(u) and G_T0(u)
         AND G_T1(u) and mi9_10(u) and de5(u) and rA5(
            u)
     => Cu4(v)

IV_Cu6: THEOREM %$$$ proven
   forall (u,v : state1, q : range(P) ) :
     step(q,u,v) AND Cu6(u) AND G_FT(u) AND Ot3(u) AND
         rA6_7(u)
         AND Ha2(u) AND nT12_13(u) AND Ha3(u)  and
            He3_4(u)
         AND in3(u) AND as3(u) AND Ne23(u) and mi5(u)
         AND mE6(u) and mE7(u) and mE10(u)
         AND mE3(u) and Ne3(u) and mi12_b(u)
         AND mi1_a(u) and mi4(u) and rA5(u)
     => Cu6(v)

IV_Cu7: THEOREM %$$$ proven
   forall (u,v : state1, q : range(P) ) :
     step(q,u,v) AND Cu7(u) AND G_FT(u) AND Ot3(u) AND
         rA6_7(u)
         AND Ha2(u) AND nT12_13(u) AND Ha3(u)  and
            He3_4(u)
         AND in3(u) AND as3(u) AND in5(u) and mi5(u)
         AND mE6(u) and mE7(u) and mE10(u)
         AND mE3(u) and Ne3(u) and mi12_b(u)
         AND mi1_a(u) and mi4(u) and de7_(u) and Ne22(
            u)
         AND mi9_10(u) and rA5(u) and Cu9_10(u)
         and fi1_(u)
     => Cu7(v)

IV_Cu8: THEOREM %$$ proven
   forall (u,v : state1, q : range(P) ) :
      step(q,u,v) AND Cu8(u) AND G_FT(u) AND Ha2(u)
         AND nT9_10(u)
            AND rA6_7(u) AND mi5(u) AND mi4(u)
            AND G_T0(u) AND mC2(u) AND mC5(u)
            AND He3_4(u) AND Cu1(u) AND Ha4(u) AND mC6
               (u)
            AND mi16(u) and rA5(u)
      => Cu8(v)

IV_Cu9_10: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
      step(q,u,v) and Cu9_10(u) and G_FT(u) AND rA6_7(
         u)
         and nT12_13(u) and Ha2(u) and He3_4(u) and
            Cu1(u) and Ha4(u)
              AND de3(u) AND in3(u) AND as3(u)
              AND mE3(u) AND mi9_10(u) AND mE10(u)
              AND mE7(u) and rA5(u)
        => Cu9_10(v)
```

```
IV_Cu11_12: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
        step(q,u,v) and Cu11_12(u) and G_FT(u) AND
           Cu9_10(u) and Cu13_14(u)
             AND fi1_(u) AND rA6_7(u) AND Ha2(u) AND
                nT12_13(u)
             AND He3_4(u) AND Cu1(u) AND Ha4(u)
             AND in3(u) AND as3(u) and mi14_15(u)
             AND de3(u) AND G_T0(u) AND in10(u) AND
                as8(u)
             AND mi12_b(u) AND Ot2(u) AND fi5_(u) AND
                Cu15(u)
             AND de11(u) and in11(u) and G_T1(u) and
                rA5(u)
        => Cu11_12(v)

IV_Cu13_14: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
    step(q,u,v)  AND Cu13_14(u) AND G_FT(u) AND He3_4(
       u) and Ot2(u) and fi1_(u)
       and Ot1(u) and rA6_7(u) and nT12_13(u) and Ha2(
          u)
       and Cu9_10(u) AND Cu1(u) AND Ha4(u) AND de3(u)
          AND in3(u)
       AND as3(u) AND Cu11_12(u) and  G_T0(u) AND in10
          (u)
       AND as8(u) AND fi5_(u) and Cu15(u) and mi17_18(
          u)
       and mi12_b(u) and mi4(u) and de11(u) and rA5(u)
     => Cu13_14(v)

IV_Cu15: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
    step(q,u,v)  AND Cu15(u) AND G_FT(u) AND He3_4(u)
       and rA6_7(u)
        AND nT12_13(u) and Ha2(u) AND Cu1(u) AND Ha4(u
           ) and fi1_(u)
        AND de3(u) AND in3(u) AND as3(u) AND fi5_(u)
        AND mi12_b(u) and mi19(u) and mi4(u)
        AND Ot2(u) and Cu9_10(u) and Cu11_12(u) and
```

```
            Cu13_14(u) and rA5(u)
      => Cu15(v)

IV_Cu16: THEOREM %$$$ proven
   forall (u: state1) :
        Cu13_14(u) and Cu15(u) and He3_4(u) and Ot1(u)
          => Cu16(u)

IV_Ne1: THEOREM %$$proven
   forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND Ne1(u) AND G_FT(u) AND nT9_10(u)
          AND mi7_a(u)
            => Ne1(v)

IV_Ne2: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND Ne2(u) AND G_FT(u) AND Ne5(u)
           AND nT3(u) AND mi8(u) AND nT9_10(u)
      => Ne2(v)

IV_Ne3: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND Ne3(u) AND G_FT(u) AND Ne1(u)
          AND nT9_10(u) AND mi8(u)
      => Ne3(v)

IV_Ne4: THEOREM %$$proven
   forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND Ne4(u) AND Ne1(u) AND nT9_10(u)
            => Ne4(v)

IV_Ne5: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND Ne5(u) AND Ot3(u) AND nT9_10(u)
         AND
           mi5(u)
        => Ne5(v)

IV_Ne6: THEOREM %$$$ proven
   forall (u: state) :
      G_FT(u) and G_T0(u) and G_T1(u) and Ne10(u) and
```

```
          Ne24(u) and He6(u)
         and He3_4(u) and Cu4(u)
       =>Ne6(u)

IV_Ne7: THEOREM %$$$ proven
   forall (u,v : state3, q : range(P) ) :
     step(q,u,v) AND Ne7(u) AND G_FT(u) AND Ha3(u) AND
          rA6_7(u)
          AND rA8(u) and nT12_13(u) AND nT11(u)
          AND He3_4(u) and mi8(u) and nT7(u)
          AND Ne5(u) and Ha2(u) and He6(u) and rA5(u)
     => Ne7(v)

IV_Ne8: THEOREM %$$$ proven
   forall (u,v : state3, q : range(P) ) :
     step(q,u,v) AND Ne8(u) AND G_FT(u) AND Ha3(u) and
          rA8(u)
          AND nT11(u) AND G_TO(u) AND mi8(u)
          AND nT6(u) and Ne5(u) AND rA5(u)
     => Ne8(v)

IV_Ne9: THEOREM %$$$ proven
   forall (u,v : state2, q : range(P) ) :
     step(q,u,v) AND Ne9(u) AND G_FT(u) AND Ha3(u) AND
          Ha2(u)
           AND Ne3(u) and rA5(u) and Ne5(u)
           AND de3(u) and mi8(u) and as3(u)
           AND rA8(u) and rA6_7(u) and nT8(u)
           AND nT11(u) and  nT4(u) and mC2(u)
     => Ne9(v)

IV_Ne9a: THEOREM %$$$ proven
   forall (u,v : state2, q : range(P) ) :
     step(q,u,v) AND Ne9a(u) AND G_FT(u) and Ha3(u)
        and  nT4(u)
           AND Ne3(u) and rA5(u) and rA8(u)
           AND de3(u) and mi8(u)
     => Ne9a(v)

IV_Ne10: THEOREM %$$$ proven
  forall (u,v : state2, q : range(P) ) :
       step(q,u,v) AND Ne10(u) AND G_FT(u) AND Ha3(u)
          AND Ha2(u)
           AND de3(u) and G_T0(u) and rA8(u)
           AND nT11(u) and G_T1(u) and Ne3(u)
           AND He6(u) and mi8(u) and nT8(u)
           AND mC2(u) and Ne5(u) and rA5(u)
           AND nT2(u)
         => Ne10(v)

