Cheney's Copying GC

Iterative copying cheaper than recursive copying

Basic copying collector

- Uses recursive call to copy
  - Recursive calls costs CPU time
  - Recursion stack occupies precious space
- Alternative:
  - Cheney's iterative copying collector
  - Just 2 pointers are needed: scan and free
  - Remember branch points of active graph as a queue
  - scan and free point to opposite ends of queue
    - Stored in new semi-space in objects already copied
  - Use tricolor abstraction

Cheney’s copying collector

- Immediately reachable objects form initial queue of objects for a breadth-first traversal
- scan pointer is advanced from first object location to end of scanned objects.
- Each encounter of pointer into from-space, pointee is copied to the end of the queue (in to-space) and the pointer to the object is updated

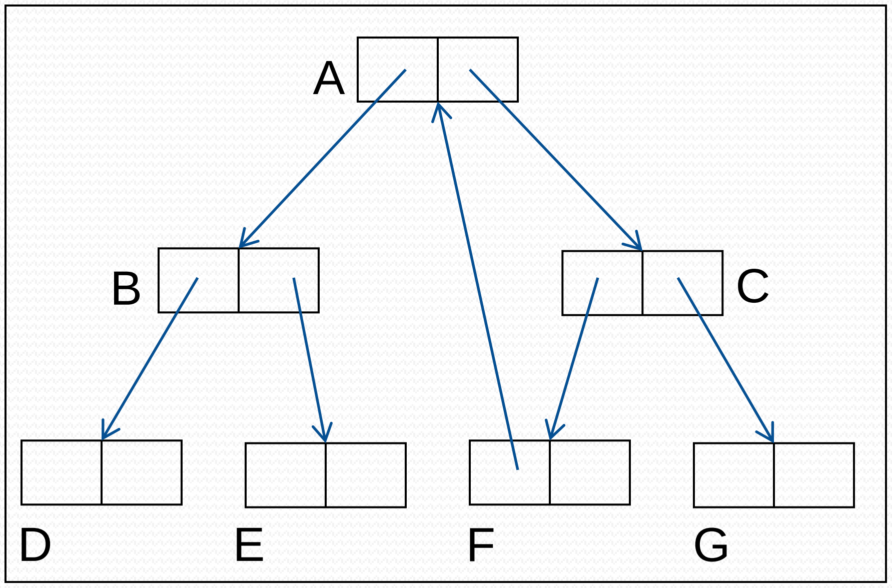
Cheney’s copying collector

- When an object is copied to to-space, a forwarding pointer is installed in the old version of the object
- The forwarding pointer signifies that the old version of object is obsolete and indicates where to find replica

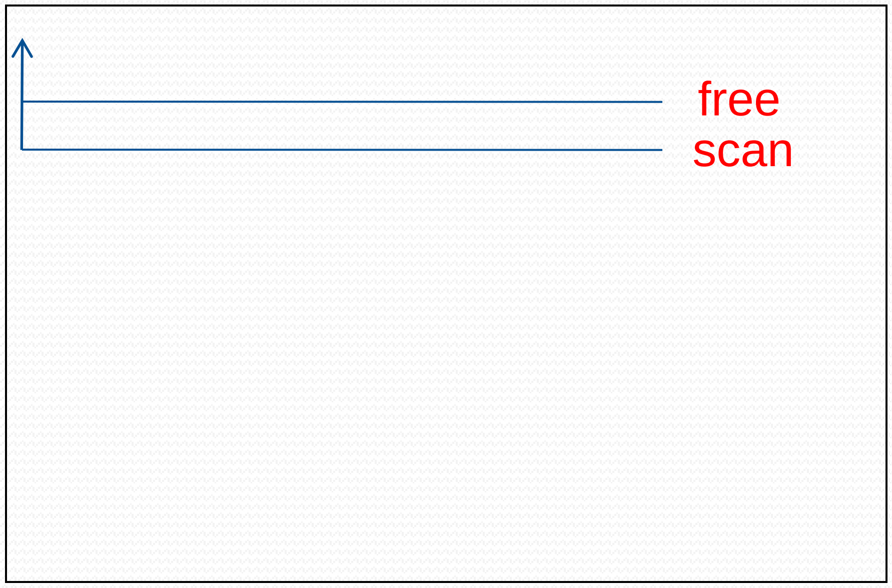
Cheney's tricolor abstraction

- Black:
  - Object scanned--object & immediate descendents visited
  - GC finished with black objects, will not visit them again
- Grey:
  - Object is visited but its descendents may not have been scanned
  - Collector must visit it again
- White
  - Object not visited and is garbage at end of tracing phase

