

# Verified multi-word compare-and-set and software transactional memory for OCAML 5

ANONYMOUS AUTHOR(S)

1 INTRODUCTION

2 OVERVIEW OF THE LIBRARY

3 IMPLEMENTATION OF THE LIBRARY

4 MULTI-WORD COMPARE-AND-SET ALGORITHM

5 BENCHMARKS

6 VERIFICATION OF MULTI-WORD COMPARE-AND-SET ALGORITHM

7 RELATED WORK

8 CONCLUSION

```
50 type 'a loc =
51   { atomic state: 'a state;
52     id: int;
53   }
54 and 'a state =
55   { casn: 'a casn;
56     mutable before: 'a;
57     mutable after: 'a;
58   }
59
60 and 'a cas =
61   { loc: 'a loc;
62     state: 'a state;
63   }
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
```

```
and 'a casn =
  { atomic status: 'a status;
    proph: (ghost_id * bool) proph;
  }
and 'a status =
  | Undetermined of 'a cas list
  | Before
  | After
```

Fig. 1. Type definitions for implementing multi-word compare-and-set

```

99 1 let status_to_bool = function
100 2 | Undetermined _ -> assert false
101 3 | Before -> false
102 4 | After -> true
103 5 let finish gid casn status =
104 6   match casn.status with
105 7   | Before -> false
106 8   | After -> true
107 9   | Undetermined _ as old_status ->
108 10    resolve (
109 11      Atomic.Loc.compare_and_set [%atomic.loc casn.status] old_status status
110 12    ) casn.proph (gid, status_to_bool status) |> ignore ;
111 13    casn.status == After
112 14
113 15 let rec determine_as casn cass =
114 16   let gid = ghost_id in
115 17   match cass with
116 18   | [] ->
117 19     finish gid casn After
118 20   | cas :: cass' ->
119 21     let { loc; state } = cas in
120 22     let state' = loc.state in
121 23     if state == state' then
122 24       determine_as casn cass'
123 25     else
124 26       let v = get_as state' in
125 27       if get_as state' != state.before then
126 28         finish gid casn Before
127 29       else
128 30         match casn.status with
129 31         | Before -> false
130 32         | After -> true
131 33         | Undetermined _ ->
132 34           if Atomic.Loc.compare_and_set [%atomic.loc loc.state] state' state
133 35           then determine_as casn cass'
134 36           else determine_as casn cass
135 37 and get_as state =
136 38   if determine state.casn then state.after else state.before
137 39 and determine casn =
138 40   match casn.status with
139 41   | Before -> false
140 42   | After -> true
141 43   | Undetermined cass -> determine_as casn cass
142 44
143 45
144 46
145 47

```

Fig. 2. Implementation of multi-word compare-and-set (1)

```

148 1 let make v id =
149 2   let _gid = ghost_id in
150 3   let casn = { status= After; proph= proph } in
151 4   let state = { casn; before= v; after= v } in
152 5   Atomic.make { state; id }
153 6
154 7 let get loc =
155 8   get_as loc.state
156 9
157 10 let cas cass =
158 11  let casn = { status= After; proph= proph } in
159 12  let cass =
160 13    Lst.map cass (fun (loc, before, after) ->
161 14      let state = { casn; before; after } in
162 15      { loc; state }
163 16    )
164 17  in
165 18  casn.status <- Undetermined cass ;
166 19  determine_aux casn cass
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196

```

Fig. 3. Implementation of multi-word compare-and-set (2)

REFERENCES

197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245