IV_Ne11: THEOREM %$$$ proven
  forall (u,v : state3, q : range(P) ) :
       step(q,u,v) AND Ne11(u) AND G_FT(u) AND Ha3(u)
          AND Ha2(u)
           AND rA5(u) and He6(u)
           AND mC2(u) and Ne3(u) and nT2(u) and nT8(u)
           AND rA8(u) and mi8(u)
           AND nT11(u) and  nT12_13(u)
     => Ne11(v)

IV_Ne12_13: THEOREM %$$$ proven
  forall (u,v : state15, q : range(P) ) :
       step(q,u,v) AND Ne12_13(u) AND G_FT(u) AND Ha3(u
          ) AND Ha2(u)
           AND Cu8(u) and He6(u)
           AND He3_4(u) and Cu1(u) and G_T0(u) and G_T1
              (u)
           AND de3(u) and in3(u) and as3(u)
           AND rA8(u) and rA6_7(u)
           AND nT11(u) and  nT12_13(u)
           AND mi12_b(u) and mi16(u) and mi5(u)
           and mi4(u) AND G_T2(u)
           AND de7_(u) and Ot2(u) and fi1_(u)
           AND Cu9_10(u) and Cu13_14(u) and Cu15(u) and
               as9(u)
           and fi5_(u) and mC2(u) and Ne3(u)
           AND Ot1(u) and Ne14(u) and Ne20(u)
           and mE16(u)and mE7(u) and mE4(u)
           and mE1(u) and  mE12(u) and rA5(u)
           AND mE2(u) and Ne15_16(u) and Ne17_18(u)
           AND mi20(u) and de11(u) and in11(u)
         => Ne12_13(v)
```

```
IV_Ne14: THEOREM %$$$proven
  forall (u,v : state3, q : range(P) ) :
      step(q,u,v) AND Ne14(u) AND G_FT(u) AND Ha3(u)
         AND Ha2(u)
          AND mE7(u) and He6(u)
          AND He3_4(u) and G_T0(u) and nT2(u) and nT8(
             u)
          AND de3(u) and in3(u) and as3(u)
          AND rA8(u) AND mC2(u)
          AND nT11(u) and mE16(u) and mE1(u)
          AND mE4(u) and mE12(u) and Ne17_18(u)
          AND Cu1(u) and rA5(u)
          and Ot2(u) and fi1_(u)
          AND Cu9_10(u) and mi8(u)  and Ne3(u)
        => Ne14(v)

IV_Ne15_16: THEOREM %$$$ proven
  forall (u,v : state15, q : range(P) ) :
      step(q,u,v) AND Ne15_16(u) AND G_FT(u) AND Ha3(u
         ) AND Ha2(u)
          AND Cu8(u) and He6(u)
          AND He3_4(u) and Cu1(u) and G_T0(u) and G_T1
             (u)
          AND de3(u) and in3(u) and as3(u)
          AND rA8(u) and rA6_7(u)
          AND nT11(u) and  nT12_13(u)
          AND mi12_b(u) and mi16(u) and mi5(u)
          and mi4(u) AND G_T2(u)
          AND de7_(u) and Ot2(u) and fi1_(u)
          AND Cu9_10(u) and Cu13_14(u) and Cu15(u) and
              as9(u)
          and fi5_(u) and mC2(u) and Ne3(u)
          AND Ot1(u) and Ne19(u) and Ne20(u) and
             Ne12_13(u)
          and mE16(u)and mE7(u) and mE4(u)
          and mE1(u) and  mE12(u) and mE10(u)
          AND rA5(u) and in11(u) and de11(u) and mE2(u
             )
        => Ne15_16(v)
```

```
IV_Ne17_18: THEOREM %$$$ proven
  forall (u,v : state3, q : range(P) ) :
      step(q,u,v) AND Ne17_18(u) AND G_FT(u) AND Ha3(u
         ) AND Ha2(u)
          AND mi8(u) and He6(u)
          AND He3_4(u) and Cu1(u) and G_T0(u) and nT2(
             u)
          AND de3(u) and in3(u) and as3(u)
          AND rA8(u) and rA6_7(u)
          AND nT11(u) and  nT12_13(u)
          AND Cu15(u) and Cu13_14(u) and Cu11_12(u)
          and as8(u) and de11(u)
          AND de7_(u) and Ot2(u) and fi1_(u)
          AND Cu9_10(u) and G_T1(u) and nT8(u)
          and mE2(u) and fi5_(u) and mC2(u)
          and Ne3(u) AND G_T2(u) and mC11(u)
          and mC6(u) and mC12(u) and rA5(u)
          and mE7(u) and mE10(u) and mE1(u)
          and Cu8(u)
        => Ne17_18(v)

IV_Ne19: THEOREM %$$$ proven
  forall (u,v : state3, q : range(P) ) :
      step(q,u,v) AND Ne19(u) AND G_FT(u) AND Ha3(u)
         AND Ha2(u)
          AND mi8(u) and He6(u)
          AND Ne3(u) and G_T0(u) and nT2(u) and nT8(u)
          AND de3(u) and in3(u) and as3(u) AND rA8(u)
          and mE7(u) AND nT11(u) and Ne14(u) AND mE16(
             u)
          and Ot1(u) and mE1(u) and mE4(u) and mE12(u)
          and Ne17_18(u) and rA5(u)
        => Ne19(v)

IV_Ne20: THEOREM %$$$ proven
  forall (u,v : state15, q : range(P) ) :
      step(q,u,v) AND Ne20(u) AND G_FT(u) AND Ha3(u)
         AND Ha2(u)
          AND Cu8(u) and He6(u)
          AND He3_4(u) and Cu1(u) and Ha4(u) and G_T0(
             u)
          AND de3(u) and in3(u) and as3(u)
          AND rA8(u) and rA6_7(u)
          AND nT11(u) and  nT12_13(u)
          AND mi12_b(u) and mi16(u) and mi5(u)
          and mi4(u) AND Ne1(u)
          AND de7_(u) and G_T1(u) and fi1_(u)
          AND Cu9_10(u) and Cu13_14(u) and Cu15(u) and
              as9(u)
          and fi5_(u) and mC2(u) and Ne3(u)
          AND Ot1(u) and mi20(u) and in11(u) and rA5(u
            )
        => Ne20(v)

IV_Ne22: THEOREM %$$$ proven
   forall (u,v : state2, q : range(P) ) :
     step(q,u,v) AND Ne22(u) AND G_FT(u) AND Ot3(u)
        AND rA8(u)
         AND Ha2(u) AND nT11(u) AND Ha3(u) and de3(u)
         AND in3(u) AND as3(u) and mi5(u)
         AND mi4(u) AND Ne3(u) and nT18(u)
         AND mE3(u) and mi8(u) and mE10(u)
         AND mE7(u) and mE6(u) and Ne5(u)
         AND nT5(u) and nT2(u) and rA5(u) and nT8(u)
            and nT12_13(u)
         and mC2(u) AND mE2(u)
     => Ne22(v)

IV_Ne23: THEOREM %$$$ proven
   forall (u: state2) :
     G_FT(u) and Cu6(u) and Cu7(u) AND Ne6(u) AND Ne7(
        u)
       and He3_4(u) and Ne22(u) and He6(u)
     => Ne23(u)

IV_Ne24: THEOREM %$$$ proven
   forall (u: state3) :
      Ne27(u) and He6(u)
          => Ne24(u)

IV_Ne25: THEOREM %$$$ proven
   forall (u: state2) :
       G_FT(u) AND Ne19(u) and G_T0(u) and He6(u)
          => Ne25(u)

IV_Ne26: THEOREM %$$$ proven
   forall (u: state3) :
      Ne17_18(u) and He6(u) => Ne26(u)

IV_Ne27: THEOREM %$$$ proven
   forall (u: state3) :
       G_FT(u) and Cu16(u) and Ne25(u) and Ne26(u) and
           Ne17_18(u)
         and He6(u)
      => Ne27(u)

IV_fi1_: THEOREM % proven
   forall (u,v : state, q : range(P)) :
      step(q,u,v) AND fi1_(u) => fi1_(v)

