Journal of Heuristics, 5, 353–376 (1999)


# Completeness and Admissibility for General Heuristic Search Algorithms—A Theoretical Study: Basic Concepts and Proofs

HENRI FARRENY
*Institut de Recherche en Informatique de Toulouse (IRIT), 118 Route de Narbonne, 31062 Toulouse cedex, France*
*email: henri.farreny@irit.fr*

*Abstract*

We propose a formal generalization for various works dealing with Heuristic Search in State Graphs. This generalization focuses on the properties of the evaluation functions, on the characteristics of the state graphs, on the notion of path length, on the procedures that control the node expansions, on the rules that govern the update operations. Consequently, we present the algorithm family $\varrho$ and the sub-family $\tilde{A}$, which include Nilsson's A or A* and many of their successors such as HPA, B, $A^*_\varepsilon$, $A_\varepsilon$, C, BF*, B′, IDA*, D, A**, SDW. We prove general theorems about the completeness and the sub-admissibility that widely extend the previous results and provide a theoretical support for using diverse kinds of Heuristic Search algorithms in enlarged contexts, specially when the state graphs and the evaluation functions are less constrained than ordinarily.

**Key Words:** heuristic search, ordered search, algorithms A and A*, admissibility

## 1. Introduction

This paper is a part of a study (Farreny, 1995, 1996a,b,c, 1997a,b,c) that aims to compare and to extend various works dealing with Heuristic Search in State Graphs. Our main goal here is to present the proofs of general results concerning two basic properties: 1) the completeness and 2) the admissibility or sub-admissibility.

### *1.1. Position of this paper and contribution*

Heuristic Search algorithms are studied from about thirty years; Steward, Liaw, and White (1994) list one thousand papers concerning this topic; among them, several hundreds refer to Heuristic Search in State Graphs. What do we propose here? We do not propose another particular algorithm but an unifying and generalizing point of view. We formally characterize some wide families of algorithms that are *complete* or *admissible* or *sub-admissible*.[1] These families include, for instance, the algorithms A and A* (Hart, Nilsson, and Raphael, 1968; Nilsson, 1971, 1980) and diverse successor algorithms[2] such as HPA (Pohl, 1969, 1970, 1977), another extended form of A (Pohl, 1973), a specific form of A (Harris, 1973, 1974), B (Martelli, 1977), $A^*_\varepsilon$ (Pearl and Kim, 1982, 1984), $A_\varepsilon$ (Ghallab, 1982; Ghallab and Allard, 1982, 1983), C (Bagchi and Mahanti, 1983, 1985), BF* (Pearl, 1984; Dechter and Pearl,

1985, 1988), B′ (Mero, 1984), IDA* (Korf, 1985a,b, 1988), D (Mahanti and Ray, 1988), A** (Dechter and Pearl, 1985, 1988) and SDW (Köll and Kaindl, 1992). We do not evoke here an application or an experimentation but we present original results concerning some general families of Heuristic Search algorithms. Obviously, applications and experimentations are a very important topic. The papers that describe actual applications of Heuristic Search in State Graphs to concrete domains (such as robotics, natural language understanding, pattern recognition, etc.) are relatively few; the papers that present experimentations with symbolic problems are far more numerous; most of them cope with the Travelling Salesman Problem or with the $n$-puzzle (generally $n = 8$ or 15, seldom $n = 24$ or more). As teacher as much as researcher we are very interested in this kind of works. Nevertheless, in the present paper the contribution is exclusively theoretical. Indeed, we are also very interested in the exact statements and proofs of the algorithm properties. But, at times, the statements of the properties of some Heuristic Search algorithms or the statements of their proofs are incomplete; in some cases they are mistaken; very often the statements of the properties are over-constrained. In the following, we show that the constraints usually stated for the Heuristic Search in State Graphs, in order to ensure the completeness or the admissibility or the sub-admissibility, can be widely relaxed; so, we prove that these properties can be ensured when working with more general state graphs, or with more general evaluation functions, or with more general paths lengths or with more general algorithmic mechanisms; for instance, the arc costs and the heuristic estimates may be not exclusively positive; for instance yet, the heuristic estimate of any node may be a variable rather than a constant. The statements and the proofs of the theorems that we present hereunder supply a theoretical support for applying diverse Heuristic Search algorithms (previously known or not) in enlarged contexts. Among the works whose *formal* motivations have notably stimulated the present study, let us quote Vanderbrug (1976), Gelperin (1977), Bagchi and Mahanti (1983), Pearl (1984), Dechter and Pearl (1985), Korf (1988) and Russell (1992).

### *1.2. Plan*

In Section 2 we recall previous results about the completeness and the admissibility of various Heuristic Search algorithms; then we discern several interesting ways to relax their definitions and running conditions. In Section 3 we propose a formalization that covers many known codes and running contexts. In Section 4 we prove general theorems concerning the termination with discovery of a path from the start node to a goal node. In Section 5 we prove general theorems concerning the length of the discovered path.

## 2. Completeness, admissibility and sub-admissibility, previous results and ways for extensions

First we recall the definitions of Nilsson's A and A* algorithms (Hart, Nilsson, and Raphael, 1968; Nilsson, 1971, 1980). Then we report under which precise conditions it has been established that these algorithms terminate by discovering a solution possibly optimal.

```
1   procedure Heuristically-Ordered-Search-A-of-Nilsson :
2     open ← {s} ; closed ← {} ; g_memo(s) ← 0 ; f_memo(s) ← h(s)
3     until open = {} repeat
4       m ← best-first-extraction
5       if m ∈ T then edit-reverse-path stop endif
6       open ← open − {m} ; closed ← closed + {m}
7       for each n son of m repeat
8         g_neo ← g_memo(m) + c(m,n)
9         if n ∈ open and g_neo < g_memo(n) then update-father&g_memo&f_memo endif
10        if n ∈ closed and g_neo < g_memo(n) then  update-father&g_memo&f_memo
11                                                  open ← open + {n}
12                                                  closed ← closed − {n} endif
13        if n ∉ closed and n ∉ open then  update-father&g_memo&f_memo
14                                         open ← open + {n} endif
15      endrepeat
16    endrepeat
17  procedure edit-reverse-path :
18    until m = s repeat write m ; m ← father(m) endrepeat ; write "s"
19  procedure update-father&g_memo&f_memo :
20    father(n) ← m ; g_memo(n) ← g_neo ; f_memo(n) ← g_memo(n) + h(n)
21  procedure best-first-extraction :  return n* of open such as f_memo(n*) = min_{q ∈ open} f_memo(q)
```

*Figure 1.* Algorithm A of Nilsson.

### *2.1. Algorithms A*

The algorithm described in figure 1 is widely known as *algorithm A* (Nilsson, 1980)[3]. It searchs a goal node by progressively expliciting the state graph from the start node $s$. The state graph is supposed to be *son-finite*.[4] A *cost* $c(m, n)$ is associated to each arc $(m, n)$ which joins a node $m$ to a node $n$. The set of the goal nodes is denoted by $T$ (line 5); if a goal is discovered, the algorithm writes the reversed node list of a path from the start node to the found goal (lines 17, 18) and terminates (**stop** in line 5). Possibly, the algorithm terminates without discovering any goal node (test in line 3), even if such a node exists in the considered state graph. Possibly, it does not terminate. At the beginning of each execution of the loop 3–16, *open* is the set of nodes available for the next step of the search; the loop is constituted by a node *extraction* (lines 4 to 6) followed by the *expansion* of the extracted node (lines 7 to 15); *closed* is the set of nodes previously extracted and expanded that are not available now. The algorithm uses an *evaluation function* $f$ for choosing the node that must be extracted (from open) and then expanded. In the following, outside of the algorithm codes, we shall denote $f_x(n)$ the later value of the evaluation function at the node $n$ when the $x$th extraction is about to be executed. Nilsson and many authors write $f(n)$ rather than $f_x(n)$; but, in order to correct and to complete the previous theoretical results of Hart, Nilsson, and Raphael (1968, 1972), Gelperin (1977) has proposed to denote explicitely that the evaluation of any node varies when the algorithm is running. Here, index

$x$ (we shall say: *the rank x*) is aimed to recall that the same node $n$ may be successively evaluated in different ways, according to the successive extractions.

The algorithm extracts from *open* one of the nodes that own the minimal evaluation (lines 4 and 21); this mechanism may be interpreted like this: the more the evaluation of a node is small, the more we hope that this node leads a path to a goal. The sons of the extracted node are evaluated (for the first time or afresh); any new son is put in *open*; if a son is already in *closed* and if its present evaluation is less than its previous evaluation then the node is got out from *closed* and put in *open* again.

By definition of A (Hart, Nilsson, and Raphael, 1968; Nilsson, 1971, 1980), the $f_x(n)$ are calculated using the following formula: $f_x(n) = g_x(n) + h(n)$, where $g_x(n)$ is the *standard term* while $h(n)$ is the *heuristic term*.

The *standard term* $g_x(n)$ is recursively calculated as the shorter length of a path[5] from $s$ to $n$ known after the $x$th extraction. When $n$ is fixed, $g_x(n)$ decreases (not strictly) when $x$ increases, thus we may consider $g_x(n)$ as a decreasing overestimate of the minimal path length, if any, from $s$ to $n$; this minimum is denoted $g^*(n)$ below. The function $g$ depends on two variables: the rank $x$ and the node $n$; it is called the *standard function*. In figure 1, $g_{\text{memo}}(n)$ keeps track of the minimum of values $g_x(n)$ calculated until now for node $n$; likewise, $f_{\text{memo}}(n)$ memorizes the minimum of values $f_x(n)$ calculated until now for $n$; father($n$) records the father node whose expansion led to fix the later value $g_{\text{memo}}(n)$. So, any node in *open* is the end of a path from $s$, whose length is $g_{\text{memo}}(n)$ and that only includes expanded nodes; in addition, any path from $s$ to $n$ whose length is smaller has to include a node not yet expanded.

The *heuristic term* $h(n)$ only depends on node $n$ at which *heuristic function* $h$ is applied and not of the extraction rank. For this reason, we say that the heuristic function $h$ is *static* while we say that the standard function $g$ (and the evaluation function $f$) is *dynamic*.[6] Moreover, Nilsson's A algorithm (and most of its successors) supposes that the heuristic $h$ is positive. The measure (according to the length function; here: $\mathcal{L}_{\text{add}}$) of a minimal path, if any, from $n$ to the set $T$ of goal nodes, is denoted $h^*(n)$; so, $h^*(s)$ is the length of a minimal path, if any, from the start node to the set of goals. Commonly, $h(n)$ is considered as an estimate of $h^*(n)$.

### *2.2. Algorithms A**

For Nilsson (1980) and other authors, algorithms A are designated by A$^*$ when the state graph $G$ and the heuristic function $h$ satisfy the relation: for any node $n$ of $G$, $h(n) \leq h^*(n)$. We say that $h$ is a *lower-bounding* function.[7]

### *2.3. Completeness and admissibility for A and A**

An algorithm of Heuristic Search in state graphs is called *complete* if and only if it is guaranteed that it terminates finding a path from the start node ($s$) to the set of goal nodes when such a path exists. An algorithm is called *admissible* if and only if it is complete and the length of the found path is $h^*(s)$. The admissibility of the $A^*$'s is proved by Hart, Nilsson, and Raphael (1968), Nilsson (1971, 1980), and Pearl (1984) supposing that, at once: 1) the state graph $G$ is son-finite and it contains at least one goal, 2) $\exists \delta > 0, \forall (m, n)$

arc of $G$, $c(m, n) \geq \delta$ (this property characterizes the so-called $\delta$-*graphs*), 3) $h \geq 0$. Under the same conditions any algorithm A is complete.

### *2.4. Admissibility/sub-admissibility for other algorithms*

An algorithm is *sub-admissible* if it terminates finding a path from $s$ to $T$ whose length is *near* to $h^*(s)$. According to the relations that concretize the *near to* notion, the *sub-admissibility* has different meanings. Figure 2 reports previous results concerning the admissibility of HPA, B, C, IDA*, A** and the sub-admissibility of Harris' A, Pohl's A, $A^*_\varepsilon$, $A_\varepsilon$, B′, BF*, D; for a while, it is sufficient to consider columns 1 and 9.

