This section is an additional appendix to "Optimal Implementation of Watched Literals and More General Techniques" by Ian P. Gent, Journal of Artificial Intelligence Research, 2013

Appendix E. Additional Proofs

Proposition 3. *The procedure* FNE-NoState *maintains Invariant 2. It makes* $O(N^2)$ *calls to* acceptable *per branch of a search tree, but requires* $\Theta(N^2)$ *in the worst case.*

Proof. Correctness should need no proof. The bound $O(N^2)$ in a branch holds because each of the N values in the list can become unacceptable at most once, and each such event needs at most N calls to acceptable. For the worst case, consider a tree in which list[0] is acceptable at all internal nodes, while at each leaf node, each element in list becomes unacceptable in numerical order. At each leaf node there are $N-1$ calls to FNE-NoState, which require in turn $2, 3, 4, \ldots, N$ calls to acceptable, for a total of $\Theta(N^2)$. □

Theorem 7. (Correctness) *If the search algorithm defines a downwards-explored search tree, and if procedure* findNewElement *is locally correct, Invariant 2 is true at all times.*

Proof. Consider any time which is not in the interval between unacceptability occuring and either backtracking or a call to findNewElement exiting, so we have to establish one of the first two conditions of Invariant 2. There are two cases.

The first case is that the last call to findNewElement exited with success at some node. At that node the value list[*last*] was acceptable (by local correctness). There are two subcases. First, if list[*last*] has *not* become unacceptable since then, it is still acceptable. Second, if list[*last*] *has* become unacceptable since then, consider the most recent occurrence of this. Backtracking must have occurred before a new call to findNewElement. At the parent node to which search returned, the value list[*last*] was acceptable, and therefore it has been from that point until now.

The second case is that the last call to findNewElement failed, so at the node n_1 where the call happened, no acceptable value existed in the list. There are two subcases. First, if the current time is during the remainder of the visit to n_1, or at any descendant node of n_1, then by monotonicity of acceptability, there must still be no acceptable element in the list. The second subcase is that the current time is after backtracking from n_1. But, when findNewElement fails, by local correctness the value of *last* is unchanged from entry. Therefore at the start of the visit to node n_1, the value list[*last*] must have been acceptable: the call to findNewElement at n_1 happened because list[*last*] became unacceptable. Let n_0 be the parent node of n_1 (which must exist because if n_1 were the root we would be in the first subcase.) The value list[*last*] was acceptable at n_0. Since the most recent call to findNewElement was at n_1, and we have exited from n_1 and all its descendants, unacceptability cannot have been detected since backtracking returned to n_0, so list[*last*] is acceptable at the current time. □

Lemma 9. *Every node in the tree is contained in exactly one left branch segment. In a downwards-explored search tree, nodes in a left branch segment are visited consecutively without search visiting any other nodes.*

Proof. The first statement is immediate from the definition of LBS. For the second, downwards-explored search always visits the left child of a node before other nodes. Since this property is recursive, from any node the next sequence of nodes visited in order is the left child, the left grandchild, etc. From the origin of a left branch segment, the nodes in the segment are therefore visited consecutively without any other nodes intervening. □

Theorem 13. (Optimality) *In any downwards-explored search tree, the circular approach requires space for one* last *pointer and has a worst case of $O(N)$ calls to* ACCEPTABLE *per branch of the tree, and no algorithm can require $o(N)$ calls per branch.*

Proof. The first part follows from Theorem 12 and the fact that we just need one *last* pointer. For the second part, any algorithm must potentially check all elements in the list for acceptability (except the current value of *last*) on every branch, or else it cannot determine if a list has no acceptable elements. Since it is possible for all these $N-1$ checks to be at leaf nodes, we can need $\Omega(N)$ calls per branch of the tree. □

Proposition 14. *For circular, there can be as many as $2k(N-2)$ calls to* ACCEPTABLE *in a downwards-explored search tree. For state restoration, the number of calls to* ACCEPTABLE *is bounded above by kN and there can be $k(N-1)$ calls to* ACCEPTABLE *in a tree.*

Proof. For the first part, we suppose that at each internal node of the tree, all list elements are acceptable, so calls to ACCEPTABLE are all at leaf nodes. To achieve $2N-2$ calls to ACCEPTABLE at each leaf node, first the current value of *last* must become unacceptable, and in one or more calls to FNE-CIRCULAR we must eventually find $last-1$ to be acceptable (or $N-1$ if $last = 0$). This requires $N-1$ calls to ACCEPTABLE. Finally this list element must become unacceptable, meaning a final call to FNE-CIRCULAR which will fail with a further $N-1$ calls to ACCEPTABLE. Since the list scanning framework is general, one could construct an ACCEPTABLE function and search procedure to satisfy these requirements. The total number of calls to ACCEPTABLE is then $2k(N-2)$.