IV_fi2: THEOREM % proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND fi2(u) AND Ot3(u)
        => fi2(v)

IV_fi3: THEOREM %$$ proven
  forall (u,v : state11, q : range(P) ) :
        step(q,u,v) AND fi3(u) AND fi4(u) AND G_T0(u)
           AND Ot3(u)
                    and rA6_7(u) and Ha2(u) and rA5(u)
        => fi3(v)

IV_fi4: THEOREM %$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND fi4(u) AND G_T0(u) AND Ot3(u)
           AND G_T1(u) and
                    rA6_7(u) and nT12_13(u)
        => fi4(v)

IV_fi5_: THEOREM %$$$ proven
   forall (u: state1) :
      G_T0(u) and Cu2(u) and de10(u) and in10(u) and
        as8(u) and fi8(u) and He3_4(u)
      =>fi5_(u)

IV_fi6: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
    step(q,u,v) AND fi6(u) AND G_FT(u) AND Ot3(u) AND
       fi1_(u)
     AND rA6_7(u) AND Ha2(u) AND nT12_13(u) AND mi9_10
        (u)
     AND Cu9_10(u) AND He3_4(u) AND Cu1(u) AND Ha4(u)
     AND fi4(u) AND in3(u) AND as3(u) and rA5(u)
    => fi6(v)

IV_fi7: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
    step(q,u,v)  AND fi7(u) AND G_FT(u) AND fi8(u) AND
        fi6(u) AND
             fi2(u) AND Ot3(u) AND fi1_(u)
     AND rA6_7(u) AND Ha2(u) AND nT12_13(u) AND mi9_10
        (u)
     AND Cu9_10(u) AND He3_4(u) AND Cu1(u) AND Ha4(u)
     AND fi4(u) AND in3(u) AND as3(u) and rA5(u)
    => fi7(v)

IV_fi8: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
    step(q,u,v) AND fi8(u) AND G_FT(u) AND fi4(u) AND
       fi7(u)
     AND fi2(u) AND Ot3(u) AND fi1_(u)
     AND rA6_7(u) AND Ha2(u) AND nT12_13(u) AND mi9_10
        (u)
     AND Cu9_10(u) AND He3_4(u) AND Cu1(u) AND Ha4(u)
     AND in3(u) AND as3(u) and rA5(u)
    => fi8(v)

IV_fi9: THEOREM %$$$ proven
  forall (u:state12) :
       Cu1(u)  AND Ha4(u) AND Cu9_10(u) AND Cu11_12(u)
           AND fi8(u)
           AND fi3(u) AND fi4(u) and fi5_(u) and He3_4
              (u)
       => fi9(u)
```

```
IV_fi10: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND fi10(u) AND G_FT(u) AND fi9(u)
            AND Ot3(u)
           => fi10(v)

IV_fi11_: THEOREM %$$ proven
  forall (u,v : state, q : range(P) ) :
    step(q,u,v) AND fi11_(u) AND G_FT(u) AND Ot3(u)
       AND nT9_10(u) AND mi9_10(u) AND
          Cu8(u) AND fi4(u) AND de3(u) AND in3(u) AND
             as3(u) AND fi3(u) AND de2(u) AND in2(u)
                AND
             as2(u)
        => fi11_(v)

IV_de2: THEOREM %$$ proven
  forall (u,v : state11, q : range(P) ) :
        step(q,u,v) AND de2(u) AND de3(u) AND G_T0(u)
           AND Ot3(u)
                    and rA6_7(u) and Ha2(u) and rA5(u)
        => de2(v)

IV_de3: THEOREM %$$proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND de3(u) AND G_T0(u) AND Ot3(u)
           AND G_T1(u) and
                    rA6_7(u) and nT12_13(u)
        => de3(v)

IV_de4_: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND Ot3(u) AND de4_(u) =>de4_(v)

IV_de5: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND de5(u) AND Ot3(u)
      => de5(v)

IV_de6: THEOREM %$$$ proven
```

```
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND de6(u) AND G_FT(u) AND Ot3(u) and
         fi1_(u)
        and de11(u)
      => de6(v)

IV_de7_: THEOREM %$$$ proven
  forall (u: state) :
       G_FT(u) AND de3(u) AND in3(u) and as3(u) and
             G_T0(u) AND Cu1(u) AND Ha4(u) and de13_(u
                )
        => de7_(u)

IV_de9: THEOREM %$$$ proven
  forall (u,v : state11, q : range(P) ) :
    step(q,u,v)  AND de9(u) AND G_FT(u) AND Ot3(u) AND
        fi1_(u)
     AND rA6_7(u) AND Ha2(u) AND nT12_13(u) AND mi9_10
        (u)
     AND Cu9_10(u) AND G_T0(u) AND de3(u) AND G_T1(u)
        AND G_T2(u)
     AND de7_(u) and rA5(u)
    => de9(v)

IV_de10: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
    step(q,u,v) AND de10(u) AND G_FT(u) AND de3(u) AND
        de9(u) AND Ot3(u)
     AND fi1_(u) AND rA6_7(u) AND Ha2(u) AND nT12_13(u
        )
     AND mi9_10(u) and Cu9_10(u) and de7_(u) and He3_4
        (u) and rA5(u)
    => de10(v)

IV_de11: THEOREM %$$$ proven
  forall (u:state12) :
       de10(u) and de2(u) and de3(u) and He3_4(u) and
           Cu1(u)
            AND Ha4(u) AND Cu9_10(u) AND Cu11_12(u)
               and fi5_(u)
       => de11(u)
```

```
IV_de13_: THEOREM %$$ proven
  forall (u,v : state1, q : range(P) ) :
      step(q,u,v) AND de13_(u) AND G_FT(u) AND He3_4(u
         ) AND Ot3(u) AND Ha2(u) and
        rA6_7(u) and nT12_13(u) and de2(u) and
        in2(u) and as2(u) and de3(u) and in3(u) and
        as3(u) and rA5(u)
       => de13_(v)

IV_in2: THEOREM % proven
  forall (u,v : state11, q : range(P) ) :
        step(q,u,v) AND in2(u) AND in3(u) AND G_T0(u)
           AND Ot3(u)
                    and rA6_7(u) and Ha2(u) and rA5(u)
        => in2(v)

IV_in3: THEOREM % proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND in3(u) AND G_T0(u) AND Ot3(u)
           AND G_T1(u) and
                    rA6_7(u) and nT12_13(u)
        => in3(v)

IV_in5: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND in5(u) AND Ot3(u)
      => in5(v)

IV_in6: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND in6(u) AND G_FT(u) AND Ot3(u) and
         fi1_(u) and in11(u)
      => in6(v)

IV_in9: THEOREM %$$$ proven
  forall (u,v : state11, q : range(P) ) :
    step(q,u,v)  AND in9(u) AND G_FT(u) AND Ot3(u) AND
        fi1_(u)
     AND rA6_7(u) AND Ha2(u) AND nT12_13(u) AND mi9_10
        (u)
     AND Cu9_10(u) AND G_T0(u) AND G_T1(u) AND in3(u)
        AND G_T2(u)
     AND de7_(u) and rA5(u)
    => in9(v)

IV_in10: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
    step(q,u,v) AND in10(u) AND G_FT(u) AND de7_(u)
       AND in9(u)
     AND rA5(u) AND Ot3(u) AND fi1_(u)
     AND rA6_7(u) AND Ha2(u) AND nT12_13(u) AND mi9_10
        (u)
     AND Cu9_10(u) AND He3_4(u) AND in3(u)
    => in10(v)

IV_in11: THEOREM %$$$ proven
  forall (u:state12) :
       in10(u) and in2(u) and in3(u) and He3_4(u) and
           Cu1(u)
             AND Ha4(u) AND Cu9_10(u) AND Cu11_12(u)
                and fi5_(u)
       => in11(u)