### *2.5. Several interesting ways to relax A* and other algorithms*

Figure 2 suggests several ways to relax A* and various other algorithms, *while preserving diverse forms of sub-admissibility*.

***2.5.1. Constraints concerning the state graph and the length function.*** Consider columns 2 to 4. All the mentioned algorithms refer to $\delta$-*graphs* (possibly infinite) or finite graphs; note: in the first case, all the costs are *strictly* positive and in the both cases they are positive.[8] The length of a solution path is always calculated as the *sum* of the costs of its arcs.[9] We shall relax these constraints.

***2.5.2. Constraints concerning the heuristic function and the evaluation function.*** Consider columns 5 and 6. Most authors associate admissibility with heuristic functions that are at once positive, static and lower-bounding. Nevertheless, we remark that B′, A** and D algorithms can assure minimal solutions using non-static heuristic functions. We observe a form of sub-admissibility for Harris'A and for $A_\varepsilon$ while these algorithms use non-lower-bounding heuristic functions. We observe also a form of sub-admissibility for Pohl's extended A while this algorithm uses a non-static and non-lower-bounding heuristic function. Moreover, except for BF* and A**, the evaluation function $f$ is always calculated as the sum (or a linear combination, for HPA) of two functions: the standard function $g$ used by Nilsson's A and a heuristic function (the $h$ of algorithm A or a variant $h'$). From these examples, we shall propose a more general point of view because: 1) in some circumstances (Gelperin, 1977) the arc costs (thus the functions $f$, $g$, $h$) may be non-positive, 2) it may be natural to adjust, at least for some nodes $n$, the quantity $h(n)$ in proportion as the algorithm is running, 3) it may be easier to know (or more attractive to use) a non-lower-bounding heuristic function, and 4) it may be pertinent to examine the interest of other forms of $f$ than a linear combination of $g$ and $h$ or $h'$.

***2.5.3. Constraints concerning the extraction mode.*** Consider column 7. Most algorithms execute *best-first* extractions: as A* does, they systematically extract one of the nodes of *open* that presently minimize the evaluation function. The $A^*_\varepsilon$ and $A_\varepsilon$ relax this mechanism. We shall offer more possibilities; so, the choice of the nodes to expand may be controlled according to a secondary criterion, as previously suggested by Ghallab and Allard (1982) and Pearl and Kim (1982).

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|
| Name | Arc costs | Length function | State graphs | Heuristic function | Evaluation function | Extraction mode | Updating type | Length of the discovered path |
| A* | $>0$ | $\mathcal{L}_{add}$ | δ-graphs | h : static $0 \le h \le h^*$ | $g + h$ | best-first | 1&2&3 | $h^*(s)$ |
| HPA | $>0$ | $\mathcal{L}_{add}$ | δ-graphs | h : static $0 \le h \le h^*$ | $(1-\omega)g + \omega h$ $[0 \le \omega \le 1]$ | best-first | 1&2&3 | $h^*(s)$ [only if $\omega \le 1/2$] |
| extended A of Pohl | $\ge 0$ | $\mathcal{L}_{add}$ | finite graphs | dynamic and positive h′ derived from static h $0 \le h \le h^*$ | $g_x(n) + h'_x(n) = g_x(n) + h(n) + \lambda(1-\frac{depth_x(n)}{N})h(n)$ $(0 \le \lambda)$ | best-first | 3 | $\le (1+\lambda)h^*(s)$ |
| A of Harris | $>0$ | $\mathcal{L}_{add}$ | δ-graphs | h : static $0 \le h$ $h \le h^* + H$ $(H \ge 0)$ | $g + h$ | best-first | 1&2&3 | $\le h^*(s) + H$ |
| B | $>0$ | $\mathcal{L}_{add}$ | δ-graphs | h : static $0 \le h \le h^*$ | $g + h$ | best-first | 1&2&3 | $h^*(s)$ |
| $A^*_\varepsilon$ | $>0$ | $\mathcal{L}_{add}$ | δ-graphs | h : static $0 \le h \le h^*$ | $g + h$ | $n_x$ if $f_x(n_x) \le (1+\varepsilon)f_x(n_x)^*$ $(\varepsilon \ge 0)$ | 1&2&3 | $\le (1+\varepsilon)h^*(s)$ |
| $A_\varepsilon$ | $>0$ | $\mathcal{L}_{add}$ | without infinite path whose length is finite | h : static $0 \le h$ $h \le (1+\alpha)h^*$ $(\alpha \ge 0)$ | $g + h$ | $n_x$ if $f_x(n_x) \le (1+\varepsilon)f_x(n_x)^*$ $(\varepsilon \ge 0)$ | 1&2&3 | $\le (1+\varepsilon)(1+\alpha)h^*(s)$ |
| C | $>0$ | $\mathcal{L}_{add}$ | δ-graphs | h : static $0 \le h \le h^*$ | $g + h$ | best-first | 1&2&3 | $h^*(s)$ |
| B′ | $>0$ | $\mathcal{L}_{add}$ | δ-graphs | dynamic and positive h′ derived from static h $0 \le h \le h^*$ | $g + h'$ | best-first | 3 | $h^*(s)$ |
| BF* | $>0$ | $\mathcal{L}_{add}$ but ideas for extension | δ-graphs | dynamic and positive h′ derived from static h $0 \le h \le h^*$ | $f_X(n) = \Psi(\mathcal{L}(P_{X,n}))$ ($\Psi$ strictly increasing) | best-first | 1 | $\le \Psi^{-1}(M)$ (M = min for all paths $\mathcal{P}$ from s to T of $\max_{p \in \mathcal{P}} f(\mathcal{P}_p)$ |
| A** | $>0$ | $\mathcal{L}_{add}$ | δ-graphs | dynamic and positive h′ derived from static h $0 \le h \le h^*$ | $f_X(n) = \max_{p \in P_{X,n}}$ of $\{g_X(p)+h(p)\}$ | best-first | 3 | $h^*(s)$ |
| IDA* | $\ge 0$ | $\mathcal{L}_{add}$ | finite graphs | h : static $0 \le h \le h^*$ | $g + h$ | best-first | 1&2&3 | $h^*(s)$ |
| D | $>0$ | $\mathcal{L}_{add}$ | δ-graphs | dynamic and positive h′ derived from static and positive h | $g + h'$ | best-first | 3 | = minimal length of paths $\mathcal{P}$ from s to T minimizing $\max_{p \in \mathcal{P}}$ $\{\mathcal{L}_{add}(\mathcal{P}_p)+h(p)\}$ [Note: if $h \le h^*$, this quantity is $h^*(s)$] |

*Figure 2.* Admissibility/sub-admissibility of diverse Heuristic Search algorithms: a survey. $Depth_x(n)$ is the minimal number of arcs from $s$ to $n$, known at rank $x$; $N$ is an upperbound for all $x$ and $n$. $P_{x,n}$ is the pointer path of node $n$ at rank $x$ (see Section 3.8). The updating types are presented in Sections 3.1 and 3.3, and the extraction modes in Section 3.4. At rank $x$, $n_x$ is the extracted node and $f_x^*$ the min of $f_x(n)$ for all $n$ in open. $\mathcal{F}_p$ is the part of $\mathcal{F}$ from $s$ to $p$. Actually, some results reported in last column were originally established with more restrictive hypothesis not explicited here (it is the case for Harris'A, BF*, A**).

***2.5.4. Constraints concerning the updating type.*** Consider column 8. An important (and perhaps undervalued) difference between algorithms lies in the manner to update the values and the pointers associated to each node. The difference appears if we compare the algorithms that use static $h$ with those that use dynamic $h$: the latter can dissociate the lowerings of $g$ and $f$. The difference also appears when an algorithm (for instance: BF*) does not exploit the $g$ values: then the pointers are updated if and only if the $f$ values are lowered. We shall develop this topic in Sections 3.1 and 3.3.

Hereafter we present an unifying and generalizing point of view.

## 3. Formalizations and generalizations

We present in Section 3.1 a general Heuristically-Ordered Search algorithm, named $\wp$ *of type* 1; its code does not refer to arc or path lengths. We propose in Section 3.2 a general definition of the *length* of a path. In Section 3.3, we introduce algorithms $\wp$ *of type* 2 and $\wp$ *of type* 3 whose codes refer to this general length, in two different ways. Until now, we have not discussed the *extraction mode* of the algorithms $\wp$, whatever the type (1, 2 or 3); in Section 3.4 we propose a new extraction mode named $\mathcal{E}$*-extraction*. In Section 3.5, among algorithms $\wp$, we distinguish the sub-family $\tilde{A}$. Releasing the contraints relative to the evaluation functions or to their heuristic components (Section 3.6) and also the constraints relative to the state graphs (Section 3.7) we access to application contexts larger than usual. This formalization leads to prove general theorems concerning the completeness (Section 4) and the admissibility/sub-admissibility (Section 5).

### *3.1. Algorithms $\wp$ of type 1*

We call the code presented in figure 3: *algorithm* $\wp$ *of type* 1. The reference to the evaluation function $f$ is only done in the assignment instructions of the lines 2 and 9. Later, we shall make the algorithm more precise by discussing the *extraction* instruction in line 5 and we shall consider the application circumstances: what kind of evaluation functions $f$, what kind of state graphs, what kind of path lengths in these graphs?

According to the manner to make the line 5 more precise, we may define different *extraction modes* (studied in Section 3.4). The lines 8–15 realize the *expansion* of the extracted node $m$. Each extracted node that it is not a goal (test in line 6) is immediately *expanded*: *all* its sons are considered (*partial* expansions are not allowed in this paper).

According to the manner for governing the updating of $f_{\text{memo}}(n)$ and father($n$), we may define different *types* for algorithms $\wp$. In algorithm $\wp$ *of type* 1, the update orders comply with the following rules:

at the time of the expansion of any node m and for any son n of m,
1) $f_{memo}$(n) receives f(n,extraction-rank) except if the $f_{memo}$(n) value was already smaller and
2) father(n) receives m except if $f_{memo}$(n) is not lowered by the step 1.

```
1  procedure Heuristically-Ordered-Search-ϱ-type-1 :
2      open ← {s} ; closed ← {} ; extraction-rank ← 0 ; f_memo(s) ← f(s,extraction-rank)
3      until open = {} repeat
4      |   extraction-rank ← extraction-rank + 1
5      |   m ← extract-node-of-open-according-to-values-f_memo-for-the-nodes-in-open
6      |   if m ∈ T then edit-reverse-path ; stop endif
7      |   open ← open – {m} ; closed ← closed + {m}
8      |   for each n son of m repeat
9      |   |   f_neo← f(n,extraction-rank)
10     |   |   if n ∈ open and f_neo < f_memo(n) then update-father&f_memo endif
11     |   |   if n ∈ closed and f_neo< f_memo(n) then update-father&f_memo ;
12     |   |                                          open ← open + {n}
13     |   |                                          closed ← closed – {n} endif
14     |   |   if n ∉ closed and n ∉ open then update-father&f_memo ; open ← open + {n} endif
15     |   endrepeat
16     endrepeat
17 procedure edit-reverse-path : until m = s repeat write m ; m ← father(m) endrepeat ; write "s"
18 procedure update-father&f_memo : father(n) ← m ; f_memo(n) ← f_neo
```

*Figure 3.* Algorithm $\varrho$ of type 1. The extraction procedure called in line 5 is not yet precisely determined (see Section 3.4).

***3.1.1. Remarks about the generalization.*** Clearly, algorithms A and A* are particular cases of algorithms $\varrho$ of type 1. It may be easily verified that Pohl's HPA, Harris'A, Martelli's B, Pearl-Kim's $A^*_\varepsilon$, Ghallab-Allard's $A_\varepsilon$, Bagchi-Mahanti's C, Pearl's BF*, Korf's IDA* and Köll-Kaindl's SDW are also $\varrho$ of type 1.

Algorithms $\varrho$ of type 1 do not refer to any cost of arc or path: they may be applied to graphs whose arcs are not valued. We shall further propose formulas of sub-admissibility for some algorithms $\varrho$ of type 1 that exploit particular evaluation functions.

We shall also present other algorithms called $\varrho$ of type 2 and $\varrho$ of type 3, whose codes explicitly refer to costs of arcs and paths. Before, we propose a broadened definition for the *length* of paths.