For the second part, no more than N calls to ACCEPTABLE can be made on any branch, hence the bound of kN. In the situation above, we have $N-1$ calls at each leaf node. □

Proposition 15. (Non Dominance) *Both techniques can check less than the other across a downwards-explored search tree. The circular method can take $\Omega(k)$ times fewer calls to* ACCEPTABLE *across the tree than state restoration, while state restoration can need $\Omega(N)$ times fewer calls than circular.*

Proof. For circular being better, suppose that at all internal nodes only the list elements 0 and $N-1$ are acceptable, while at each leaf node 0 becomes unacceptable. Initialisation will set *last* to 0. At the first leaf node circular will set *last* to $N-1$ using $N-2$ further calls, and *last* will not change during the rest of search. By contrast state restoration must make these $N-2$ calls at every leaf node. So circular makes $N-1$ calls across the tree while state restoration uses $k(N-2)+1$.

For state restoration being better, suppose that at each internal node the list elements 0 and 1 are acceptable, while leaf nodes alternate between only 1 being acceptable and only 0 being acceptable. Initialisation will again set *last* to 0. At half of leaf nodes, state restoration does no work, and at the other half it requires just one call to ACCEPTABLE, for a total of $\lceil k/2 \rceil$ calls. The circular method again needs one call at half of leaf nodes, when moving from 0 to 1, but at the other half moves from 1 to 0 at a cost of $N-1$ calls each time. This totals $\lceil k/2 \rceil + (N-1)\lfloor k/2 \rfloor + 1 = \Omega(kN)$. □

Lemma 19. *In a downwards-explored search tree, each call to* FNE-Circular-W *either returns false or reduces by one the number of unacceptable elements in the set* watched $\cup$ $\{$unwatched$[last]\}$.

Proof. The first case is that $elt = $ watched$[i]$ at Line 3 and therefore unwatched$[last]$ is unacceptable. This reduces to a pass-through call to FNE-Circular and the result follows from Theorem 7: after a successful call unwatched$[last]$ must now be acceptable. The other case is that $elt \neq$ watched$[i]$. We now swap watched$[i]$ and unwatched$[last]$, which does not change the set watched $\cup$ $\{$unwatched$[last]\}$. As in the first case, unwatched$[last]$ will change from being unacceptable to acceptable after a successful call. □

Theorem 20. (Correctness) *In a downwards-explored search tree,* FNE-Circular-W *maintains Invariant 18.*

Proof. Each time the number of unacceptable elements in watched $\cup$ $\{$unwatched$[last]\}$ increases by one, it causes one pending call to findNewElement. If any such call returns false the second clause of the invariant applies, and when some remain pending the last one does. Therefore we need consider only the case that all pending calls have been made and were successful. In this case, Lemma 19 shows that the number of unacceptable elements in the set decreases by one per call, meaning that there is no net change overall. □

Corollary 21. *For a constraint of arity r where each variable is domain size d, in a downwards-explored search tree the circular approach to maintaining the last pointer in MGAC2001/3.1 can be achieved using space to store $O(dr)$* last *pointers (beyond the storage space for the constraint itself), and requires time to check $O(rd^r)$ tuples per branch.*

Proof. We need to maintain support for $O(dr)$ variable-value pairs in the constraint. Each one requires a separate *last* pointer so we need space for $O(dr)$ pointers. Acceptability is that every value in a supporting tuple is still in its variable's domain, i.e. a tuple is valid. The time complexity part of the result follows from Theorem 12. In a tree with k branches, there are at most $k(2N-2)$ calls to acceptable, i.e. $2N-2$ per branch. The maximum number of supporting tuples for a variable-value pair is $O(d^{r-1})$, if every combination of every *other* variable is a support, so $N = O(d^{r-1})$. acceptable is a call to check the validity of a tuple, so we have $O(d^{r-1})$ such calls per branch per variable-value pair. With $O(dr)$ pairs we need time to check $O(rd^r)$ tuples per branch. □

Theorem 22. *In a downwards-explored search tree, the total number of calls to* acceptable *made by successful calls to* FNE-MiddleOut *in an LBS is no more than $2N$.*

Proof. As in Theorem 10, all calls to FNE are consecutive in time and each call is either at the same node as the previous one or a descendent node. Without loss of generality, assume that $delta = +1$ on the first call to findNewElement in an LBS.