IV_as2: THEOREM % proven
  forall (u,v : state11, q : range(P) ) :
        step(q,u,v) AND as2(u) AND as3(u) AND G_T0(u)
           AND Ot3(u)
                    and rA6_7(u) and Ha2(u) and rA5(u)
        => as2(v)

IV_as3: THEOREM % proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND as3(u) AND G_T0(u) AND Ot3(u)
           AND G_T1(u) and
                    rA6_7(u) and nT12_13(u)
        => as3(v)

IV_as7: THEOREM %$$$ proven
  forall (u,v : state11, q : range(P) ) :
    step(q,u,v)  AND as7(u) AND G_FT(u) AND Ot3(u) AND
        fi1_(u)
     AND rA6_7(u) AND Ha2(u) AND nT12_13(u) AND mi9_10
        (u)
     AND Cu9_10(u) AND G_T0(u) AND G_T1(u) AND G_T2(u)
         AND as3(u)
     AND de7_(u)  and rA5(u)
    => as7(v)

IV_as8: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
    step(q,u,v) AND as8(u) AND G_FT(u) AND de7_(u) AND
        as7(u)
     AND rA5(u) AND Ot3(u) AND fi1_(u)
     AND rA6_7(u) AND Ha2(u) AND nT12_13(u) AND mi9_10
        (u)
     AND Cu9_10(u) AND He3_4(u) AND as3(u)
    => as8(v)

IV_as9: THEOREM %$$$ proven
  forall (u:state12) :
       as8(u) and as2(u) and as3(u) and He3_4(u) and
          Cu1(u)
            AND Ha4(u) AND Cu9_10(u) AND Cu11_12(u)
               and fi5_(u)
       => as9(u)

IV_rA1: THEOREM % proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND rA1(u) AND Ha2(u)=> rA1(v)

IV_rA2: THEOREM %$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND rA2(u) AND Ot3(u)
        => rA2(v)

IV_rA3: THEOREM %$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND rA3(u) AND Ot3(u) and rA9(u)
           and He2_a(u) and He1(u) and rA2(u) and rA13
              (u)
        => rA3(v)
```

```
IV_rA4: THEOREM %proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND rA4(u) AND Ot3(u) AND nT14(u)
          => rA4(v)

IV_rA5: THEOREM %proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND rA5(u) AND Ot3(u) AND rA1(u)
           AND rA2(u)
            AND Ha3(u) AND He2_a(u)
          => rA5(v)

IV_rA6_7: THEOREM %$$proven
  forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND rA6_7(u) AND Ot3(u) and nT12_13(
         u)
         and nT14(u) and rA11(u) and mi4(u)
         and bu2_3(u) and Ha3(u) and mi6_a(u) and Ha2(
            u)
         and He3_4(u) and He2_b(u) and rA2(u)
        => rA6_7(v)

IV_rA8: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND rA8(u) and Ot3(u) AND bu4(u)
            AND nT14(u) AND mi6_a(u) AND Ne2(u)
                AND mi5(u)
     => rA8(v)

IV_rA9 : THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) and rA9(u) and Ot3(u) and Ha2(u)
           and nT14(u) and He1(u) and He2_b(u)
      => rA9(v)

IV_rA10: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND rA10(u) AND Ot3(u)
     => rA10(v)

IV_rA11: THEOREM %$$proven
```

```
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND rA11(u) AND Ot3(u) and nT12_13
           (u) and
           mi2(u)
        => rA11(v)

IV_rA12: THEOREM %$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND rA12(u) AND Ot3(u) and nT9_10(
           u)
        => rA12(v)

IV_rA13: THEOREM %$$ proven
  forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND rA13(u) AND Ot3(u) and rA5(u)
        => rA13(v)

IV_rA14: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND rA14(u) AND Ot3(u) AND rA4(u)
           AND He1(u)
            AND rA2(u)
     => rA14(v)

IV_nT1 : THEOREM %%% proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND nT1(u) AND Ot3(u) AND pr5(u)
          AND Ha3(u) and nT14(u) and nT16(u)
          AND Ha2(u)
      => nT1(v)

IV_nT2: THEOREM %$$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND nT2(u) and Ot3(u) and nT14(u)
             and Ha3(u) and rA5(u)
     => nT2(v)

IV_nT3: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND nT3(u) AND Ot3(u) AND nT9_10(u)
        => nT3(v)
```

```
IV_nT4: THEOREM %$$$ proven
  forall (u,v : state5, q : range(P) ) :
    step(q,u,v) AND nT4(u) AND Ot3(u) and Ha3(u) and
       de3(u)
         and nT12_13(u) and nT15(u) and rA5(u)
        => nT4(v)

IV_nT5: THEOREM %$$$ proven
  forall (u,v : state5, q : range(P) ) :
    step(q,u,v) AND nT5(u) AND G_FT(u) AND Ot3(u) and
       Ha3(u)
         and in3(u) and as3(u) and nT12_13(u) and nT15
            (u)
         and G_T0(u) and nT18(u) and mE3(u)
         and mi4(u) AND rA5(u)
        => nT5(v)

IV_nT6: THEOREM %$$$ proven
  forall (u,v : state5, q : range(P) ) :
    step(q,u,v) AND nT6(u) AND Ot3(u) and nT12_13(u)
         and nT14(u) and G_T0(u) and Ha3(u)
         and rA5(u)
        => nT6(v)

IV_nT7: THEOREM %$$$ proven
  forall (u,v : state5, q : range(P) ) :
    step(q,u,v) AND nT7(u) AND G_FT(u) AND Ot3(u) and
       nT12_13(u)
         and nT15(u) and G_T0(u) and rA6_7(u)
         and Ha2(u) and mi9_10(u) and nT14(u) and Ha3(
            u)
         and nT16(u) and rA5(u)
        => nT7(v)

IV_nT8: THEOREM %$$$ proven
  forall (u,v : state5, q : range(P) ) :
    step(q,u,v) AND nT8(u) AND G_FT(u) AND Ot3(u) and
       de3(u)
         and in3(u) and as3(u) and nT12_13(u) and nT15
            (u)
         and G_T0(u) and nT18(u) and mE3(u)
         and mi4(u) and Ha3(u) and mC2(u)
         and nT16(u) and nT2(u) and Ha2(u)
         and rA5(u)
        => nT8(v)

IV_nT9_10: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND nT9_10(u) and Ot3(u) and pr2_3
           (u)
            and nT18(u)
     => nT9_10(v)

IV_nT11: THEOREM %$$provens
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND nT11(u) and G_FT(u) and Ot3(u)
           AND pr4(u)
            AND nT16(u) AND mi8(u)
     => nT11(v)

IV_nT12_13: THEOREM  %$$ proven
  forall (u:state) :
        nT9_10(u) and Ha3(u) and He3_4(u)=> nT12_13(u)

IV_nT14: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND nT14(u) and Ot3(u) and nT9_10(
           u)
             and nT18(u) and nT16(u) and pr7(u)
     => nT14(v)

IV_nT15: THEOREM %$$ proven
  forall (u: state) :
        nT14(u) and Ha3(u) and nT2(u)=> nT15(u)

IV_nT16: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND nT16(u) and Ot3(u) AND pr8(u)
     => nT16(v)

IV_nT17: THEOREM %$$ proven
```

```
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND nT17(u) and G_FT(u) and Ot3(u)
            AND mi5(u)
             AND pr4(u) AND nT11(u) AND mi9_10(u)
     => nT17(v)

IV_nT18: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND nT18(u) and G_FT(u) and Ot3(u)
            and pr9(u)
          and mi5(u) and nT11(u)
     => nT18(v)

IV_nT19: THEOREM %$$ proven, next(index) is stable
   when process in [77,84].
   forall (u,v : state, p,q : range(P) ) :
        step(q,u,v) AND nT12_13(u) and pc(u)(p)>=77
           and pc(u)(p)<=84
             and next(u)(index(u)(p))/=0
     => next(v)(index(v)(p))/=0