### *3.2. Generalization of the notion of path length*

Commonly the length of a path is calculated as the sum of the costs of its arcs (this length is denoted by $\mathcal{L}_{\text{add}}$; see column 3 of figure 2). However, (Pearl, 1984) looks at other path lengths, especially the maximal cost of the arcs defining the path: (Yager, 1986; Dubois, Lang, and Prade, 1987) calculate the path length as the minimum of the arc costs; (Gonella, 1989) aggregates the arc costs by multiplication. Hereafter we propose a general definition for the lengths of paths.

In essence, the operation $+$ and the set $\mathbb{R} = ]-\infty, +\infty[$, involved in the definition of the classical length $\mathcal{L}_{\text{add}}$, can be respectively replaced by any two place operation $\Theta$ and any subset $\mathbb{V}$ of $\mathbb{R}$, provided that $\mathbb{V}$ and $\Theta$ forms a monoid.[10] We denote $c$ a function

which associates to each arc $u$ a *cost* $c(u)$ in $\mathbb{V}$. We call *length associated to the monoid* $(\mathbb{V},\Theta)$ *and to the function* $c$, the function $\mathcal{L}$ that respects the following rules: 1) for any arc $u : \mathcal{L}(u) = c(u)$, 2) $\mathcal{L}$ (empty sequence of arcs) $= e_n$, identity element of $(\mathbb{V},\Theta)$, 3) for any sequences of arcs $\mathcal{S}'$ and $\mathcal{S}''$, $\mathcal{L}$ (concatenation of $\mathcal{S}'$ and $\mathcal{S}''$) $= \Theta(\mathcal{L}(\mathcal{S}'), \mathcal{L}(\mathcal{S}''))$.

***3.2.1. Remarks about the generalization.*** Taking $\mathbb{V} = \mathbb{R}$ or $\mathbb{R}^+$ or $\mathbb{Q}$ (rational numbers) or $\mathbb{Q}^+$ or $\mathbb{Z}$ (relative numbers), or $\mathbb{N}$ (integer numbers), and $\Theta = +$, we recognize the current forms of the length $\mathcal{L}_{\text{add}}$. With some subsets $\mathbb{V}$ of $\mathbb{R}$, we may choose $\Theta(x, y) = x \cdot y$ or $\min(x, y)$ or $\max(x, y)$ or $\sqrt{x^2 + y^2}$, etc. (for other examples: (Farreny, 1995)). We shall prove in Sections 4 and 5 that our generalization is compatible with interesting properties related to the completeness and the admissibility.

Now, we can define, for any node $n$, a *generalized standard term* $g_x(n)$ substituting operation $\Theta$ to $+$ in the previous definition: $g_x(n)$ will be the length $\mathcal{L}(\mathcal{C})$ of the shorter path $\mathcal{C}$ from $s$ to $n$ known after the $x$th extraction; if $n$ is fixed, $g_x(n)$ decreases in the wide sense when $x$ increases. The minimal path length from $s$ to $n$, if it exists, is still denoted $g^*(n)$; the minimal path length from $n$ to $T$, if it exists, is also denoted $h^*(n)$.

### *3.3. Other types of updating—algorithms ℘ of type 2 or 3*

Now, we distinguish two types of updating mechanisms that depend on the generalized standard term $g_x$. We shall speak of algorithms ℘ *of type* 2 if and only if:

at the time of the expansion of any node m and for any son n of m,
1) $f_{\text{memo}}$(n) receives f(n,extraction-rank) except if the $f_{\text{memo}}$(n) value was already smaller and
2) father(n) receives m except if the value of $g_{\text{memo}}$(n) is not lowered when m is expanded.

We shall speak of algorithms ℘ *of type* 3 if and only if:

at the time of the expansion of any node m and for any son n of m,
1) $f_{\text{memo}}$(n) receives f(n,extraction-rank)
except if the value of $g_{\text{memo}}$(n) is not lowered when m is expanded and
2) father(n) receives m except if the value of $g_{\text{memo}}$(n) is not lowered when m is expanded.

***3.3.1. Remarks about the generalization.*** Column 8 of figure 2 indicates the updating type of the particular algorithms to which we refer from the beginning. Nilsson's A and A* algorithms are simultaneously of types 1, 2 and 3, because for any node $n$, the staticity of the heuristic term (function $h$) leads $f_{\text{memo}}(n)$ and $g_{\text{memo}}(n)$ to be lowered at the same time. For the same reason, it may be easily verified that Pohl's HPA, Harris'A, Martelli's B, Pearl-Kim's $A^*_\varepsilon$, Ghallab-Allard's $A_\varepsilon$ , Bagchi-Mahanti's C, Korf's IDA* and Köll-Kaindl's SDW are simultaneously of types 1, 2 and 3. Pearl's BF* is a particular algorithm ℘ of type 1.

Pohl's extended A, Mero's B′, Mahanti-Ray's D and Dechter-Pearl's A** are particular $\varrho$ of type 3. We have not met, in the literature of the field, any particular algorithm that is purely of type 2. However, when the updating type is 2, the interesting property of *homogeneity* (defined just below) is satisfied. So, one may suppose that particular algorithms of type 2 will be proposed sooner or later.

***3.3.2. Property of homogeneity.*** For any algorithm $\varrho$, for any rank of extraction $x$ and for any appeared node $n$, we call *pointer path of node n at rank x* the path from $s$ to $n$ determined by reversing the sequence $n$, father($n$), father(father($n$)), ..., $s$. We denote it $C_{x,n}$. For any updating type, we have: $\mathcal{L}(C_{x,n}) \geq g_x(n)$. For algorithms $\varrho$ of type 2 or 3, it may be easily verified that: $\mathcal{L}(C_{x,n}) = g_x(n)$. We shall say that the *property of homogeneity* is satisfied when for any $x$ rank of extraction, for any $n$ node already appeared *and situated on a path from s to T*, $\mathcal{L}(C_{x,n}) = g_x(n)$. Clearly: algorithms $\varrho$ of type 2 or 3 satisfy the property of homogeneity. *A priori* that is not the case for $\varrho$'s of type 1; nevertheless, it may be verified that the specific constraints considered by Pearl imply that algorithm BF* satisfy the property of homogeneity. The homogeneity property will be exploited for proving a theorem (Section 5.6) which provides a general relation concerning the lengths of the solution paths.

### *3.4. Extraction modes—algorithms $\varrho_\varepsilon$*

Line 4 in figure 1 (Nilsson's A) orders a *best-first extraction* that is described in line 21. Many other Heuristic Search algorithms use the same extraction mode. The best-first extraction has an intuitive motivation when the algorithms use the evaluations given by $g + h$ in order to guide the search; indeed, for any node $n$, $g_x(n)$ may be interpreted as an estimate of $g^*(n)$ and $h(n)$ may be interpreted as an estimate of $h^*(n)$; so $g_x(n) + h(n)$ may be interpreted as an estimate of $g^*(n) + h^*(n)$, that is to say: the length of a shortest path from $s$ to $T$ passing by $n$; finally, the minimal value of $g_x(n) + h(n)$, for any $n$ in *open*, may be interpreted as an estimate of the length of a shortest path from $s$ to $T$; which leads to respect the *best-first constraint*: extract $\tilde{n}$ of *open* such as $f_{\text{memo}}(\tilde{n}) = \min_{q \in \text{open}} f_{\text{memo}}(q)$. However, according to the application context, it may seem not pertinent to choose $\tilde{n}$ only by virtue of this formula; (Ghallab, 1982; Ghallab and Allard, 1982, 1983; Pearl and Kim, 1982; Pearl, 1984) have proposed to relax the *best-first* constraint: given a positive number $\varepsilon$, $\tilde{n}$ may be extracted seeing that $f_{\text{memo}}(\tilde{n}) \leq (1 + \varepsilon) \cdot \min_{q \in \text{open}} f_{\text{memo}}(q)$; this extraction mode allows to use a secondary criterion, depending on the application, in order to complete the choice of $\tilde{n}$. Hereafter we propose a generalization of the previous extraction modes. We shall see that this generalization is also compatible with interesting properties concerning the lengths of the solution paths. Beforehand we have to introduce the notion of *overpassed function*.

***3.4.1. Overpassed functions.*** Let a function $f \mid x \in D \subset \mathbb{R} \rightarrow f(x) \in \mathbb{R}$. We shall say that *f is overpassed on D* when: $\forall x \in D, \exists y \in D,\ f(x) \leq y$. Likewise, we shall say that *a set $S_1$ is overpassed by a set $S_2$* when: $\forall x \in S_1, \exists y \in S_2, x \leq y$.

***3.4.2. $\mathcal{E}$-extraction.*** Consider a particular application of an algorithm $\varrho$. Suppose that there exists a function $\mathcal{E}$ widely increasing and overpassed on $\mathbb{W}$, such as during this application, for any rank of extraction, $\min_{q \in \text{open}} f_{\text{memo}}(q) \leq \mathcal{E}(\min_{q \in \text{open}} f_{\text{memo}}(q))$. This condition may be satisfied, for instance, if $\forall x \in \mathbb{W}, x \leq \mathcal{E}(x)$. Suppose that the algorithm extracts always an $\tilde{n}$ of *open* such as $f_{\text{memo}}(\tilde{n}) \leq \mathcal{E}(\min_{q \in \text{open}} f_{\text{memo}}(q))$. We shall say that the algorithm $\varrho$ uses *$\mathcal{E}$-extraction* and it will be then denoted by $\varrho_{\mathcal{E}}$. We may distinguish algorithms $\varrho_{\mathcal{E}}$ whose updating type is 1, 2 or 3.

*Remarks about the generalization:* The best-first extraction is a particular case of $\mathcal{E}$-extraction: take $\mathcal{E}$ equal to the identity function $\mathcal{J}$. Many known algorithms (see column 7 of figure 2) use the best-first extraction too: HPA, A of Harris, extended A of Pohl, SDW, BF* (note: in BF*, BF means Best-First), B′, D and $A^{**} \cdot A^*_\varepsilon$ and $A_\varepsilon$ use another simple form of $\mathcal{E}$-extraction:[11] take $\mathcal{E} = (1 + \varepsilon) \cdot \mathcal{J}$. It may be verified that algorithms B, C, IDA* use other particular (and less evident, see justification in Farreny (1997c)) forms of $\mathcal{E}$-extraction.

### *3.5. Algorithms $\tilde{A}$*

In order to define this sub-family of algorithms $\varrho$ we suppose that a length $\mathcal{L}$ is available; thus there exists a monoid $(\mathbb{V}, \Theta)$ such that $\Theta$ is the operation used by $\mathcal{L}$ and any arc of the state graphs is valued by a number taken in $\mathbb{V}$. An algorithm $\tilde{A}$ is a $\varrho$ whose evaluation function satisfies the following constraint: for any rank of extraction $x$, for any son $n$ of the node extracted at rank $x$, $f_x(n) = \Theta(g_x(n), h_x(n))$, where $g_x(n)$ is the standard term defined in Section 2.1, while $h_x(n)$ is a value of $\mathbb{V}$ (called *heuristic term*).

For any $x$ and any $n$, $h_x(n)$ may always be *interpreted* (well knowing or not, according to the context), as *an estimate* of the length of a minimal path, if any, from $n$ to $T$. Note: if the monoid $(\mathbb{V}, \Theta)$ is a group, any $\varrho$ may be seen as an algorithm $\tilde{A}$: it is sufficient to take $h_x(n) = \Theta(g'_x(n), f_x(n))$ where $g'_x(n)$ is the inverse of $g'_x(n)$ in the monoid $(\mathbb{V}, \Theta)$; if $(\mathbb{V}, \Theta)$ is not a group we may define some $\varrho$'s that are not algorithms $\tilde{A}$: then $\tilde{A}$'s form *a proper* subset of the set made by the $\varrho$'s.

