The Admissibility of A^*

Here I want to prove that, in fact, algorithm A^* is admissible. That is, it will terminate in finding an optimal path to a goal node if a path from start to a goal exists. In particular:
Given: a graph search which does algorithm A search – if I can guarantee that the heuristic function has the property $h(n) \leq h\prime(n)$ for all nodes n (so that my heuristic function which estimates the cost from a given node to a goal is actually a lower bound on the actual cost to the goal), then the algorithm as given is guaranteed to find an optimal solution (if a path from start to a goal exists).

Remember in algorithm A search is distinguished by two things; the heuristic function it uses and what the implementation of that heuristic requires of the search package itself:

1. The heuristic function that we use estimates: $f\prime(n)$ = the actual cost of an optimal path (a minimal cost path to the "cheapest" goal) from start to a goal node constrained to go through n. We saw that this function was composed of two parts: $f\prime(n) = g\prime(n) + h\prime(n)$ = the actual cost of an optimal path from s to n + the actual cost of an optimal path from n to a goal node.

   We estimate $f\prime$ with the function f:

   $$f(n) = g(n) + h(n)$$

   where:
   $g(n)$ is an estimate of $g\prime(n)$ – here we will use the minimal cost path found so far. So it will always be the case that $g\prime(n) \leq g(n)$, and
   $h(n)$ is an estimate of $h\prime(n)$ – here we will use a heuristic information from the problem domain.

2. The algorithm itself:

   - nodes taken off open list according to the smallest value of $f(n)$
   - when generating the successors of a node, check to see if any have been generated before (and thus have a different parent). Check to see of their g values would be better through the new parent. If so, update the node and possibly all of that node's children as well.

Now, Algorithm A^* is an algorithm A search, where constraints are put on the h function. In particular, algorithm A^* requires:

$$h(n) \leq h\prime(n)$$

Given an h function that satisfies these constraints, we must prove that Algorithm A will find a cheapest path to a goal node (an optimal solution). So we want to **prove** that Algorithm A^* is admissible (if there is a path from start to a goal node, A^* terminates by finding an optimal path). We start by taking two things for granted (both can themselves be proved, but given time.....)

Given: The graphsearch as described always terminates for finite graphs (if no goal – we will at least run out of things on open since there are only a finite number of nodes in the graph there are a finite number of things to be put on the open list).

Given: Even if the graph is infinite then the graph search will terminate if there is a path from start to the goal node. (Here the basic reasoning is that some element of that path will always be of the open list. This is because the start node (at the beginning of the path) starts off on open. The second node on the path is a child of start, and then each are children of the previous. Notice that this path must be finite. If the graph is infinite, then at some point the g portion of the f function for the nodes that are not on this path but are on open will become quite large (because they are on an infinite path). At that time, the g values of the nodes to the goal will appear more reasonable to the algorithm, and will eventually be chosen off of the open for expansion.)

Prove Algorithm A^* is admissible (if there is a path from S to a goal node, A^* terminates by finding an optimal path).

First: A^* can either terminate by finding a goal or depleting OPEN. But OPEN can never become empty before termination if there is a path from start to a goal node because one of the nodes along the path will always be on open. [Note: suppose this is the optimal path to the goal node n_k, optimal-path=$(Start = n_0, n_1, ..., n_k = Goal)$. Notice that S starts out on open and each step along the way one of the elements of the path will be on open – since they are all children of the previous node by definition of path. The only way that no member of this path could be on open is if n_k were already on closed. But since n_k is a goal node, the algorithm would have terminated at that point!]

Therefore, the algorithm must terminate in finding a goal node. I will go on to prove by contradiction that the algorithm will terminate in finding an optimal path to a goal.

Suppose that the algorithm does terminate at some goal node, t, without finding an optimal path. Then it must be the case that:

$$f(t) = g(t) > f\prime(start)$$

Note that $f(t) = g(t)$ by our assumption of $h(t) \leq h\prime(t)$. Since t is a goal node, $h(t)$ must be 0.

Note that $f(t) > f\prime(start)$ by the definition of optimal path. One thing that we should notice is that the $f\prime$ value is the same for each node along the optimal path. This makes sense if we note that $f\prime$ is capturing the cost of the entire path (both to a particular node and from there to the goal node). Along the optimal path, every node would have the same value for this function.

Now, by our assumption above (that we terminate at a goal node but that we have not found an optimal path) and because we know that some member of the optimal path must always be on open (from above), it must be the case that $\exists$ a node ϵ of the optimal path that is on the open list at the time t is chosen. Let's call that node n_i. Now, since n_i was not chosen from the open list when t was, we know that n_i must have a greater f value. Thus it must be the case that:

$$f(n_i) \geq f(t) = g(t) > f\prime(start)$$

It is this statement that I will find a contradiction with. I will show that this statement cannot hold because there must $\exists$ on open a node n_i that is on an optimal path from start to a goal with $f(n_i) \leq f\prime(start)$. This will create a contradiction and my original premise must be wrong.

Creating the Contradiction. Suppose that my optimal path is
optimal-path=$(Start = n_0, n_1, ..., n_k = Goal)$.
For any time before termination, let n_i be the lowest element in this sequence on the open list. Thus we know that:

$$f(n_i) = g(n_i) + h(n_i)$$

But we know that the algorithm has already found an optimal path to n_i since n_i is on an optimal path to a goal and all of its ancestors along this path are on the closed list since n_i is the lowest element of the path still on open.

[Note that if a node is on the optimal path from start to a goal, then that same path (prior to the node) must be the optimal path from start to each node on the path – of course, if there was a cheaper way to get to a node within that path, there would also be a cheaper way to get to the goal. Now, if all nodes previous to n_i have already been explored (made Expl) that means that the optimal path to n_i has already been found (in those previous nodes!).]

Therefore we know that:

$$g(n_i) = g\prime(n_i)$$
and thus
$$f(n_i) = g\prime(n_i) + h(n_i)$$

But notice that since we know that $h(n_i) \leq h\prime(n_i)$, we have:

$$f(n_i) \leq g\prime(n_i) + h\prime(n_i) = f\prime(n_i)$$

But note that the $f\prime$ value for any node along the optimal path is equal to $f\prime(start)$. This means that:

$$f(n_i) \leq f\prime(start)$$

This contradicts our above statement. And thus, n_i must have been chosen off of the open list before t!