IV_mi1_a: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND mi1_a(u) AND G_FT(u) and Ot3(u)
         AND mi9_10(u)
        => mi1_a(v)

IV_mi1_b: THEOREM %$$proven
   forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND G_FT(u) and mi1_b(u) AND Ot3(u)
         AND mi9_10(u)
        => mi1_b(v)

IV_mi2: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND mi2(u) AND Ot3(u) AND Ne4(u)
          => mi2(v)

IV_mi3: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND mi3(u) AND G_FT(u) AND Ot3(u)
```

```
            AND Ne5(u) AND fi11_(u) AND nT9_10(u)
         => mi3(v)

IV_mi4: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND mi4(u) AND Ot3(u) AND mi9_10(u
           )
            and mi3(u)
         => mi4(v)

IV_mi5: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND mi5(u) AND Ot3(u) AND nT9_10(u
           ) AND
            Ne5(u) and mi4(u) AND mi9_10(u)
         => mi5(v)

IV_mi6_a: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND mi6_a(u) AND G_FT(u) and Ot3(u)
        and mi5(u)
             and bu6(u) and rA8(u)
          => mi6_a(v)

IV_mi6_b: THEOREM %proven
   forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND mi6_b(u) AND G_FT(u) AND Ot3(u)
        and mi9_10(u)
            and bu4(u) and rA8(u) and mi5(u)
            and mi4(u)
          => mi6_b(v)

IV_mi7_a: THEOREM %$$ proven
   forall (u,v : state4, q : range(P) ) :
      step(q,u,v) AND mi7_a(u) AND Ot3(u) AND mi2(u)
            and mi7_b(u) AND mi4(u) AND nT18(u)
        => mi7_a(v)

IV_mi7_b: THEOREM %$$ proven
   forall (u,v : state4, q : range(P) ) :
      step(q,u,v) AND mi7_b(u) AND Ot3(u) AND Ne2(u)
```

```
           AND mi9_10(u) and nT17(u) AND mi3(u)
        => mi7_b(v)

IV_mi8: THEOREM %$$ proven
   forall (u,v : state4, q : range(P) ) :
      step(q,u,v) AND mi8(u) AND Ot3(u) AND mi9_10(u)
          AND Ne2(u) AND mi3(u)
        => mi8(v)

IV_mi9_10: THEOREM %$$ proven
   forall (u,v : state4, q : range(P) ) :
        step(q,u,v) AND mi9_10(u) AND Ot3(u) AND He3_4
           (u)
           AND nT9_10(u) AND nT18(u) AND Ne3(u)
           AND Ha3(u) AND mi3(u) AND nT17(u)
           AND G_T0(u) and He2_b(u) and mi4(u)
           AND mi12_a(u) and mi6_b(u) AND He6(u)
          => mi9_10(v)

IV_mi11: THEOREM %$$$ proven
  forall (u,v : state4, q : range(P) ) :
      step(q,u,v) AND mi11(u) AND Ot3(u) AND nT18(u)
         AND mi9_10(u) AND mi6_b(u) AND mi6_a(u)
        => mi11(v)

IV_mi12_a: THEOREM %$$$ proven
  forall (u,v : state4, q : range(P) ) :
      step(q,u,v) AND mi12_a(u) AND Ot3(u) AND rA8(u)
         AND nT2(u) AND He6(u) AND mi9_10(u)
         AND mi5(u) AND mi3(u) AND Ha3(u) AND mi4(u)
         AND rA5(u)
        => mi12_a(v)

IV_mi12_b: THEOREM %$$$ proven
  forall (u,v : state4, q : range(P) ) :
      step(q,u,v) AND mi12_b(u) AND Ot3(u) AND G_T0(u)
          and
         mi12_a(u) AND nT18(u) AND G_T1(u) AND mi6_a(u
           )
         AND Ha3(u) and mi4(u) and rA5(u)
        => mi12_b(v)
```

```
IV_mi13: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
      step(q,u,v) AND mi13(u) AND G_FT(u) and Ot3(u)
         AND rA6_7(u)
         AND Ha2(u) AND nT12_13(u) AND He3_4(u) AND
            mi9_10(u)
         AND mC9(u) and rA5(u)
        => mi13(v)

IV_mi14_15: THEOREM %$$$ proven
  forall (u:state6) :
      G_FT(u) AND Ne12_13(u) and mi5(u) and Cu15(u)
         and mi13(u)
       and Ot2(u) and He3_4(u) and G_T0(u) and
          Ne17_18(u)
       and G_T1(u) and Cu8(u) and He6(u)
       and He5(u) and mi4(u) AND Ot1(u)
    => mi14_15(u)

IV_mi16: THEOREM %$$$ proven
  forall (u: state6) :
     G_FT(u) AND Ne11(u) and mi5(u) and  mi4(u)
     => mi16(u)

IV_mi17_18: THEOREM %$$$ proven
  forall (u:state6) :
      G_FT(u) AND Ne15_16(u) and mi5(u) and Cu15(u)
         and mi13(u)
       and Ot2(u) and He3_4(u) and Ne17_18(u) AND Cu8(
          u)
       and He6(u) and He5(u) and mi4(u)
    => mi17_18(u)

IV_mi19: THEOREM %$$$ proven
  forall (u:state6) :
      G_FT(u) AND Ne20(u) and mi5(u) and Cu15(u) and
         mi13(u)
       and Ot2(u) and He3_4(u)
    => mi19(u)
```

```
IV_mi20: THEOREM %$$$ proven
  forall (u,v : state13, q : range(P) ) :
      step(q,u,v) AND mi20(u) AND G_FT(u) AND Ha3(u)
         AND Ha2(u)
          AND Cu8(u) and He6(u)
          AND He3_4(u) and Cu1(u) and Ha4(u) and He5(u
             )
          AND de3(u) and in3(u) and as3(u)
          AND rA8(u) and rA6_7(u)
          AND nT11(u) and  nT12_13(u)
          AND G_T0(u) and G_T1(u) and mi5(u)
          and mi4(u) AND in11(u)
          AND de7_(u) and Ot2(u) and fi1_(u)
          AND Cu9_10(u) and Cu13_14(u) and Cu15(u) and
              as9(u)
          and fi5_(u) and mC6(u) and Ne3(u)
          AND Ot3(u) and mC11(u) and mi13(u) and
             mi9_10(u)
          AND mC2(u) AND mE3(u) AND mE10(u)
          AND mE7(u) and mC12(u) and mE1(u)
          and mE13(u) and Ne17_18(u) AND mE2(u)
          and mE4(u) and Ot1(u) and mE6(u)
          AND Ne10(u) and rA5(u)
        => mi20(v)

IV_mC1: THEOREM %$$$ proven
  forall (u,v : state4, q : range(P) ) :
      step(q,u,v) AND mc1(u) AND Ot3(u) AND G_T0(u)
         and mi6_a(u)
         and G_T1(u) and mi11(u) AND nT18(u)
        => mc1(v)

IV_mC2: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND mC2(u) AND Ot3(u) AND rA6_7(u)
         and
         G_T0(u) AND nT12_13(u) AND mC2(u)
        => mC2(v)

IV_mC3: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
     step(q,u,v) AND G_FT(u) AND Ot3(u) AND mC3(u) and
         nT12_13(u)
        and G_T0(u) and rA6_7(u) and Ha2(u) and rA5(u)
        =>mC3(v)

IV_mC4: THEOREM %$$$ proven
  forall (u,v : state16, q : range(P) ) :
     step(q,u,v) AND G_FT(u) AND Ot3(u) AND mC4(u)
        and mC2(u) and mC3(u) and He3_4(u)
        and rA6_7(u) and Ha2(u) and rA5(u)
        =>mC4(v)

IV_mC5: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND mC5(u) AND Ot3(u)
        => mC5(v)

IV_mC6: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
      step(q,u,v) AND mC6(u) AND Ot3(u) AND rA6_7(u)
         and
         Ha2(u) AND nT12_13(u) AND mC2(u) and rA5(u)
        => mC6(v)