*Remarks about the generalization.* Nilsson's A is a particular $\tilde{A}$ that uses a static heuristic term.[12] It may be easily verified that the following algorithms are particular $\tilde{A}$ too: extended A of Pohl, B, C, $A^*_\varepsilon$, $A_\varepsilon$, B′, D, SDW. At first sight, the evaluation functions of algorithms BF* and A** have not the form required for $\tilde{A}$; but BF* and A** are defined on the monoid $(\mathbb{R}^+, +)$ and use evaluation functions $f$ such as: $f \geq g$; thus, the evaluation function $f$ may be formulated as a sum $g + h$ (with $h \geq 0$) and so we can conclude that BF* and A** are particular algorithms $\tilde{A}$. Algorithm HPA behaves as an $\tilde{A}$, because its evaluation function is *proportional* to the evaluation function of an $\tilde{A}$. It may be verified that algorithm IDA* behaves as an $\tilde{A}$ *during each iteration*.

Clearly, we may distinguish algorithms $\tilde{A}$ whose updating type is 1, 2 or 3.

### *3.6. Worth-considering constraints for h or f*

We propose to relax the constraints commonly required for the evaluation function $f$ or for the heuristic function $h$. In this less constrained context we shall prove general results

concerning the completeness and the admissibility or sub-admissibility; so, it will be possible to use a larger variety of functions $f$ or $h$ and thus the field of applications will be potentially extended. Still we denote by $\mathbb{V}$ the domain of the arc values (a length $\mathcal{L}$ is defined on $\mathbb{V}$, the range of $h$ is included in $\mathbb{V}$), by $\mathbb{W}$ the range of $f$ and by $G$ the state graph dealt with. Given a Heuristic Search algorithm **a** and a state graph $G$, we denote by $S(\mathbf{a}, G)$ the set of the nodes $n$ such that: 1) $h^*(n)$ is defined and 2) $n$ is evaluated during the application of **a** to $G$.

***3.6.1. Covered h or f.*** We say that the heuristic function $h$ of an algorithm $\tilde{A}$ *is covered by F when the algorithm is applied to G*, if and only if: 1) there exists a function $F$ whose domain is $S(\tilde{A}, G)$, whose range is $\mathbb{V}$ and 2) for any node $n$ of $S(\tilde{A}, G)$, if $n$ is evaluated at rank $x$ then $h_x(n) \leq F(n)$.

*Remarks about the generalization*: The preceding constraint widely extends the basic constraint respected by Nilsson's $A^*$ and many successors: $h(n) \leq h^*(n)$ (in this particular case we said that the *static* heuristic term was *lower-bounding* or *admissible*, see Section 2.2).

We can immediately propose several general examples of covered heuristic functions: 1) the $h$'s that are upper-bounded (i.e., $\exists M \in \mathbb{V}$, for any node $n$ evaluated at any rank $x$: $h_x(n) \leq M$), 2) the $h$'s that are *semi-static* (i.e., for any evaluated node $n$, $h_x(n)$ is allowed to take only a finite number of distinct values), 3) the $h$'s that are simply *static* (i.e., for any evaluated node $n$, for any $i$ and $j$ ranks for which $n$ is evaluated: $h_i(n) = h_j(n)$), 4) the $h$'s that satisfy relations such as: for any node $n$ evaluated at any rank $x$, $h_x(n) \leq (1+\alpha) \cdot h^*(n) + \mathrm{H}$, where the constants $\alpha$ and H are positive or zero (but note: $h$ is not necessarily static).

Now, algorithms A, HPA, B, C, $A^*_\varepsilon$, $A_\varepsilon$, IDA$^*$ and SDW use static (thus covered) heuristic functions (see column 5 in figure 2); moreover, $A_\varepsilon$ satisfy a relation of the form $h \leq (1+\alpha) \cdot h^*$, while Harris'A satisfies a relation of the form $h \leq H$. It may be verified that the extended A of Pohl, which has introduced the *dynamic weighting*, uses a semi static (thus covered) $h$ and that B′ and D use other particular forms of covered $h$'s.

We say that the evaluation function $f$ of a $\wp$ is *covered by F when the algorithm is applied to G*, if and only if: 1) there exists a function $F$ whose domain is $S(\wp, G)$, whose range is $\mathbb{V}$ and 2) for any node $n$ of $S(\wp, G)$, if $n$ is evaluated at rank $x$ then $f_x(n) \leq F(n)$.

*Remark about the generalization*: It may be verified that $A^{**}$ use a covered $f$.

We denote $\tilde{A}^F$ an algorithm $\tilde{A}$ that exploits a covered $h$, $\tilde{A}^F_{\mathcal{E}}$ an $\tilde{A}^F$ that uses $\mathcal{E}$-extraction, $\wp^F$ an algorithm $\wp$ that exploits a covered $f$, $\wp^F_{\mathcal{E}}$ a $\wp^F$ that use $\mathcal{E}$-extraction.

***3.6.2. Finitely-decreasing h or f.*** Given a node $n$, we denote $[h_x(n)]_x$ the sequence of values taken by the heuristic function $h$ for the successive ranks of extraction $x$ for which $n$ is evaluated. We say that the heuristic function $h$ is *finitely-decreasing when an algorithm* $\tilde{A}$ *is applied to G*, if and only if: for any evaluated node $n$ of $G$, there does not exist any infinite sub-sequence of $[h_x(n)]_x$ that is strictly decreasing. Likewise, we say that the evaluation function $f$ is *finitely-decreasing when an algorithm* $\wp$ *is applied to G*, if and only if: for this application, for any evaluated node $n$ of $G$, there does not exist any infinite sub-sequence of $[f_x(n)]_x$ that is strictly decreasing.

*Remarks about the generalization*: The *finite-decrease* of $h$ or $f$ is a rather soft constraint. Indeed, it seems often natural that for any node $n$ the estimates $h_x(n)$ cannot decrease

indefinitely. Obviously, all the semi-static functions and therefore all the static functions are finitely decreasing. It may be easily verified that algorithms A, HPA, extended A of Pohl, B, $A^*_\varepsilon$, $A_\varepsilon$, C, B′, IDA*, D and SDW use finitely decreasing (most of them use a static or semi static $h$; see Column 5 in figure 2).

***3.6.3. Quasi-coincident h.*** We say that heuristic function $h$ is *quasi-coincident when an algorithm $\tilde{A}$ is applied to G*, if and only if: $\exists m_T \in \mathbb{V}, \exists m'_T \in \mathbb{V}, \Theta(m_T, m'_T) \geq e_n$ ($e_n$: identity element), for any evaluated goal node $t$, $h_x(t) \geq m_T$.

*Remarks about the generalization*: It may be easily verified that A, HPA, extended A of Pohl, B, $A^*_\varepsilon$, $A_\varepsilon$, C, BF*, B′, IDA*, D, A**, SDW are quasi-coincident;[13] indeed, all these algorithms use $\mathbb{V} = \mathbb{R}^+$, $\Theta = +(e_n = 0)$ and heuristic function $h$ (implicit in the case of BF* and A**) that is positive or zero.

### *3.7. A larger family of state graphs*

We identify now a wide family of state graphs: the *$\mathcal{L}$-standard* state graphs. It includes, at least, all the state graphs considered in the literature related to A, HPA, extended A of Pohl, B, $A^*_\varepsilon$, $A_\varepsilon$, C, BF*, B′, IDA*, D, A** and SDW.

***3.7.1. $\mathcal{L}$-uncompressibility.*** Let $\mathcal{L}$ be the used length.[14] We say that a state graph $G$ is *$\mathcal{L}$-uncompressible* if and only if: $\forall M \in \mathbb{V}, \exists k \in \mathbb{N}, \forall \mathcal{C}$ elementary[15] path of $G$ from $s, \mathcal{N}(\mathcal{C}) > k \Rightarrow \mathcal{L}(\mathcal{C}) > M$, where $\mathcal{N}(\mathcal{C})$ denotes the number of arcs of $\mathcal{C}$. That is to say: for any $M$ of $\mathbb{V}$, beyond some number of arcs, the length $\mathcal{L}(\mathcal{C})$ of any elementary path $\mathcal{C}$ issued from $s$ is greater than $M$.

*Remarks about the generalization*: Clearly: any $\delta$-graph (see Section 2.3) is $\mathcal{L}_{\text{add}}$-uncompressible and any finite graph is $\mathcal{L}$-uncompressible (whatever is $\mathcal{L}$). All the algorithms above-mentioned, except $A_\varepsilon$, have been analyzed by their authors in case of application to $\delta$-graphs (see Column 4 in figure 2); Ghallab and Allard have defined and studied $A_\varepsilon$ when it is applied to graphs less constrained than $\delta$-graphs, but these graphs are still particular $\mathcal{L}_{\text{add}}$-uncompressible graphs.

***3.7.2. $\mathcal{L}$-standard state graphs.*** We say that a state graph is $\mathcal{L}$-standard if and only if it satisfies the four following conditions: it is son-finite, it contains at least a goal, it is $\mathcal{L}$-uncompressible, it does not contain any absorbant circuit.[16] Note: it may be easily verified that any $\mathcal{L}$-standard graph owns at least one minimal path (in the sense of $\mathcal{L}$) from $s$ to $T$.

*Remarks about the generalization*: All the above-mentioned algorithms have been analyzed by their authors in case of application to particular $\mathcal{L}$-standard graphs.

Within the general framework that we have proposed, it is possible now to establish various general results related to the completeness (Section 4 below) and to the admissibility or sub-admissibility (Section 5).

## 4. General results about the completeness

Remember that $\mathbb{W}$ designates the range of the evaluation function and $\mathbb{V}$ the set of the arc values. Here are other notations that will be used for expressing some properties of the algorithms.

For a particular application of the algorithm, $extract_i$ designates the $i$th node extracted from *open*; always: $\text{extract}_1 = s$; in figure 3, $\text{extract}_i$ is identified by the programming variable *m* (lines 5 to 8); $open_i$ designates the set *open* at the time to decide the $i$th extraction.

$\underline{f}_i(n)$ designates the value of the evaluation function *f already assigned to the node n, available when the i*th *extraction* (or *i*th expansion) *begins*, i.e., at the time when line 5 is interpreted in figure 3; $\underline{f}_i(n)$ is identified by the programming variable $f_{\text{memo}}(n)$, that does not explicit the extraction rank $i$. But $f_i(n)$ designates the value of the evaluation function $f$ which is *calculated for the node n when the i*th *extraction is realized*; in figure 3, the value $f_i(n)$ is identified by the programming variable $f(n$, extraction-rank) (see line 9).

### *4.1. Applications with ceiling, circumscribed applications, flattened applications*

We say that the application of an algorithm $\varrho$ to a state graph $G$ admits a *ceiling* if and only if: $\exists M \in \mathbb{W}, \forall i$ rank of extraction, $\underline{f}_i(\text{extract}_i) \leq M$. We say that it is *circumscribed* if and only if: $\forall M \in \mathbb{W}, \exists k \in \mathbb{N}, \forall i$ rank of extraction, $\underline{f}_i(\text{extract}_i) \leq M \Rightarrow \mathcal{N}^*(s, \text{extract}_i) \leq k$, where $\mathcal{N}^*(s, \text{extract}_i)$ is the minimum of the number of arcs of the paths from $s$ to $\text{extract}_i$ in $G$. We say that it is *flattened* if and only if: $\forall n$ appeared node of $G, \exists i \in \mathbb{N}, \forall j$ rank of extraction, $j > i \Rightarrow \underline{f}_j(n) \geq \underline{f}_i(n)$.

### *4.2. Input nodes*

Let $G$ be a state graph that contains at least one goal and at which an $\tilde{\text{A}}$ is applied. It is easily verified that: at the time of any extraction $i$, on any path of $G$ from $s$ to $T$, there exists a node belonging to $open_i$ whose predecessors along the path belong to $closed_i$; this node is called: *input of the path for the considered extraction*.

### *4.3. Lemma*

*Any algorithm $\varrho$ applied to a state graph that contains at least one goal only can terminate extracting a goal node.*

**Proof:** Let $\mathcal{C}$ be a path from $s$ to $T$. $\forall i$ rank of extraction, $\exists n \in \text{open}_i$: we may consider the input of $\mathcal{C}$ at rank $i$. Thus, the algorithm can terminate only executing line 5 of the code, that is to say: extracting a goal node. □

### *4.4. General completeness theorem for $\varrho$'s of type 1, 2 or 3*

*Let an algorithm $\varrho$ of type* 1, 2 *or* 3 *that is applied to a $\mathcal{L}$-standard state graph. If the algorithm is of type* 1 *or* 2, *in order that the algorithm terminates extracting a goal node, it is necessary and sufficient that the application 1) admits a ceiling and 2) is circumscribed and 3) is flattened; if the algorithm is of type* 3 *it is necessary and sufficient that the application 1) admits a ceiling and 2) is circumscribed.*

**Proof:** a) Let us prove that conditions 1, 2, 3, are necessary for any type of algorithm $\varrho$.

Let $I$ be the last rank of extraction before the termination. $\forall n$ appeared node, $\exists i = I, \forall j$ rank of extraction, $j > i \Rightarrow \underline{f}_j(n) = \underline{f}_i(n)$; thus the application is flattened.