Cheney's algorithm after the flip

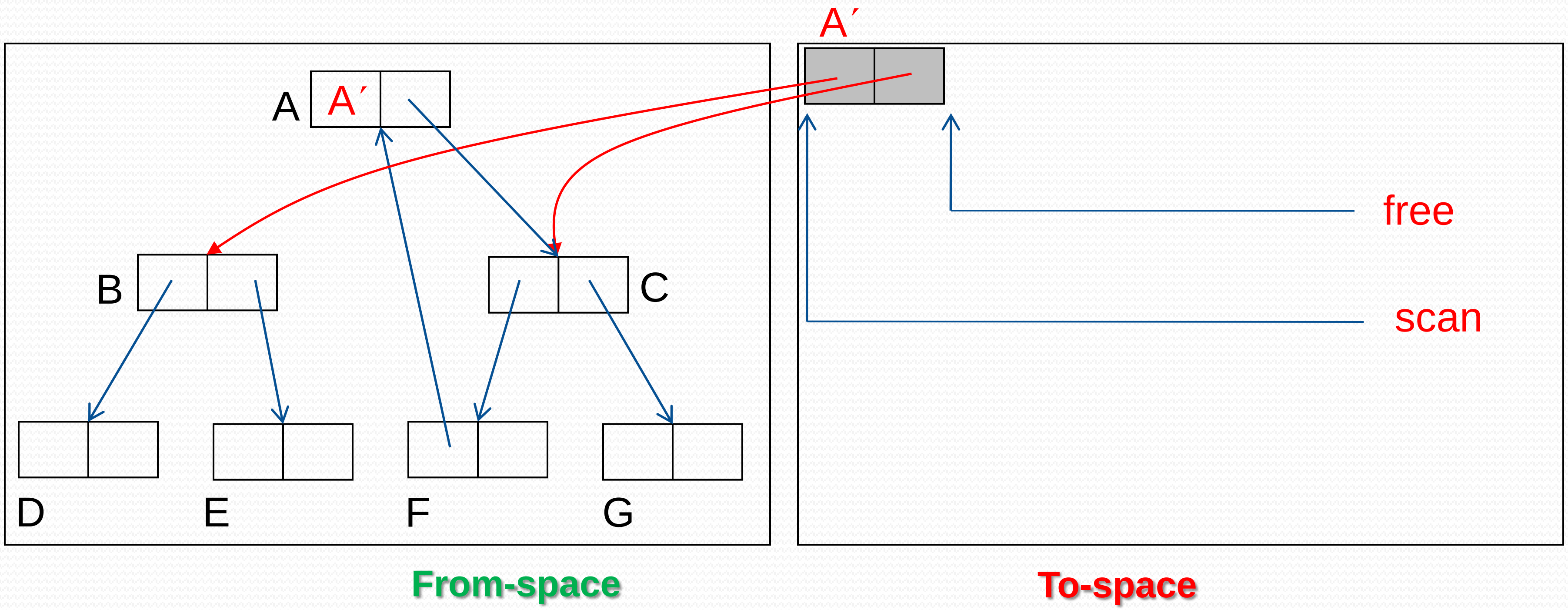


From-space



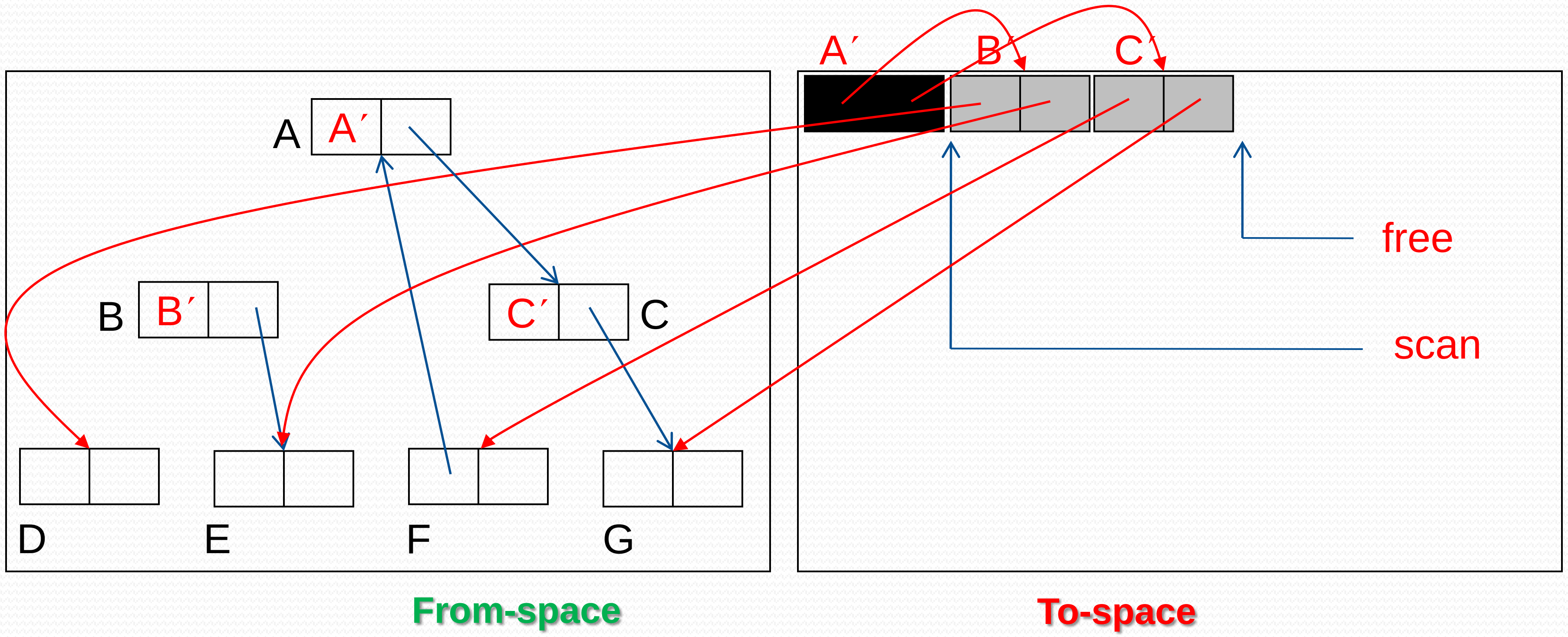
To-space

Roots of structure copied



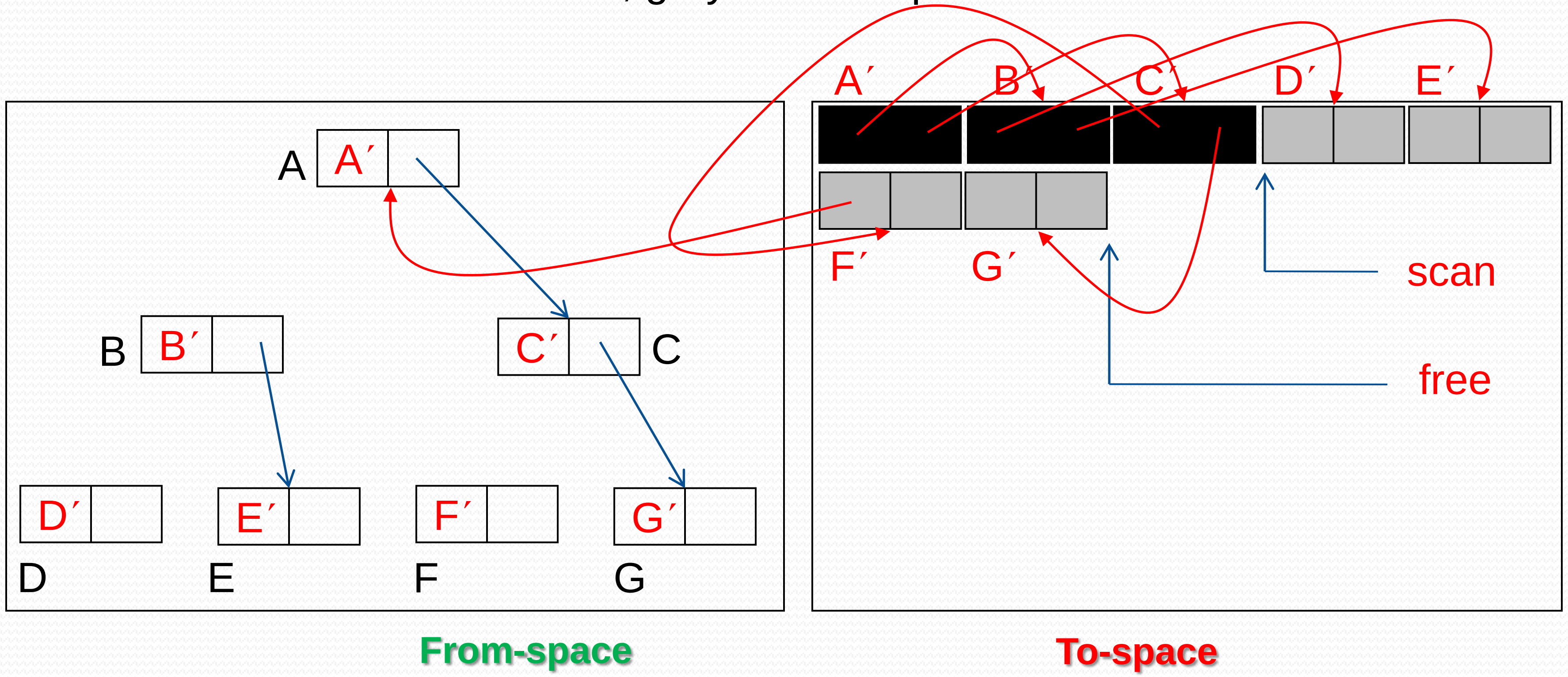
A′ scanned, copying B and C

Black nodes have been scanned; grey nodes copied but not scanned