The value of *last* is monotonically increasing in an LBS until FNE-MiddleOut-Helper fails. Before the first failure there can be at most N calls to acceptable. At the first failed call to FNE-MIDDLEOUT-HELPER, we will have $last = k$ for some $0 \leq k < N$, and all indices $k \leq i < N$ in LIST must be unacceptable. After the failure the value of *delta* is swapped to -1 and the next sequence of calls to FNE-MIDDLEOUT-HELPER have *last* monotonically decreasing until a second failed call to FNE-MIDDLEOUT-HELPER and 0 is reached. There can be at most k more calls to ACCEPTABLE before this happens, and if it happens then all values $0 \leq i < k$ must be unacceptable. But then all values $0 \leq i < N$ are unacceptable so any future call to FNE-MIDDLEOUT must fail. Therefore the maximum number of calls to ACCEPTABLE in successful calls is N before the first failed call to FNE-MIDDLEOUT-HELPER, then $N - k$, and finally k, for a maximum of $2N$ as required. □

Proposition 24. *In a downwards-explored search tree, the total number of calls to* ACCEPTABLE *made by* FNE-MIDDLEOUT-FIXED *can be* $\Omega(N^2)$ *per branch of the search tree.*

Proof. First consider a single branch. Suppose that $last = N - 1$, and that at the leaf node all elements of LIST are initially acceptable but become unacceptable in decreasing order. The first call to FNE-MIDDLEOUT-FIXED calls FNE-MIDDLEOUT-HELPER, which fails after 0 calls to ACCEPTABLE, then *delta* is reversed and the next call succeeds, setting $last = N - 2$ after 1 call to ACCEPTABLE. Successive calls to FNE-MIDDLEOUT-FIXED require 1 call to ACCEPTABLE forwards plus 1 backwards, then 2 forwards and 1 backwards, $\ldots$, then $N - 1$ forwards and 0 backwards. This is $\Omega(N^2)$ calls and finally $last = 0$.

Now suppose at the next leaf node, all values are unacceptable except $last = N - 1$. With a further $N - 1$ calls to ACCEPTABLE we will have $last = N - 1$. Therefore these two branches have taken an average of $(\Omega(N^2) + \Omega(N))/2 = \Omega(N^2)$ per branch. Further, the preconditions for the first branch have been re-established, so this behaviour can repeat across the whole tree for $\Omega(N^2)$ calls to ACCEPTABLE per branch of the tree. □

Theorem 25. *Suppose that* W *pointers* $\mathrm{last}_1, \mathrm{last}_2, \ldots \mathrm{last}_W$ *to the same list are maintained simultaneously, with the same definition of acceptability, and that calls to* FNE *for a pointer* last_i *are made only when it points to an unacceptable element or to the same value as another pointer currently has. Then: if more than* cN *calls to* ACCEPTABLE *are made in any LBS in a downwards-explored search tree, either at least one of the calls to* FNE-CIRCULAR *fails or at least two of the pointers take the same value.*

Proof. If at least $WN + 1$ calls to ACCEPTABLE are made in an LBS then at least one element is checked $W + 1$ times. Say this is LIST$[i]$. By the pigeonhole principle, at least one $last_a$ calls ACCEPTABLE(LIST, i) twice in the LBS. As in Theorem 10, the value of $last_a$ was incremented at least N times. Now consider each element LIST$[j]$. At some point in the LBS, we had $last_a = j$ but it was incremented. This happened only because either LIST$[j]$ was unacceptable, or at that time some other pointer $last_b = j$. In the latter case, either LIST$[j]$ is now unacceptable or there is at least one pointer $d \neq a$ now with $last_d = j$. The reason is that when only one pointer remains with $last = j$ then by hypothesis it can be moved only because it becomes unacceptable. Since this applies to any list element LIST$[j]$, every list element is either unacceptable or equal to $last_d$ for some $d \neq a$. Therefore there are at most $c - 1$ acceptable elements in the list. So if every call to FNE-CIRCULAR is successful, at least two of the pointers must take the same value. □