$\exists M = \max_{i \leq I} \underline{f}_i(\text{extract}_i)$, $M \in \mathbb{W}$ (because $\underline{f}_i(\text{extract}_i) \in \mathbb{W}$), $\forall i$ rank of extraction, $\underline{f}_i(\text{extract}_i) \leq M$; thus the application admits a ceiling.

$\forall M' \in \mathbb{W}, \exists k = \max_{i \leq I} \mathcal{N}^*(s, \text{extract}_i)$, $k \in \mathbb{N}$, $\forall i$ rank of extraction, $\underline{f}_i(\text{extract}_i) \leq M' \Rightarrow \mathcal{N}^*(s, \text{extract}_i) \leq k$; thus the application is circumscribed.

b) Let us prove that conditions 1, 2, 3 are sufficient when the algorithm $\varrho$ is of type 1 or 2, while the conditions 1, 2 are sufficient when the algorithm $\varrho$ is of type 3.

Because the application is circumscribed: $\forall M \in \mathbb{W}, \exists k \in \mathbb{N}, \forall i$ rank of extraction, $\underline{f}_i(\text{extract}_i) \leq M \Rightarrow \mathcal{N}^*(s, \text{extract}_i) \leq k$. Because of the application admits a ceiling: $\exists M \in \mathbb{W}, \forall i$ rank of extraction $\underline{f}_i(\text{extract}_i) \leq M$. Thus: $\exists k \in \mathbb{N}, \forall i$ rank of extraction, $\text{extract}_i \in \mathcal{E} = \{n \mid \mathcal{N}^*(s, n) \leq k\}$.

Because the graph is son-finite, it may be easily verified that $\mathcal{E}$ is finite. Thus, in order that the algorithm does not terminate, it is necessary that a node $n_0$ of $\mathcal{E}$ may be indefinitely extracted of open (line 4 in the code of the algorithm, figure 4). Now:

if the algorithm is of type 1 or 2, $\underline{f}_x(n_0)$ strictly decreases each time that $n_0$ is extracted of open; because the application is flattened, it is impossible to obtain an infinite and strictly decreasing sequence of values $\underline{f}_x(n_0)$;

if the algorithm is of type 3, $g_x(n_0)$ strictly decreases each time that $n_0$ is extracted of open; because the application is son-finite, $\mathcal{L}$-uncompressible and without absorbant circuit, it may be easily verified that $g_x(n_0)$ only takes a finite number of distinct values when $x$ runs; thus it is impossible to obtain an infinite and strictly decreasing sequence of values $g_x(n_0)$.

Thus the algorithm terminates. According to Lemma 4.3: extracting a goal node. □

### *4.5. Infra strictly increasing functions*

We shall say that a function $f$ defined from $D \subset \mathbb{R}$ to $\mathbb{R}$ is *infra strictly increasing* on $D$ if and only if $\forall x \in D, \exists x' \in D, \forall y \in D, y > x' \Rightarrow f(y) > f(x)$.

### *4.6. Overpassing functions*

Let a function $f \mid x \in D \subset \mathbb{R} \rightarrow f(x) \in \mathbb{R}$. We shall say that $f$ *is overpassing on* $D$ if and only if: $\forall x \in D, \exists y \in D, x \leq f(y)$.

### *4.7. Sufficient conditions of completeness for $\varrho_{\mathcal{E}}^{F}$'s of type 1, 2 or 3*

*Let an algorithm $\varrho_{\mathcal{E}}^{F}$ of type* 1, 2, *or* 3, *that is applied to a $\mathcal{L}$-standard state graph $G$. Suppose that $\mathbb{W}$ is overpassed by $\mathbb{V}$ (see Section 3.4). Suppose that there exists $\varphi$ function that is overpassing and infra strictly increasing on $\mathbb{V}$. Suppose that for any n evaluated node of $G$: $f_x(n) \geq \varphi(g^*(n))$. If the algorithm is of type* 1 *or* 2 *and if $f$ is finitely decreasing for the considered application to $G$, then the algorithm terminates extracting a goal node*[17]*; if the algorithm is of type* 3, *without other condition, then it terminates extracting a goal node.*

**Proof:** a) Because $G$ is son-finite, $\mathcal{L}$-uncompressible and without absorbant circuits, it may be easily verified that $g^*(n)$ is defined for any node $n$ of $G$, that is to say: there exists

a minimal path from $s$ to $n$ and, even: there exists an *elementary* minimal path from $s$ to $n$; let us call it $\mathcal{C}_n$: $g^*(n) = \mathcal{L}(\mathcal{C}_n)$; by hypothesis, for any evaluated node $n$ we have: $f_x(n) \geq \varphi(\mathcal{L}(\mathcal{C}_n))$.

b) Let us prove that: if $f$ is finitely decreasing for the considered application then this application is flattened. Else: $\exists n$ appeared node of $G, \forall i \in \mathbb{N}, \exists k$ rank of extraction for which $n$ est evaluated again, with: $k > i$ and $f_k(n) < \underline{f}_i(n)$. Thus, for $i = 1, \exists k1$ rank of extraction for which $n$ is evaluated again, with: $k1 > 1$ et $f_{k1}(n) < \underline{f}_1(n)$; for $i = k1, \exists k2$ rank of extraction for which $n$ is evaluated again, with: $k2 > k1$ and $f_{k2}(n) < \underline{f}_{k1}(n)$; and so on; the algorithm will calculate an *infinite sequence* of successive evaluation values $f_k(n)$: $f_1(n),\ f_{k1}(n),\ f_{k2}(n), \ldots$ (with: $1 < k1 < k2 < \ldots$) that are strictly decreasing thus *distinct*. This is contradictory with the definition of an evaluation function that is finitely decreasing for the considered application.

c) Because $G$ contains at least one goal, $\exists \mathcal{C}$ path from $s$ to $T$. Because $G$ is $\mathcal{L}$-standard, it may be easily verified that: $\forall n \in \mathcal{C}, h^*(n)$ is defined. Because we consider an algorithm $\wp^F$, $f$ is locally upper-bounded on $\mathbb{W}$ along $\mathcal{C}$, for the considered application, thus: $\exists M' \in \mathbb{W}, \forall n$ evaluated node of $\mathcal{C}$, $f_x(n) \leq M'$. By definition, extraction function $\mathcal{E}$ is overpassed on $\mathbb{W}$, thus: $\exists M \in \mathbb{W}, \mathcal{E}(M') \leq M$. Because $G$ contains at least one goal, according to Lemma 4.3: $\forall i$ *rank of extraction*, $\exists e_i \in \mathcal{C} \cap \text{open}_i$.

Now, by definition of $\mathcal{E}$: $\underline{f}_i(\text{extract}_i) \leq \mathcal{E}(\min_{q \in \text{open}} \underline{f}_i(q))$.

Because $\mathcal{E}$ is increasing: $\underline{f}_i(\text{extract}_i) \leq \mathcal{E}(\underline{f}_i(e_i))$. Let $j$ be the rank of extraction, $j < i$, for which value $\underline{f}_i(e_i)$ has been assigned to $e_i$: $f_j(e_i) = \underline{f}_i(e_i)$. Because $e_i \in \mathcal{C}$, $f_j(e_i) \leq M'$ thus $\underline{f}_i(e_i) \leq M'$. Because $\mathcal{E}$ is increasing: $\underline{f}_i(\text{extract}_i) \leq \mathcal{E}(M')$, thus: $\underline{f}_i(\text{extract}_i) \leq M$. Recapitulate: $\exists M \in \mathbb{W}, \forall i$ rank of extraction, $\underline{f}_i(\text{extract}_i) \leq M$, thus: the application admits a ceiling.

d) Let $\mathbb{V}_1 = \mathbb{W} \cap \mathbb{V}$ and $\mathbb{V}_2 = \mathbb{W} - \mathbb{V} \cdot \mathbb{W}$ overpassed by $\mathbb{V} \Leftrightarrow \forall x \in \mathbb{V}_2, \exists y \in \mathbb{V}, x < y$. Because $\varphi$ is overpassing on $\mathbb{V}$ : $\forall M \in \mathbb{V} : \exists x \in \mathbb{V}$ such as $\varphi(x) \geq M$. Because $\varphi$ is infra strictly increasing on $\mathbb{V} : \exists x' \in \mathbb{V}, x' \geq x, \forall y \in \mathbb{V}, y > x' \Rightarrow \varphi(y) > \varphi(x) \geq M$. Because $G$ is $\mathcal{L}$-uncompressible: $\forall M' \in \mathbb{V}$, especially for $M' = x', \exists k \in \mathbb{N}, \forall \mathcal{C}'$ elementary path issued from $s$: $\mathcal{N}(\mathcal{C}') > k \Rightarrow \mathcal{L}(\mathcal{C}') > x'$. By hypothesis, for any evaluated $n$ and for any rank $x$, $f_x(n) \geq \varphi(\mathcal{L}(\mathcal{C}_n))$, thus: $\forall i$ *rank of extraction*, $\underline{f}_i(\textit{extract}_i) \leq M \Rightarrow \varphi(\mathcal{L}(\mathcal{C}_n)) \leq M$; in this case, necessarily: $\mathcal{L}(\mathcal{C}_n) \leq x'$, else, because $\mathcal{L}(\mathcal{C}_n) \in \mathbb{V}$ we shall have: $\mathcal{L}(\mathcal{C}_n) > x' \Rightarrow \varphi(\mathcal{L}(\mathcal{C}_n)) > M$. Let us take: $\mathcal{C}' = \mathcal{C}_n$. $\mathcal{L}(\mathcal{C}_n) \leq x' \Rightarrow \mathcal{N}(\mathcal{C}_n) \leq k \Rightarrow \mathcal{N}^*(s, \textit{extract}_i) \leq k$. Recapitulate: $\forall M \in \mathbb{V}, \exists k \in \mathbb{N}, \forall i$ rank of extraction, $\underline{f}_i(\text{extract}_i) \leq M \Rightarrow \mathcal{N}^*(s, \text{extract}_i) \leq k$. This proposition is namely verified for $\forall M \in \mathbb{V}_1$. Moreover, $\forall M \in \mathbb{V}_2, \exists M' \in \mathbb{V}_1, M < M'$; for $M', \exists k \in \mathbb{N}, \forall i$ *rank of extraction*, $\underline{f}_i(\text{extract}_i) \leq M' \Rightarrow \mathcal{N}^*(s, \text{extract}_i) \leq k$; because $\underline{f}_i(\text{extract}_i) \leq M \Rightarrow \underline{f}_i(\text{extract}_i) \leq M'$, we have: $\underline{f}_i(\textit{extract}_i) \leq M \Rightarrow (s, \textit{extract}_i) \leq k$. Finally: $\forall M \in \mathbb{W}, \exists k \in \mathbb{N}, \forall i$ rank of extraction, $\underline{f}_i(\text{extract}_i) \leq M \Rightarrow \mathcal{N}^*(s, \text{extract}_i) \leq k$, thus the application of algorithm $\wp$ is circumscribed.

e) Thus we may apply Theorem 4.4. □

*Remarks about some rediscoveries and generalizations*: Nilsson's A is a particular case of $\wp^F_{\mathcal{E}}$ of type 1, 2, 3: $F$ is $h^*$ and $\mathcal{E}$ is the identity function $\mathcal{J}$. The reference monoid is $(\mathbb{R}^+, +)$ thus the reference length is $\mathcal{L}_{\text{add}}$. Nilsson consider son-finite $\delta$-graphs that contain at least one goal, that is to say particular $\mathcal{L}_{\text{add}}$-standard state graphs. Heuristic term $h$ is

positive. Taking $\mathbb{W} = \mathbb{V} = \mathbb{R}^+$, $\varphi = \mathcal{J}$ (identity function) we may apply Theorem 4.7: A terminates finding a path from $s$ to $T$.