IV_mC7: THEOREM %$$$ proven
  forall (u,v : state7, q : range(P) ) :
      step(q,u,v) AND mC7(u) AND Ot3(u) AND rA6_7(u)
         and
         Ha2(u) AND nT12_13(u) AND mC2(u) and rA5(u)
        => mC7(v)

IV_mC8: THEOREM %$$$ proven
  forall (u,v : state1, q : range(P) ) :
      step(q,u,v) AND mC8(u) AND G_FT(u) AND Ot3(u)
         AND rA6_7(u) and
         Ha2(u) AND nT12_13(u) AND He3_4(u) AND G_T0(u
            )
         and mC7(u) and rA5(u)
        => mC8(v)

IV_mC9: THEOREM %$$$ proven
```

```
  forall (u,v : state1, q : range(P) ) :
      step(q,u,v) AND mC9(u) AND G_FT(u) AND Ot3(u)
         AND rA6_7(u) and
         Ha2(u) AND nT12_13(u) AND He3_4(u) AND mi9_10
            (u)
         AND He5(u) and mC7(u) and mC8(u) and rA5(u)
        => mC9(v)

IV_mC10: THEOREM %$$$ proven
  forall (u,v : state8, q : range(P) ) :
      step(q,u,v) AND mC10(u) AND G_FT(u) AND Ot3(u)
         AND rA6_7(u) and
         Ha2(u) AND nT12_13(u) AND mC2(u) and fi1_(u)
         AND mi6_a(u) AND Ha3(u) and mi4(u) and nT18(u
            )
         AND G_T0(u) and mE15(u) and mC11(u)
         and mi5(u) and rA5(u)
         => mC10(v)

IV_mC11: THEOREM %$$$ proven
  forall (u,v : state7, q : range(P) ) :
      step(q,u,v) AND mC11(u) AND Ot3(u) AND rA6_7(u)
         and
         Ha2(u) AND nT12_13(u) AND mC2(u) and rA5(u)
        => mC11(v)

IV_mC12: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND mC12(u) AND G_FT(u) AND Ot3(u)
         AND G_T0(u) and
         G_T1(u) AND G_T2(u) AND mC2(u) and
         mC11(u) and Cu9_10(u) and de7_(u) and mi9_10(
            u) and
         mC6(u)
        => mC12(v)

IV_mE1: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND mE1(u) AND Ot3(u)
        => mE1(v)
```

```
IV_mE2: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
       step(q,u,v) AND mE2(u) AND Ot3(u)
        => mE2(v)

IV_mE3: THEOREM %$$$ proven
  forall (u,v : state4, q : range(P) ) :
       step(q,u,v) AND mE3(u) AND mc1(u) AND Ot3(u)
         AND G_T0(u) and mi6_a(u) and G_T1(u) AND nT18
            (u)
        => mE3(v)

IV_mE4: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND mE4(u) AND Ot3(u) AND mE1(u)
     =>  mE4(v)

IV_mE5: THEOREM %$$$ proven
  forall (u,v : state9, q : range(P) ) :
     step(q,u,v) AND G_FT(u) AND Ot3(u) AND mE5(u)
        and mE3(u) and Ha3(u) and mi6_a(u) and mi4(u)
        and nT18(u) and Ha2(u) and rA5(u)
     =>mE5(v)

IV_mE6: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND mE6(u) AND Ot3(u)
     =>  mE6(v)

IV_mE7: THEOREM %$$$ proven
  forall (u,v : state9, q : range(P) ) :
      step(q,u,v) AND mE7(u) AND Ot3(u) and Ha2(u)
         AND Ha3(u) AND mi6_a(u) and mi4(u) and mE3(u)
         AND rA5(u)
        => mE7(v)

IV_mE8: THEOREM %$$$ proven
  forall (u,v : state6, q : range(P) ) :
       step(q,u,v) AND mE8(u) AND G_T0(u) AND Ot3(u)
         AND Ha3(u) and mi6_a(u) and mi4(u) AND nT18(u
            )
         AND Ha2(u) and mE3(u) AND rA5(u)
        => mE8(v)

IV_mE9: THEOREM %$$$ proven
  forall (u,v : state9, q : range(P) ) :
      step(q,u,v) AND mE9(u) and G_FT(u) AND G_T0(u)
         and Cu1(u) AND Ha4(u)
         AND Ot3(u) and Ha2(u) AND G_T1(u) AND Ha3(u)
         and mi6_a(u) and mi4(u) and mE3(u)
         and mC2(u) and mC10(u) and mE1(u)
         and mc1(u) and fi1_(u) and mi13(u) and mi12_b
            (u)
         and mC6(u) AND mE2(u) AND rA5(u)
        => mE9(v)

IV_mE10: THEOREM %$$$ proven
  forall (u,v : state9, q : range(P) ) :
      step(q,u,v) AND mE10(u) and fi1_(u) AND mE3(u)
         AND mi6_a(u) AND Ot3(u) and Ha2(u) AND G_T0(u
            )
         AND Ha3(u) and mi4(u) and  mE11(u)
         AND mE9(u) and mE7(u) and rA5(u)
        => mE10(v)

IV_mE11: THEOREM %$$$ proven
  forall (u : state9) :
       G_FT(u) AND mE10(u) and mi13(u) and mE16(u)
        and mi16(u) and mi5(u) and mE3(u)
        and Ne12_13(u) and mC12(u) and G_T0(u)
        and mE2(u) and mE1(u) and mE4(u)
        and mC6(u) and mE12(u) and mi12_b(u)
        and G_T1(u) and Cu13_14(u) and He3_4(u) and
           mi4(u)
       => mE11(u)

IV_mE12: THEOREM %$$$ proven
   forall (u: state) :
      G_FT(u) AND G_T0(u) and Ne23(u) and Ne22(u) and
         mE16(u)
         and He6(u) and Ne8(u)
      =>mE12(u)
```

```
IV_mE13: THEOREM %$$$ proven
  forall (u,v : state9, q : range(P) ) :
      step(q,u,v) AND mE13(u) AND Ot3(u) and Ha2(u)
         AND mE14(u) and fi1_(u) AND Ha3(u) AND mi6_a(
            u)
         and mi4(u) AND mE3(u) AND rA5(u)
     =>  mE13(v)

IV_mE14: THEOREM %$$$ proven
  forall (u,v : state9, q : range(P) ) :
      step(q,u,v) AND mE14(u) AND Ot3(u) and Ha2(u)
         AND G_T0(u) and fi1_(u) AND Ha3(u) AND mi6_a(
            u)
         and mi4(u) and nT18(u) AND mE3(u) and
         G_T1(u) and rA5(u)
     =>  mE14(v)

IV_mE15: THEOREM %$$$ proven
  forall (u,v : state6, q : range(P) ) :
      step(q,u,v) AND mE15(u) AND G_FT(u) AND Ot3(u)
         AND mE1(u) and
         Ha2(u) AND G_T0(u) and fi1_(u) AND Ha3(u) AND
             mi6_a(u)
         and mi4(u) and nT18(u) AND mE3(u) and
         mE2(u) and mE7(u) and mE14(u) and
         mE4(u) and rA5(u)
     =>  mE15(v)

IV_mE16: THEOREM %$$$ proven
  forall (u,v : state9, q : range(P) ) :
      step(q,u,v) AND mE16(u) AND Ha3(u) AND Ha2(u)
          AND mE3(u) and G_T0(u) and fi1_(u)
          AND mi6_a(u) and G_T1(u) and G_T2(u)
          AND mE2(u) and mE4(u) and mE1(u)
          and mE7(u) and mi4(u) and Ot3(u)
          and mE14(u) and mE13(u) and rA5(u)
        => mE16(v)

G_IV_prot0: THEOREM %$$ proven
   forall (u:state,q:range(P),i:range(2*P)) :
        G_protsize1(u,i)=size(remove(q,prSet1(u,i)))+
            size(remove(q,prSet2(u,i)))
            + IF member(q,prSet1(u,i)) THEN 1 ELSE 0
               ENDIF
            + IF member(q,prSet2(u,i)) THEN 1 ELSE 0
               ENDIF