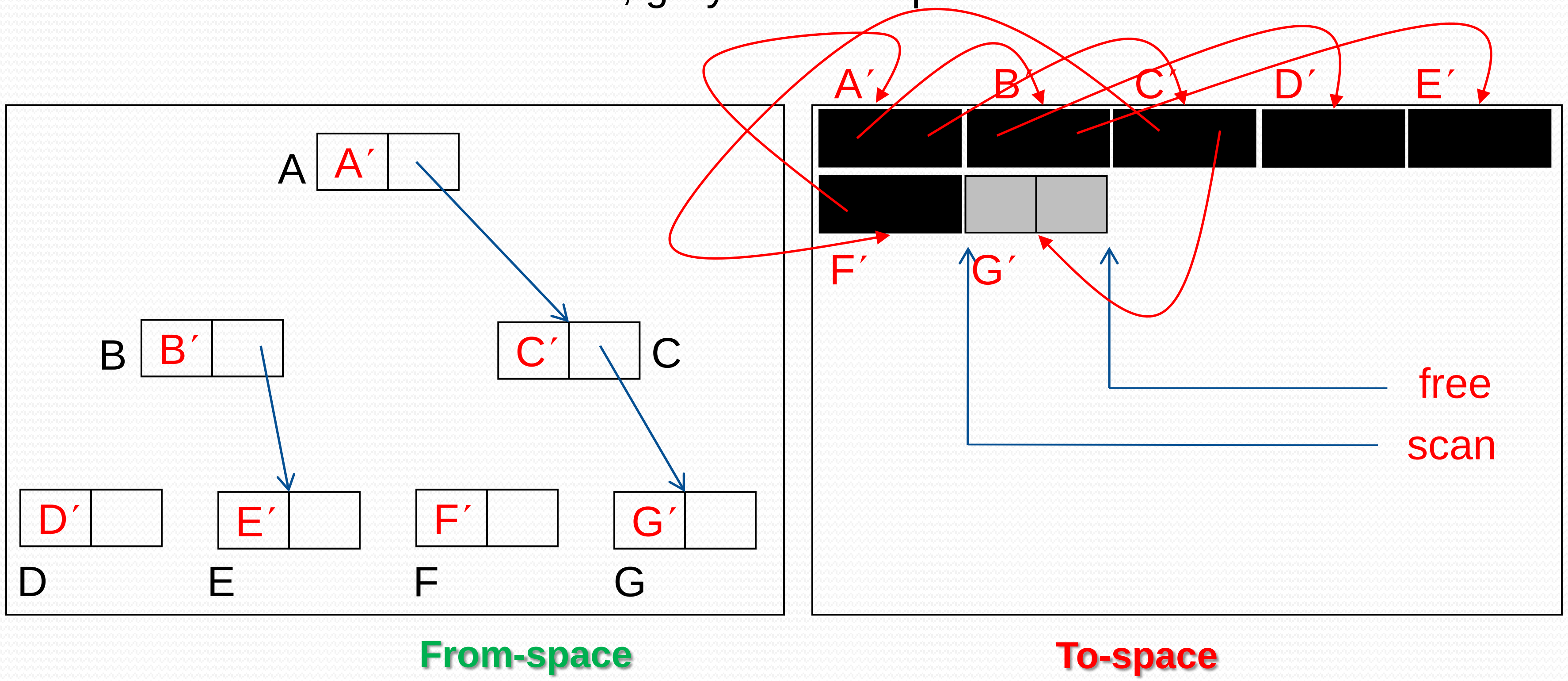
All from-space objects copied

Black nodes have been scanned; grey nodes copied but not scanned

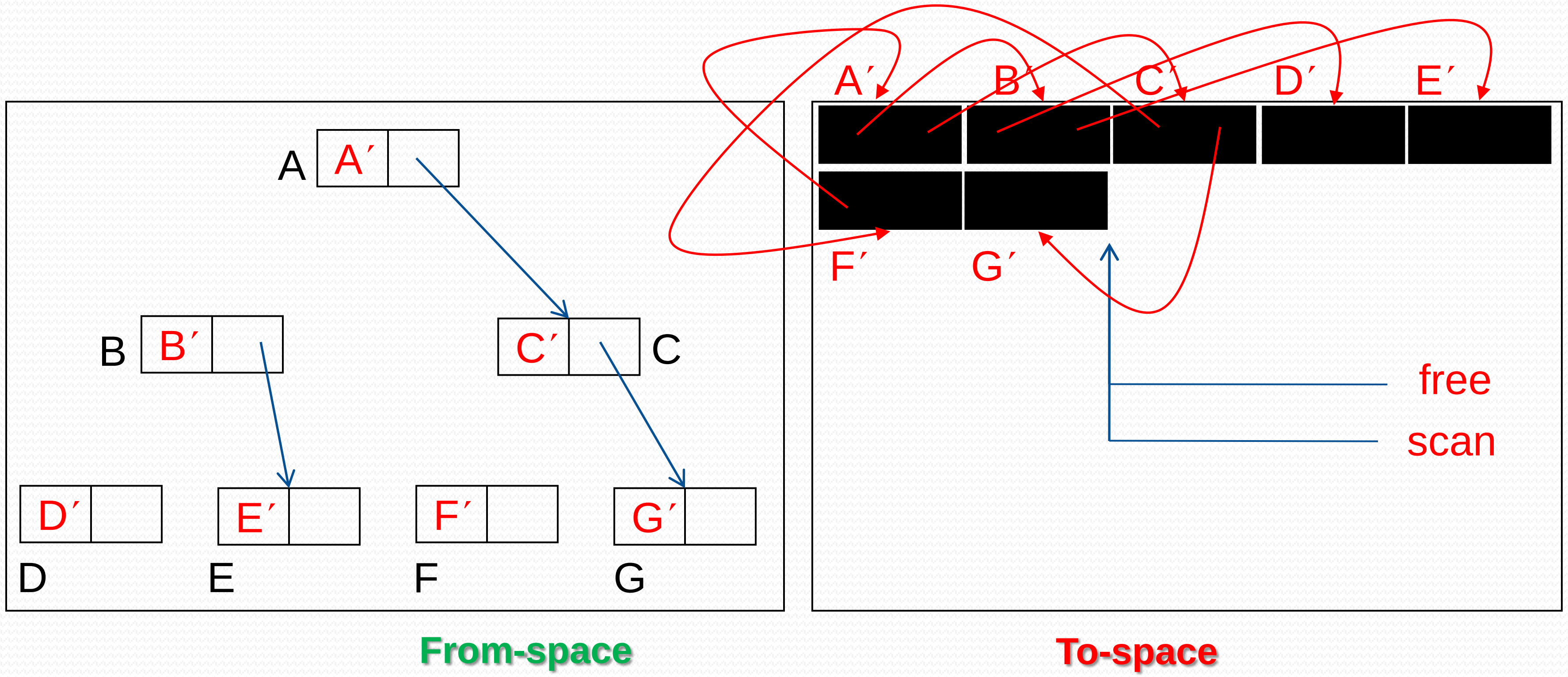


left(F) updated

Black nodes have been scanned; grey nodes copied but not scanned



Algorithm terminates



Cheney’s algorithm

```
flip() {
    to_space, from_space  = from_space, to_space
    top_of_space = to_space + space_size
    scan = free = to_space
    for R in Roots
            R = copy(R)                    // Root pointer now points to copy of R
    while scan < free
            for P in children(scan)
                    *P = copy(*P)
            scan = scan + size(scan)
}
```

Cheney's algorithm

```
copy() {
    if forwarded(P)
            return forwarding_address(P)
    else
            addr = free
            move(P free)
            free = free + size(P)
            forwarding_address(P) = addr
            return addr
}
```