Let us note that Theorem 4.7 may be invoked in very more general contexts; for instance, if we relax A, using a *non static* heuristic term $h$ but keeping the type 3 (and also: $f = g + h, 0 \leq h \leq h^*, \varphi = \mathcal{J}$), then we obtain an $\tilde{\text{A}}$ which necessarily terminates finding a path from $s$ to $T$.

Likewise, general Theorems 4.4 and 4.7 may be[18] applied to rediscover and extend the results about the completeness concerning the following algorithms: HPA, extended A of Pohl, B, $\text{A}^*_\varepsilon$, $\text{A}_\varepsilon$'s, C's, BF*'s, Mero's B′ algorithms, IDA*'s, D's, A**'s and SDW's.

## 5. General results to approach the sub-admissibility

The six original properties presented below form a set of tools to establish formulas of sub-admissibility. Statements 5.4, 5.5 and 5.6 directly provide formulas of sub-admissibility; Statements 5.4 and 5.6 presuppose the termination of the considered algorithms while the termination is a part of the conclusion in Statement 5.5.

### *5.1 Lemma for $\tilde{A}^F$'s of type 1, 2 or 3*

*Let $G$ be a state graph at which is applied an $\tilde{A}$ of type $1, 2$ or $3$ whose heuristic term $h$ is covered by $F$ (thus the algorithm is an $\tilde{A}^F$). Let $\mathcal{C}$ be a path of $G$ from $s$ to $T$ such as, $\forall i$ rank of extraction, $g_{i-1}(e_i) = \mathcal{L}(\mathcal{C}_{e_i})$ and $h^*(e_i)$ is defined, where $e_i$ is the input of $\mathcal{C}$ at the time to decide the $i$th extraction and $\mathcal{C}_{e_i}$ is the part of $\mathcal{C}$ from $s$ to $e_i$. Then: $\underline{f}_i(e_i) \leq \Theta(\mathcal{L}(\mathcal{C}_{e_i}), F(e_i))$ where $\underline{f}_i(e_i)$ is the evaluation value fastened to $e_i$ at the $i$th extraction.*

This result generalizes *Lemma 3.1* proposed by Nilsson[19] (1971) (also: *Result 2* by Nilsson (1980) and *Lemma 1* of Pearl (1984)).

**Proof:** By hypothesis, at the time to decide the $i$th extraction, we have $g_{i-1}(e_i) = \mathcal{L}(\mathcal{C}_{e_i})$. Let $j$, $j < i$, the rank of the extraction which orders: $g_j(e_i) \leftarrow \mathcal{L}(\mathcal{C}_{e_i})$; at the time of this same extraction, the algorithm has also evaluated $f_j(e_i) = \Theta(g_j(e_i), h_j(e_i)) = \Theta(\mathcal{L}(\mathcal{C}_{e_i}), h_j(e_i))$. If the algorithm is of type 3, taking into account that the last update of $g_x(e_i)$, before the rank $i$, has been realized at rank $x = j$, it is also at rank $j$ that has been realized the last update of $\underline{f}_x(e_i)$, thus: $\underline{f}_i(e_i) = \Theta(\mathcal{L}(\mathcal{C}_{e_i}), h_j(e_i))$; if the algorithm is of type 1 or 2, for any rank $x$ greater than $j$ we must have $\underline{f}_x(e_i) \leq f_j(e_i)$ thus $\underline{f}_i(e_i) \leq \Theta(\mathcal{L}(\mathcal{C}_{e_i}), h_j(e_i))$. Because $h^*(e_i)$ is defined, $F(e_i)$ is also defined and $h_j(e_i) \leq F(e_i)$. Because $\Theta$ is increasing in the right place: $\Theta(\mathcal{L}(\mathcal{C}_{e_i}), h_j(e_i)) \leq \Theta(\mathcal{L}(\mathcal{C}_{e_i}), F(e_i))$. Thus: $\underline{f}_i(e_i) \leq \Theta(\mathcal{L}(\mathcal{C}_{e_i}), F(e_i))$. □

### *5.2. Lemma for $\varrho'$s of type 3*

*Let $G$ be a state graph without absorbant circuit. Suppose that $\Theta$ is strictly increasing in the left place[20]. Let an algorithm $\varrho$ of type $3$ applied to $G$. Let $n$ be any node of $G$. Let $\mathcal{C}$ be any minimal path from $s$ to $n$. Denote $\mathcal{C} = n_0, \ldots, n_t$ where $n_0 = s$ and $n_t = n$. $\forall i$ rank of*

*extraction, if $n \notin$ closed, let $n_k (k \in \mathbb{N}, 0 \leq k \leq t)$ the input of $\mathcal{C}$ at the time to decide the ith extraction. Then:* $\forall j \in \mathbb{N}, 0 \leq j \leq k \Rightarrow g_{i-1}(n_j) = g^*(n_j)$.

This result generalizes the *Lemma 2* proposed by Pearl[21] (1984).

**Proof:** Because $G$ does not admit absorbant circuit, we have: $g_{i-1}(n_0) = e_n$; now: $g^*(n_0) = e_n$, thus: $g_{i-1}(n_0) = g^*(n_0)$. If $k = 0$ then the proposition is verified. Else, we shall execute a recurrence along $\mathcal{C}$. Suppose that: $\exists q \in \mathbb{N}, 0 \leq q < k, \forall j \in \mathbb{N}, 0 \leq j \leq q \Rightarrow g_{i-1}(n_j) = g^*(n_j)$.

By definition of input $n_k$: $n_q \in$ closed. Suppose that, at the time to decide the $i$th extraction, $n_q$ has been put in *closed* for the last time when the $p$th extraction has been executed (thus: $p < i$). Note: $n_q$ is the extracted node at the time of this $p$th extraction, thus $g_x(n_q)$ may be lowered only if $n_q$ is son of itself; but, because $G$ does not admit absorbant circuit: $g_p(n_q) = g_{p-1}(n_q)$. Then: $g_{p-1}(n_q) = g^*(n_q)$ else: $g_{p-1}(n_q) > g^*(n_q)$ (because $g_{p-1}(n_q)$ measures the length of a path from $s$ to $n_q$) and, in order to lower $g_x(n_q)$ from $g_{p-1}(n_q)$ to $g_{i-1}(n_q) = g^*(n_q)$, it is necessary that $n_q$ leaves *closed* (because the type of the algorithm is 3) and goes back in *closed* after the $p$th extraction, contrarily to the preceding hypothesis.

At the time of the $p$th extraction, the son $n_{q+1}$ of $n_q$ has been considered, thus: $g_p(n_{q+1}) \leq \Theta(g_{p-1}(n_q), c(n_q, n_{q+1})) = \Theta(g^*(n_q), c(n_q, n_{q+1}))$. Because $i - 1 \geq p$, we have: $g_{i-1}(n_{q+1}) \leq \Theta(g^*(n_q), c(n_q, n_{q+1}))$. Let $\mathcal{C}_q$ be the elementary sub-path of $\mathcal{C}$ from $s$ to $n_q$. Because $g^*(n_q) \leq \mathcal{L}(\mathcal{C}_q)$ (by definition of $g^*$) and because $\Theta$ is increasing in the left place: $g_{i-1}(n_{q+1}) \leq \Theta(\mathcal{L}(\mathcal{C}_q), c(n_q, n_{q+1})) = \mathcal{L}(\mathcal{C}_{q+1})$.

Because $\mathcal{C}$ is minimal from $s$ to $n$ and because $\Theta$ is strictly increasing in the left place, it may be easily verified that: $\mathcal{L}(\mathcal{C}_{q+1}) = g^*(n_{q+1})$. Thus: $g_{i-1}(n_{q+1}) \leq g^*(n_{q+1})$.

Moreover, because $g_{i-1}(n_{q+1})$ measures the length of a path from $s$ to $n_{q+1}$: $g_{i-1}(n_{q+1}) \geq g^*(n_{q+1})$. Finally: $g_{i-1}(n_{q+1}) = g^*(n_{q+1})$.

We may iterate, while $q < k$. So we obtain: $g_{i-1}(n_k) = g^*(n_k)$. □

### *5.3. Corollary for $\tilde{A}^F$'s of type 3*

*Let $G$ be a state graph without absorbant circuit. Suppose that $\Theta$ is strictly increasing in the left place. Let $\mathcal{C}$ be any minimal path of $G$ from $s$ to $T$. Suppose that an $\tilde{A}$ of type 3 is applied to $G$, whose heuristic term $h$ is covered by $F$ (thus the $\tilde{A}$ is an $\tilde{A}^F$). Then, $\forall i$ rank of extraction: $\underline{f}_i(e_i) \leq \Theta(g^*(e_i), F(e_i))$ where $e_i$ is the input of $\mathcal{C}$ at the time to decide the ith extraction while $\underline{f}_i(e_i)$ is the evaluation value fastened to $e_i$ at this precise moment.*

This result, derived from the general Lemma 5.2, yet widely includes *Lemma 3.1* proposed by Nilsson (1971).

**Proof:** Let $\mathcal{C} = n_0, \ldots, n_t$ with $s = n_0$ and $n_t \in T$; let $n_k (k \in \mathbb{N}, 0 \leq k \leq t)$ the input of $\mathcal{C}$ at the time to decide the $i$th extraction. Because $\mathcal{C}$ is minimal from $s$ to $n$ and because $\Theta$ is strictly increasing in the left place, it may be easily verified that the length of the part of $\mathcal{C}$ from $s$ to $n_k$ is minimal between $s$ and $n_k$: it defines $g^*(n_k)$, while the length of the part of $\mathcal{C}$ from $n_k$ to $T$ is minimal between $n_k$ and $T$: it defines $h^*(n_k)$.

According to Lemma 5.2, at the time to decide the $i$th extraction, we have: $g_{i-1}(n_k) = g^*(n_k)$. Let $j$, $j < i$, the rank of the extraction that orders: $g_j(n_k) \leftarrow g^*(n_k)$; at the time of the same extraction, the algorithm has also evaluated $f_j(n_k) = \Theta(g_j(n_k), h_j(n_k)) = \Theta(g^*(n_k), h_j(n_k))$.

Because the algorithm is of type 3, because the last update of $g_x(n_k)$ before rank $i$ has been executed at rank $x = j$, we conclude that the last update of $\underline{f}_x(n_k)$ has been executed at rank $j$, thus: $\underline{f}_i(n_k) = \Theta(g^*(n_k), h_j(n_k))$.

Now, there exists a function $F$ taking its values on $\mathbb{V}$, such as: for any evaluated node $n$ for which $h^*(n)$ is defined, $h_x(n) \leq F(n)$. Here: $h^*(n_k)$ is defined, thus: $h_j(n_k) \leq F(n_k)$. And, because $\Theta$ is increasing in the right place: $\Theta(g^*(n_k), h_j(n_k)) \leq \Theta(g^*(n_k), F(n_k))$. Thus: $\underline{f}_i(n_k) \leq \Theta(g^*(n_k), F(n_k))$. □

*5.4. Theorem of extraction in case of termination, for $\tilde{\mathrm{A}}_{\mathcal{E}}^{F}$'s of type 3*

*Let G be a state graph without absorbant circuit and that contains at least one goal. Let an algorithm $\tilde{\mathrm{A}}_{\mathcal{E}}$ of type 3 applied to G, which terminates. Suppose that, when the algorithm is applied, h is covered by F (thus the algorithm is an $\tilde{A}_{\mathcal{E}}^{F}$) and quasi-coincident (thus: $\exists m_T \in \mathbb{V}, \exists m'_T \in \mathbb{V}, \Theta(m_T, m'_T) \geq e_n$). Suppose that $\Theta$ is strictly increasing in the left place. Then, at the time of termination, the algorithm has found a path $\mathcal{C}$ from s to T such as: $\mathcal{L}(\mathcal{C}) \leq \Theta(\mathcal{E}(\Theta(g^*(e_i), F(e_i))), m'_T)$, where $e_i$ is the input, when is decided the ith and last extraction, of any minimal path from s to T in G.*

**Proof:** According to Lemma 4.3, at the time of the termination, the algorithm extracts a goal $t$. Whatever the type of the algorithm: $\exists k \in \mathbb{N}, k < i$, $\underline{f}_i(t) = f_k(t) = \Theta(g_k(t), h_k(t))$. According the definition of $g$: $g_k(t) \geq g_{i-1}(t)$. Because $h_x$ is quasi-coincident: $h_k(t) \geq m_T$. Because $\Theta$ is increasing in the left and right places: $\underline{f}_i(\mathrm{extract}_i) = \underline{f}_i(t) \geq \Theta(g_{i-1}(t), m_T)$.