G_IV_prot1: THEOREM %$$ proven
   forall (u,v : state, q : range(P),i:range(2*P)) :
        step(q,u,v) => G_protsize1(v,i)=size(remove(q,
           prSet1(u,i)))+
            size(remove(q,prSet2(u,i)))
            + IF member(q,prSet1(v,i)) THEN 1 ELSE 0
               ENDIF
            + IF member(q,prSet2(v,i)) THEN 1 ELSE 0
               ENDIF

IV_pr1: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
        step(q,u,v) AND pr1(u) AND G_FT(u) AND Ot3(u)
           AND rA11(u)
             AND rA10(u) AND nT9_10(u)
             AND Ne5(u) AND mi2(u) AND G_T0(u)
             AND mi8(u) AND mi5(u)
         => pr1(v)

IV_pr2_3: THEOREM %$$ proven
   forall (u,v : state, q : range(P)) :
     step(q,u,v) AND pr2_3(u) AND pr1(u) AND Ot3(u)
        AND rA11(u)
        AND mi1_b(u) AND rA6_7(u) and rA5(u)
=> pr2_3(v)

IV_pr4: THEOREM %$$ proven
   forall (u: state): pr1(u) =>pr4(u)

IV_pr5 : THEOREM %$$ proven
  forall (u: state) :
       pr6(u) and pr1(u) and bu1(u)=> pr5(u)

IV_pr6 : THEOREM %$$ proven
```

```
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) and pr6(u) and Ot3(u) and Ha2(u)
           AND nT9_10(u) and nT14(u) and nT16(u)
           AND He2_a(u) and rA2(u) and pr1(u) and bu1(
              u)
           AND pr10(u) and rA9(u) and He1(u) and rA4(u
              )
      => pr6(v)

IV_pr7_8_9: THEOREM %$$ proven
   forall (u: state): pr1(u) AND mi4(u)=>
               pr8(u) AND pr9(u) AND pr7(u)

IV_pr10 : THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) and pr10(u) and Ot3(u) and pr1(u)
           AND nT9_10(u) and nT14(u) and nT16(u)
           AND nT17(u)
      => pr10(v)

G_IV_busy0: THEOREM %$$ proven
   forall (u:state,q:range(P),i:range(2*P)) :
        G_busysize1(u,i)=size(remove(q,buSet1(u,i)))+
            size(remove(q,buSet2(u,i)))
            + IF member(q,buSet1(u,i)) THEN 1 ELSE 0
               ENDIF
            + IF member(q,buSet2(u,i)) THEN 1 ELSE 0
               ENDIF

G_IV_busy1: THEOREM %$$ proven
   forall (u,v : state, q : range(P),i:range(2*P)) :
        step(q,u,v) => G_busysize1(v,i)=size(remove(q,
           buSet1(u,i)))+
            size(remove(q,buSet2(u,i)))
            + IF member(q,buSet1(v,i)) THEN 1 ELSE 0
               ENDIF
            + IF member(q,buSet2(v,i)) THEN 1 ELSE 0
               ENDIF

IV_bu1: THEOREM %$$ proven
   forall (u,v : state, q : range(P) ) :
         step(q,u,v) AND bu1(u) AND G_FT(u) AND Ot3(u)
            AND rA11(u)
              AND rA10(u) AND nT9_10(u)
              AND Ne5(u) AND mi2(u) AND G_T0(u)
              AND mi8(u) AND mi5(u) AND bu5(u)
          => bu1(v)

IV_bu2_3: THEOREM %$$ proven
   forall (u: state): bu1(u) AND Ot3(u) AND rA10(u)
       =>bu2_3(u)

IV_bu4: THEOREM %$$ proven
   forall (u: state): bu1(u)=>bu4(u)

IV_bu5: THEOREM %$$ proven
   forall (u,v : state, q : range(P)) :
       step(q,u,v) AND bu5(u) AND Ot3(u) AND nT9_10(u)
         AND nT16(u) AND nT18(u) AND pr1(u) AND
             bu1(u)
         => bu5(v)

IV_bu6: THEOREM %$$ proven
   forall (u: state): bu1(u) and mi4(u)=>bu6(u)

IV_Ot1: THEOREM % proven
   forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND Ot1(u) AND fi1_(u) => Ot1(v)

IV_Ot2: THEOREM %proven
   forall (u,v : state, q : range(P) ) :
      step(q,u,v) AND Ot2(u) AND fi1_(u) => Ot2(v)

IV_Ot3: THEOREM % @proven
   forall (u,v : state, q : range(P)) :
      step(q,u,v) AND Ot3(u) => Ot3(v)

IV_Ot4: THEOREM % include all labels in the program
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND Ot3(u) AND Ot4(u) =>Ot4(v)

% Global inductive invariant
```

```
INV(s:state):bool=
    He3_4(s) and He6(s) and Cu8(s) and mi4(s)
         and nT2(s) and mi12_a(s) and mi12_b(s)
         and mC6(s) and mE8(s) and mE3(s)
         and de13_(s) and fi4(s) and de3(s) and in3(s)
         and as3(s) and fi3(s) and de2(s) and in2(s)
         and as2(s) and mi5(s) and Cu1(s) and Ha3(s)
    AND
         Ot3(s) and fi1_(s) and Ot2(s) and Ot1(s) and
            Ha2(s)
         and Ha1(s) and He2_a(s) and rA4(s) and He2_b(
            s) and He1(s)
         and rA2(s) and Ha4(s) and mi1_b(s) and Ne1(s)
         and Ne2(s) and Ne3(s) and Ne4(s)
         and Ne5(s) and mi7_a(s) and mi7_b(s)
         and mi8(s) and mi9_10(s) and mi2(s)
         and mi6_a(s) and rA5(s) and mi3(s)
         and mi6_b(s) and nT3(s) and nT9_10(s)
         and nT18(s) and nT14(s) and nT16(s)
         and nT11(s) and nT17(s) and nT12_13(s) and
            nT15(s)
         and nT4(s) and nT8(s) and nT5(s) and nT6(s)
         and nT7(s) and nT1(s) and rA8(s)
         and rA10(s) and rA6_7(s) and rA9(s)
         and rA11(s) and rA14(s) and rA12(s)
         and pr1(s) and pr5(s) and pr6(s) and pr10(s)
         and bu1(s) and pr2_3(s) and bu2_3(s) and bu6(
            s)
         and pr7(s) and pr9(s) and pr8(s) and bu5(s)
         and pr4(s) and bu4(s) and mi13(s) and mi12_b(
            s)
         and mi12_a(s) and mi14_15(s) and Ne12_13(s)
            and Ne14(s)
         and mi16(s) and Ne11(s) and mi17_18(s) and
            Ne15_16(s)
         and Ne17_18(s) and Ne19(s) and mi19(s) and
            mi20(s)
         and Ne20(s) and mC9(s) and mi1_a(s) and Ne10(
            s)
         and mC2(s) and mC5(s) and mC7(s)
         and mC8(s) and mC10(s) and mC11(s)
         and mC12(s) and mE1(s) and mc1(s)
         and mi11(s) and mE2(s) and mE10(s)
         and mE11(s) and rA13(s) and mE16(s)
         and mE9(s) and mE7(s) and mE12(s)
         and mE15(s) and mE14(s) and mE13(s)
         and mE4(s) and mE6(s) and He5(s) and fi11_(s)
         and Cu9_10(s) and Cu11_12(s) and Cu13_14(s)
            and Cu15(s) and fi10(s)
         and fi9(s) and fi8(s) and de7_(s) and fi2(s)
            and fi7(s)
         and fi6(s) and de11(s) and de10(s) and de9(s)
         and de5(s) and de6(s) and in11(s) and in10(s)
         and in9(s) and in5(s) and in6(s) and as9(s)
         and as8(s) and as7(s) and fi5_(s) and rA3(s)
         and Cu2(s) and Cu6(s) and Cu7(s) and Ne6(s)
         and Ne9a(s) and Ne23(s) and Ne22(s) and Cu4(s
            )
         and Ne9(s) and Cu3(s) and Ne7(s) and Ne8(s)
         and Cu16(s) and Ne25(s) and Ne26(s) and Ne27(
            s)
         and Ne24(s) and Co1(s) and Co2(s) and Co3(s)
         and No1(s) and No2(s) and Cn1(s) and Cn6(s)
            and Cn5(s)
         and Cn2(s) and Cn7(s) and Cn8(s) and Cn3(s)
            and Cn10(s)
         and Cn9(s) and Cn4(s) and Cn11(s) and Ot4(s)
            and de4_(s)
         and mC4(s) and mC3(s) and mE5(s)
         and rA1(s)