Thus: $\Theta(\underline{f}_i(\mathrm{extract}_i), m'_T) \geq \Theta(\Theta(g_{i-1}(t), m_T), m'_T) = \Theta(g_{i-1}(t), \Theta(m_T, m'_T)) \geq \Theta(g_{i-1}(t), e_n) = g_{i-1}(t)$.

Let $\mathcal{C}$ be the pointer path of $t$ at rank $i$. For any algorithm ℚ of type 2 ou 3, it may be easily verified that: $\mathcal{L}(\mathcal{C}) = g_{i-1}(t)$; thus: $\mathcal{L}(\mathcal{C}) \leq \Theta(\underline{f}_i(\mathrm{extract}_i), m'_T)$.

According to Corollary 5.3: at the time to execute the $i$th extraction, the input $e_i$ of $\mathcal{C}$ is in $open_i$ and: $\underline{f}_i(e_i) \leq \Theta(g^*(e_i), F(e_i))$. According to the definition of the $\mathcal{E}$-extraction:

$$\underline{f}_i(\mathrm{extract}_i) \leq \mathcal{E}\left(\min_{n \in \mathrm{front}_i} \underline{f}_i(n)\right).$$

Because $\mathcal{E}$ is increasing on $\mathbb{V}$ and because $e_i \in open_i$, we have: $\underline{f}_i(\mathrm{extract}_i) \leq \mathcal{E}(\Theta(g^*(e_i), F(e_i)))$.

Thus: $\mathcal{L}(\mathcal{C}) \leq \Theta(\mathcal{E}(\Theta(g^*(e_i), F(e_i))), m'_T)$. □

In the following theorem, the termination of the algorithm is not an hypothesis but a conclusion.

*5.5. Theorem of sub-admissibility for $\tilde{A}_{\mathcal{E}}^{F}$'s of type 3 when h is covered, quasi-coincident and lower-bounded*

*Let G be a $\mathcal{L}$-standard state graph at which is applied an algorithm $\tilde{A}_{\mathcal{E}}$ of type* 3, *whose heuristic function h is covered by F during the application* (*thus the algorithm is an $\tilde{A}_{\mathcal{E}}^{F}$*) *quasi-coincident* (*thus: $\exists m_T \in \mathbb{V}, \exists m'_T \in \mathbb{V}, \Theta(m_T, m'_T) \geq e_n$*) *and lower-bounded by an element m of $\mathbb{V}$ such as $\exists m' \in \mathbb{V}, \Theta(m', m) \geq e_n$. Suppose that $\Theta$ is strictly increasing in the left place. Then*, *the algorithm terminates finding a path $\mathcal{C}$ of G from s to T such as: $\mathcal{L}(\mathcal{C}) \leq \Theta(\mathcal{E}(\Theta(g^*(e_i), F(e_i))), m'_T)$*, *where $e_i$ is the input*, *at the time to decide the ith and last extraction*, *of any minimal path from s to T in G.*

This result generalizes Theorem 3.1 proposed by Nilsson[22] (1971) (also: Result 4 in Nilsson (1980) and Theorem 2 in Pearl (1984)).

**Proof:** a) Because the evaluation function has the form: $f_x(n) = \Theta(g_x(n), h_x(n))$, it takes its values in $\mathbb{V}$; thus $\mathbb{W} \subset \mathbb{V}$; thus $\mathbb{W}$ is overpassed by $\mathbb{V}$.

Because $\Theta$ is increasing in the right place: $f_x(n) \geq \Theta(g_x(n), m)$. Let $\varphi$ be the function $\varphi \mid y \in \mathbb{V} \rightarrow \varphi(y) = \Theta(y, m) \in \mathbb{V}$. Thus: $f_x(n) \geq \varphi(g_x(n))$. Because $g_x(n)$ measures the length of a path from $s$ to $n$: $\varphi(g_x(n)) \geq \varphi(g^*(n))$, thus: $f_x(n) \geq \varphi(g^*(n))$.

Function $\varphi$ is overpassing on $\mathbb{V}$ because: $\forall x \in \mathbb{V}, \exists y \in \mathbb{V}$ such as: $x \leq \varphi(y)$; indeed, taking $y = \Theta(x, m')$ and knowing that $\Theta$ is associative and increasing in the right place, we verify that: $\varphi(y) = \Theta(y, m) = \Theta(\Theta(x, m'), m) = \Theta(x, \Theta(m', m)) \geq \Theta(x, e_n) = x$. Because $\Theta$ is strictly increasing in the left place, we observe that $\varphi$ is strictly increasing thus infra strictly increasing on $\mathbb{V}$.

So we may apply Theorem 4.7 and conclude that the algorithm terminates immediately after extracting a goal from *open*.

b) Thus we may apply Theorem 5.4 and infer that the algorithm discovers a path $\mathcal{C}$ from $s$ to $T$ such as $\mathcal{L}(\mathcal{C}) \leq \Theta(\mathcal{E}(\Theta(g^*(e_i), F(e_i))), m'_T)$, where $e_i$ is the input of $\mathcal{C}$ at the time of the last extraction $i$. □

In the following lemma, the termination of the algorithm is anew an hypothesis.

*5.6. Lemma of the found path, for $\wp$'s of type 1, 2 or 3*

*Let G be a state graph that contains at least one goal. Let an algorithm $\wp$ of type* 2 *or* 3, *that is applied to G and terminates at the time of the ith extraction. Suppose that there exists a function $\Omega$, whose domain is the range $T_f$ of $f_x(n)$ when n runs on T such as*: *for any evaluated goal node t, $g_x(t) \leq \Omega(f_x(t))$. Then*, *at the time of the termination*, *the algorithm has found a path $\mathcal{C}$ from s to T such as*: *$\mathcal{L}(\mathcal{C}) \leq \Omega(\underline{f}_i(extract_i))$. This lemma holds for the algorithms $\wp$ of type* 1 *when the property of homogeneity is satisfied.*

This result generalizes *Theorem 2*$^*$ proposed by Pearl[23] (1984).

**Proof:** According to Lemma 4.3, at the time of the termination, the algorithm extracts a goal $t_0$. Whatever the type of the algorithm: $\exists k \in \mathbb{N}, k < i, \underline{f}_i(\text{extract}_i) = \underline{f}_i(t_0) = f_k(t_0)$. Let $\mathcal{C}$ be the pointer path of $t_0$ at rank $i$; we have noted (Section 3.8) that if the algorithm is of type 2 or 3 then $\mathcal{L}(\mathcal{C}) = g_{i-1}(t_0)$; this relation holds for the algorithms of type 1 when the

property of homogeneity paths is satisfied, because $t_0$ is previously appeared and is situated on a path from $s$ to $T$. According to the definition of $g$: $g_{i-1}(t_0) \leq g_k(t_0)$; by hypothesis: $g_k(t_0) \leq \Omega(f_k(t_0))$. Thus: $\mathcal{L}(\mathcal{C}) \leq \Omega(f_k(t_0)) = \Omega(\underline{f}_i(\text{extract}_i))$. □

*Remarks about some rediscoveries and generalizations:* Nilsson's A* algorithm is also a particular case of $\tilde{A}^{\mathbf{F}}_{\mathcal{E}}$ of type 3. The state graphs considered by Nilsson contain at least one goal and do not admit absorbant circuits. The heuristic term is quasi-coincident (take $m_T = m'_T = 0$). The operation $\Theta$ (that is to say $+$), is strictly increasing in the left place. Thus we may apply Theorem 5.4: *in case of termination*, $\mathcal{L}_{\text{add}}(\mathcal{C}) \leq g^*(e_i) + h^*(e_i)$, where $\mathcal{L}_{\text{add}}(\mathcal{C})$ is the length of the found path $\mathcal{C}$ from $s$ to $T$, while $e_i$ is the final input of any minimal path $\mathcal{C}_0$ from $s$ to $T$[24]. Because $e_i$ belongs to $\mathcal{C}_0$ (minimal), it may be easily verified that: $g^*(e_i) + h^*(e_i) = \mathcal{L}_{\text{add}}(\mathcal{C}_0)$. Thus $\mathcal{L}_{\text{add}}(\mathcal{C}) = h^*(s)$: *in case of termination* the found path is minimal. Really, we may right away apply Theorem 5.5, which moreover assures the termination (also proved in Section 4.8, applying Theorem 4.7): the admissibility of A*'s is thus rediscovered.

The admissibility may be still proved when some constraints applied to Nilsson's A* algorithm are relaxed; indeed, providing that the updating type remains type 3, Theorem 5.5 is yet applicable to monoids $(\mathbb{V}, \Theta)$ other than group $(\mathbb{R}^+, +)$, to state graphs that are not necessarily $\delta$-graphs, to heuristic terms $h$ that are not necessarily static or positive. Moreover, Theorem 5.5 gives some formulas of sub-admissibility for extraction modes which may be not best-first and for $h_x$ which may be not lower-bounding. If we relax Nilsson's A* algorithms towards $\tilde{A}$'s of type 1 or 2, we can establish some formulas of sub-admissibility by combining the Result 4.7 (possibly 4.4) with Result 5.6.

Likewise, Theorems 5.1–5.6 may be[25] applied to rediscover and extend the known results about the admissibility or the sub-admissibility concerning the HPA's, extended A's of Pohl, B's, $A^*_\varepsilon$'s, $A_\varepsilon$'s, C's, BF*'s, Mero's B′ algorithms, IDA*'s, D's, A**'s and SDW's.

## 6. Concluding remarks and perspectives

We have proposed a formalization that generalizes diverse works concerning the Heuristically-Ordered Search in state graphs. We have considered 5 dimensions: 1) the notion of length to measure the paths between nodes, 2) the characteristics of the state graphs dealt with, 3) the choices of the nodes to expand, 4) the kinds of updating to realize, 5) the properties of the evaluation functions that guide the search. We have employed this formalization to present several general theorems about the completeness and about the admissibility/sub-admissibility. This formalization and the derived results allow a comparative presentation of the algorithms; they facilitate a better understanding of the key points and limitations, so as the revelation of non exploited potentialities.

This work may be developed to tackle the problems of completeness, admissibility or sub-admissibility for other variants of algorithms or perhaps other variants of evaluation functions, or even other variants of state graphs, beyond the hypotheses here considered. So we may be concerned with real-time algorithms such as RTA* (Korf, 1988b, 1990), with restricted-memory algorithms such as MREC (Sen and Bagchi, 1989), MA* (Chakrabarti et al., 1989), SMA* (Russell, 1992), RFBS (Korf, 1992, 1993) or with algorithms for dynamic environnements such as D* (Stentz, 1995). We may also try to rediscover and

extend the sub-admissibility results relative to the bidirectional algorithms (see (Farreny, 1995), chapter 8, for preliminary work). The generalizing formalization that we have only applied here to the properties of completeness, admissibility and sub-admissibility may be also contribute to better apprehend *other properties* knowingly ignored in this paper; for instance: such algorithm always finds a better solution than such other, such algorithm presents such kind of complexity in time or space, etc.

## Notes

1. The definitions of the completeness, the admissibility and the sub-admissibility are recalled in Section 2.
2. It is not possible to recall here the definitions of these algorithms.
3. The notation A* (Nilsson, 1971) is considered further.
4. The number of sons of any node is finite. Some authors say that the graph is *locally finite*.
5. For Nilsson, the length of a path $\mathcal{C}$ is the sum of the arc costs of $\mathcal{C}$. We denote it: $\mathcal{L}_{\text{add}}(\mathcal{C})$.
6. We use here the adjective *dynamic* with the meaning introduced by Pohl (1973). Mero (1984) then Mahanti and Ray (1988) use the adjective *modifiable*. Kainz, Kaindl and Köll (1992, 1996) use *dynamic* with a more restrictive meaning than Pohl and us.
7. Other authors (Pohl, 1977; Pearl, 1984; Korf, 1985; Mahanti and Ray, 1988) say that $h$ is *admissible* rather than lower-bounding.
8. Gelperin (1977) gives a particular condition for guaranteeing admissibility of A* when the state graphs contain non-positive arc costs. We do not know another work dealing with this problem.
9. Nevertheless the interest of other forms of length is suggested by Schoppers (1983) and Pearl (1984) and illustrated by Yager (1986), Farreny (1996c) and Gonella (1989).
10. $\mathbb{V}$ is closed under $\Theta$, $\Theta$ is associative and admits an identity element in $\mathbb{V}$. We denote: $(\mathbb{V},\Theta)$.
11. This particular form of $\mathcal{E}$-extraction is also considered by Davis (1988).
12. Note: the tilde on A reminds the *a priori* non staticity of $h$.
13. Several authors consider a static $h$ such that: for any goal state $t$, $h(t) = 0$; this property is sometimes called *coincidence*.
14. Length relative to some monoid $(\mathbb{V}, \Theta)$; the arc valuations are taken in $\mathbb{V}$ (see Section 3.1).
15. That is to say: without repeating any state.
16. Absorbant circuit: whose length $< e_n$ ($e_n$: identity element of the considered monoid).
17. Because $f \geq \varphi(g^*)$, if $\forall n$ state, $\exists \delta_n > 0$, $\forall i$ and $j$ ranks of extraction $| f_i(n) - f_j(n) | \geq \delta_n$ then it is assured that f is finitely decreasing. This case appears for instance when $f$ only may take decimal values with at most $p$ figures after the point (particular case: integer values).
18. We don't give the details by lack of place.
19. Statement of *Lemma 3.1* of Nilsson (1971): *If $\hat{h}(n) \leq h(n)$ for all $n$, then at any time before A* terminates and for any optimal path P from node s to goal, there exists an open node $n'$ on P with $\hat{f}(n') \leq f(s)$.*
20. For instance, this property is true if $(\mathbb{V}, \Theta)$ is a group rather than a simple monoid.
21. Statement of *Lemma 2* of Pearl (1984): *Let $n'$ be the shallowest OPEN node on an optimal path $P^*_{s-n''}$ to any arbitrary node $n''$, not necessarily in $\Gamma$. Then: $g(n') = g^*(n')$, stating that A* has already found the optimal pointer-path to $n'$ (i.e., $n'$ is along $P^*_{s-n''}$) and that path will remain unaltered throughout the search.*
22. Statement of Theorem 3.1 in Nilsson (1971): *If $h(n) \leq h(n)$ for all nodes $n$, and if all arc costs are greater than some small positive number $\delta$, then algorithm A* is admissible.*
23. *BF* is $\psi^{-1}(M)$-admissible, that is, the cost of the solution path found by BF* is at most $\psi^{-1}(M)$.*
24. For any $\mathcal{L}$-standard graph, there exists a minimal path from $s$ to $T$.
25. We don't give the details by lack of place.

## References

Bagchi, A. and A. Mahanti. (1983). "Search Algorithms Under Different Kinds of Heuristics—A Comparative Study," *J. ACM* 30(1), 1–27.

Bagchi, A. and A. Mahanti. (1985). "Three Approaches to Heuristic Search in Networks," *J. ACM* 32(1), 1–27.

Chakrabarti, P.P., S. Ghose, A. Acharya, and S.C. de Sarkar. (1989). "Heuristic Search in Restricted Memory," *Art. Int.* 41(2), 197–221.

Davis, H.W., A. Bramanti-Gregor, and J. Wang. (1988). "The Advantages of Using Depth and Breadth Components in Heuristic Search." In Z.W. Ras and L. Saitta (eds.), *Methodologies for Intelligent Systems 3*. Elsevier Science Publishing Co., pp. 19–28.

Dechter, R. and J. Pearl. (1985). "Generalized Best-First Search Strategies and the Optimality of A*," *J. ACM* 32(3), 505–536.

Dechter, R. and J. Pearl. (1988). "The Optimality of A*." In L. Kanal and D. Kumar (eds.), *Search in Art. Int.* Springer-Verlag, pp. 166–199.

Dubois, D., J. Lang, and H. Prade. (1987). "Theorem Proving Under Uncertainty. A Possibility Theory-Based Approach." *Proc. 10th IJCAI*. Milan, Italy, pp. 984–986.

Farreny, H. (1995). *Recherche Heuristiquement Ordonnée: Algorithmes et Propriétés*. Masson, Paris.

Farreny, H. (1996a). "Une Généralisation Pour la Recherche Heuristiquement Ordonnée: Les Algorithmes $\varrho$ et la Propriété d'arrêt Avec Découverte de Solution." *Proc. RFIA 96*. Rennes, France, pp. 225–234.

Farreny, H. (1996b). "A Generalization for Heuristically-Ordered Search: Algorithms $\varrho$, Results About Termination and Admissibility." *Proc. IEEE Int. Conf. on SMC* (*2*). Beijing, China, pp. 1442–1447.

Farreny, H. (1996c). "Algorithms $\varrho$ for $\mathcal{L}$-Admissible Standard State Graphs." *Proc. SBIA 96*. Curitiba, Brazil. In D.L. Borges and C.A.A. Kaestler (eds.), *Lectures Notes in Artificial Intelligence 1159*. Springer, pp. 41–50.

Farreny, H. (1997a). "Recherche Heuristiquement Ordonnée dans les Graphes d'états: Élargissement des cas D'admissibilité et Sous-Admissibilité," *Revue d'Intelligence Artificielle* 11(4), 407–448.

Farreny, H. (1997b). "New Results About Sub-Admissibility for General Families of Heuristic Search Algorithms." *Proc. EPIA 97*. Coimbra, Portugal. In E. Costa and A. Cardoso (eds.), *Lectures Notes in Artificial Intelligence 1323*. Springer, pp. 241–254.

Farreny, H. (1997c). "Recherche Heuristiquement Ordonnée dans les Graphes d'états: Élargissement des cas D'admissibilité et Sous-Admissibilité," *Revue d'Intelligence Artificielle* 11(4), 407–448.

Gelperin, D. (1977). "On the Optimality of A*," *Art. Int.* 8(1), 69–76.

Ghallab, M. (1982). Optimisation de processus décisionnels pour la robotique, Thèse de Doctorat d'Etat, Université Paul Sabatier, Toulouse, France.

Ghallab, M. and D.G. Allard. (1982). "Near Admissible Heuristic Search Algorithm." *Proc. 2d World Conf. on Mathematics at the Service of Man*. Las Palmas, Spain.

Ghallab, M. and D.G. Allard. (1983). "$A_\varepsilon$: An Efficient Near Admissible Heuristic Search Algorithm." *Proc. 8th IJCAI*. Karlsruhe, Germany, pp. 789–791.

Gonella, R. (1989). Diagnostic de pannes sur avions: mise en oeuvre d'un raisonnement révisable, Thèse de l'École Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, France.

Harris, L.R. (1973). "The Bandwith Heuristic Search." *Proc. 3d IJCAI*. Stanford, USA, pp. 23–29.

Harris, L.R. (1974). "The Heuristic Search Under Conditions of Error," *Art. Int.* 5(3), 217–234.

Hart, P.E., N.J. Nilsson, and B. Raphael. (1968). "A Formal Basis for the Heuristic Determination of Minimal Cost Paths," *IEEE Trans. on SSC* 4, 100–107.

Hart, P.E., N.J. Nilsson, and B. Raphael. (1972). "Correction to: A Formal Basis for the Heuristic Determination of Minimal Cost Paths," *SIGART Newsletter* 37, 28–29.

Kainz, G. and H. Kaindl. (1996). "Dynamic Improvements of Heuristic Evaluations During Search." *Proc. 13th AAAI*. Portland, USA, pp. 311–317.

Köll. L.A. and H. Kaindl. (1992). "A New Approach to Dynamic Weighting." *Proc. 10th ECAI*. Vienna, Austria, pp. 16–17.

Korf, R.E. (1985a). "Iterative-Deepening-A*: An Optimal Admissible Tree Search." *Proc. 9th IJCAI*. Los Angeles, USA, pp. 1034–1036.

Korf, R.E. (1985b). "Depth-First Iterative-Deepening: An Optimal Admissible Tree Search," *Art. Int.* 27, 97–109.

Korf, R.E. (1988a). "Optimal Path-Finding Algorithms." In L. Kanal and D. Kumar (eds.), *Search in Art. Int.* Springer-Verlag, pp. 223–267.

Korf, R.E. (1988b). "Real-Time Heuristic Search: New Results." *Proc. 7th AAAI*. Saint-Paul, Minnesota, USA, pp. 139–144.

Korf, R.E. (1990). “Real-Time Heuristic Search,” *Art. Int.* 42, 189–211.
Korf, R.E. (1992). “Linear-Space Best-First Search.” *Proc. 10th AAAI*. San José, Cal., USA, pp. 533–538.
Korf, R.E. (1993). “Linear-Space Best-First Search,” *Art. Int.* 62, 41–78.
Mahanti, A. and K. Ray. (1988). “Network Search Algorithms with Modifiable Heuristics.” In L. Kanal and D. Kumar (eds.), *Search in Art. Int.* Springer-Verlag, pp. 200–222.
Martelli, A. (1977). “On the Complexity of Admissible Search Algorithms,” *Art. Int.* 8(1), 1–13.
Mero, L. (1984). “A Heuristic Search Algorithm with Modifiable Estimate,” *Art. Int.* 23(1), 13–27.
Nilsson, N.J. (1971). *Problem-Solving Methods in Artificial Intelligence*. Mc Graw-Hill.
Nilsson, N.J. (1980). *Principles of Artificial Intelligence. Tioga*.
Pearl, J. (1981). “Heuristic Search Theory: Survey of Recent Results.” *Proc. 7th IJCAI*. Vancouver, Canada, pp. 554–562.
Pearl, J. (1984a). “Some Recent Results in Heuristic Search Theory,” *IEEE Trans. on PAMI* 6(1), 1–12.
Pearl, J. (1984b). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.
Pearl, J. and J.H. Kim. (1982). “Studies in Semi-Admissible Heuristics,” *IEEE Trans. on PAMI* 4(4), 392–400.
Pohl, I. (1969). “First Results on the Effect of Error in Heuristic Search.” In B. Meltzer and D. Michie (eds.), *Mach. Int.* 5 Edinburgh University Press, pp. 219–236.
Pohl, I. (1970). “Heuristic Search Viewed as Path Finding in a Graph,” *Art. Int.* 1(4), 193–204.
Pohl, I. (1973). “The Avoidance of (Relative) Catastrophe, Heuristic Competence, Genuine Dynamic Weighting and Computational Issues in Heuristic Problem Solving.” *Proc. 3d IJCAI*. Stanford, USA, pp. 12–17.
Pohl, I. (1977). “Practical and Theoretical Considerations in Heuristic Search Algorithms.” In E.W. Elcock and D. Michie (eds.), *Mach. Int.* 8, Wiley, pp. 55–71.
Russell, S. (1992). “Efficient Memory-Bounded Search Methods.” *Proc. 10th ECAI*. Vienna, Austria, pp. 1–5.
Schoppers, M.J. (1983). “On A$^*$ as a Special Case of Ordered Search.” *Proc. 8th IJCAI*. Karlsruhe, Germany, pp. 783–785.
Sen, A.K. and A. Bagchi. (1989). “Fast Recursive Formulations for Best-First Search that Allow Controlled Use of Memory.” *Proc. 11th IJCAI*. Detroit, USA, pp. 297–302.
Stentz, A. (1995). “The Focused D$^*$ Algorithm for Real-Time Replanning.” *Proc. 14th IJCAI*. Montreal, Canada, pp. 1652–1659.
Steward, B., C.F. Liaw, and C.C. White III. (1994). “A Bibliography of Heuristic Search Research Through 1992,” *IEEE Trans. on SMC* 24(2), 268–293.
Vanderbrug, G.J. (1976). “Problem Representations and Formal Properties of Heuristic Search,” *Information Sciences*. 11(4), 279–307.
Yager, R.R. (1986). “Paths of Least Resistance in Possibilitic Production Systems,” *Fuzzy Sets and Systems* 19, 121–132.