IV_INV: THEOREM %$$$ proven
  forall (u,v : state, q : range(P) ) :
     step(q,u,v) AND INV(u) => INV(v)

%progress

IV_prog: THEOREM %$$$ proven
  forall (u : state, q:range(P)) :
     INV(u) => exists (v:state): pc(u)(q)/=pc(v)(q)
        and step(q,u,v)
```

```
%Initial state
St0: state
Ha0: Hashtable_=(# size:=4*P+2, occ:=0, dels:=0, bound
   :=1+2*P,
                  table_:=lambda (k:below(4*P+2)):nor
                     (null) #)

Init:{s:state|
   (forall (p:range(P)):
       pc(s)(p)=0 and index(s)(p)=currInd(s) and
          return_getAccess(s)(p)=10 and
       return_releaseAccess(s)(p)=0 and return_refresh
          (s)(p)=10 and
       return_newTable(s)(p)=30 and a_insert(s)(p)=ADR
          (v_insert(s)(p)) and
       a_assign(s)(p)=ADR(v_assign(s)(p)) and v_assign
          (s)(p)/=null and
       v_insert(s)(p)/=null and a_find(s)(p)/=
          nullAddress and
       a_delete(s)(p)/=nullAddress and val(r_find(s)(p
          ))=rS_find(s)(p) and
       h_releaseAccess(s)(p)=0) AND
   (forall (a:Address):
       X(s)(a)=null) AND
   H_index(s)=2 AND currInd(s)=1 AND
   H(s)(currInd(s))=1  AND i?(Heap(s)(H(s)(currInd(s))
      )) AND
   (forall (k: below(size(i_(Heap(s)(H(s)(currInd(s)))
      )))):
       table_(i_(Heap(s)(H(s)(currInd(s)))))(k)=nor(
          null) and Y(s)(k)=nor(null)) AND
   size(i_(Heap(s)(H(s)(currInd(s)))))=4*P+2 AND
   dels(i_(Heap(s)(H(s)(currInd(s)))))=0 AND
   occ(i_(Heap(s)(H(s)(currInd(s)))))=0 AND
   bound(i_(Heap(s)(H(s)(currInd(s)))))=2*P+1  AND
   next(s)(currInd(s))=0 AND
   prot(s)(currInd(s))=1 AND busy(s)(currInd(s))=1 AND
   (forall (i:range(2*P)): (i/=currInd(s) =>
                        H(s)(i)=0 and Heap(s)(H(s)(i)
                           )=bot and
```

```
                         prot(s)(i)=0 AND busy(s)(i)
                            =0)) AND
   NOT(i?(Heap(s)(0))) }

IV_Init: THEOREM
    INV(Init)

END hashtable7
```

Appendix B

PVS Cheat Sheet

In the next page.

PVS Cheat Sheet

Theories

```
function_properties[D, R: TYPE]: THEORY
BEGIN
    % this is a comment

    f, g: VAR [D -> R]
    x, x1, x2: VAR D
    y: VAR R

    injective?(f): bool = (FORALL x1, x2: (f(x1) = f(x2)) => (x1 = x2))
    surjective?(f): bool = (FORALL y: (EXISTS x: f(x) = y))
END function_properties

finite[T: TYPE]: THEORY
BEGIN
    IMPORTING function_properties

    is_finite: bool = (EXISTS (n: nat), (f: [upto[n] -> T]): surjective?(f))
END finite
```

Constants

`some: int`	An undefined integer constant
`some: int = 10`	The integer number 10
`abs(x: int): nat = if x >= 0 then x else -x endif`	Function definition

Expressions

`=`	Equality
`/=`	Inequality
`true, false, and, or, not, =>, <=>, FORALL, EXISTS`	Logical
`if x = 0 then 1 elsif x = 1 then 1 else x / 2 endif`	If expression
`CASES x OF`	Pattern matching
` lst(val, nxt): lst(1, x)`	
` ELSE lst(1, null)`	
`ENDCASES`	
`(lambda x: x + 1)`	Function expression
`f with [(0) := 1, (1) := 0]`	Function update
`{x: int | x < 10}`	Set expression
`(# amount := 10, curr := EUR #)`	Record construction
``(# amount := 10, curr := EUR #)`amount``	Record field access
`(# amount := 10, curr := EUR #) with [amount := 0]`	Record update
`(1, true, (lambda x: x + 1))`	Tuple construction
`proj_3((1, true, (lambda x: x + 1)))`	Tuple projection
`(1, true, (lambda x: x + 1)) with [2 := false]`	Tuple update
`a + b:nat`	Type coercion

Variables

`x, y, z: VAR int`	Three named variables of `int` type
`f: VAR [int -> int]`	A variable function
Usage	
`abs(x): nat = if x >= 0 then x else -x endif`	Avoided type declaration
`bound(x, y, z, f): bool = f(x, y) > z`	Not necessarily the same `x`

Types

`foo: TYPE`	Uninterpreted type
`spam: TYPE+`	Non-empty type
`positive_int: TYPE = {x: integer | x > 0}`	Subtype
`bin_int_f: TYPE = [int, int -> int]`	Function type
`bin_int_f: TYPE = ARRAY[int, int -> int]`	Equivalent to above
`tup: TYPE = [int, bool, [int -> int]]`	Tuple type
`currency: TYPE = {USD, EUR, JPY}`	Enumeration type
`value: TYPE = [# amount: nat, curr: currency #]`	Record type (struct)

Recursive and higher-order types

A higher-order type can be thought of like a generic type in Java or C++.

```
linked_list[T: TYPE]: DATATYPE
BEGIN
    null: null?
    lst (value: T, next: linked_list): lst?
END linked_list

a: linked_list[nat] = lst(1, lst(2, lst(3, null)))
empty: linked_list[bool] = null
```

The recursion is the `lst ...: lst?` part of the datatype definition.

Recursive functions

```
factorial(n: nat): RECURSIVE nat =
    if n = 0 then 1 else n * factorial(n-1) endif
MEASURE (LAMBDA n: n)  % the measure must always decrease
```

Prover commands

Control	
`(quit)`	Close prover session
`(postpone)`	Defer evaluation of this proof branch
`(undo)`	Revert the changes of the previous command
`(help)`	General help
`(help` *`command`*`)`	Help on a specific command
Strategies	
`(grind)`	And pray for a `Q.E.D.`
`(tcc)`	Apply common rules for type-checking conditions
`(assert)`	Simplify using decision procedures
Structural Rules	
`(copy` *`fnum`*`)`	Copy the formula *`fnum`*
`(delete` *`fnum`*`)`	Delete the formula *`fnum`*
Propositional Rules	
`(bddsimp)`	Propositional simplification using BDDs
`(case "`*`condition`*`")`	Split into cases; e.g. `(case "x > 0")`
`(flatten)`	Simplify disjunctions
`(lift-if)`	Lift embedded *if* expressions
`(prop)`	Propositional simplifications
Quantifier Rules	
`(skolem!)`	Skolemize with generated names
Using definitions and lemmas	
`(expand "`*`name`*`")`	Apply the definition of *`name`*
`(use "`*`name`*`")`	Apply lemma *`name`*