

Helena 3.0

Example 1 - The distributed database system

Sami Evangelista - (Sami [dot] Evangelista [at] lipn.univ-paris13 [dot] fr)

November 29, 2017

We consider in this system a set of N database managers which communicate to maintain consistent replica of a database. It is a well-known and recurrent example of the colored Petri nets literature, initially presented by Genrich and later by Jensen.

When a manager updates its local copy of the database, he sends requests to other managers for updating their local copy (transition *Update*). As soon as a manager receives such a request (transition *Receive*) he starts the update of its copy. Its update finished, each manager acknowledges the initiating manager (transition *Send ack*). This process finishes when the initiating manager collects all the acknowledgments (transition *Receive acks*). Managers can be either *Inactive*, either *Waiting* for acknowledgments, either *Performing* an update. Places *Msgs*, *Received*, *Acks* and *Unused* model communication channels between sites. Thus, $N \cdot (N - 1)$ tokens are distributed upon these places at each marking. At last the correctness of the protocol is ensured by place *Mutex* which guarantees that two managers cannot concurrently update their local copy.

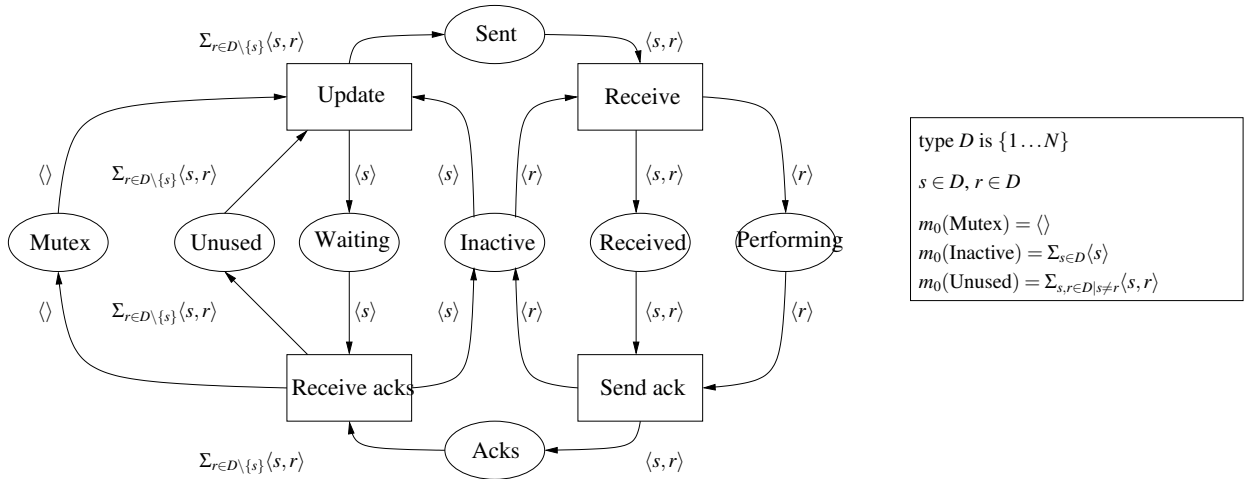


Figure 1: The distributed database system

Listing 1: Helena file of the distributed database system (file examples/dbm.lna)

```

1  /* *****
2  *
3  *   Example file of the Helena distribution
4  *
5  *   File   : dbm.lna
6  *   Author : Sami Evangelista
7  *   Date   : 27 oct. 2004
8  *   Source :
9  *       Coloured Petri Nets: A high level language for system design and analysis
10 *       In Application and Theory of Petri Nets, p.342--416, Springer, 1989
11 *       Kurt Jensen
12 *
13 *   If symbol UNUSED is defined, the model includes the place unused.
14 *
15  ***** */

```

```

16
17 dbm (N := 10) { /* N = number of sites */
18
19     type site_id : mod N;
20
21     /*
22     * process places modelling the control flow of processes
23     */
24     place inactive {
25         dom : site_id;
26         init : for(s in site_id) <( s )>;
27         capacity : 1;
28         type: process;
29     }
30     place waiting {
31         dom : site_id;
32         capacity : 1;
33         type: process;
34     }
35     place performing {
36         dom : site_id;
37         capacity : 1;
38         type: process;
39     }
40
41     /*
42     * places modelling communication channels
43     */
44     place sent {
45         dom : site_id * site_id;
46         capacity : 1;
47         type: buffer;
48     }
49     place received {
50         dom : site_id * site_id;
51         capacity : 1;
52         type: buffer;
53     }
54     place acks {
55         dom : site_id * site_id;
56         capacity : 1;
57         type: ack;
58     }
59 #ifdef UNUSED
60     place unused {
61         dom : site_id * site_id;
62         init : for(s in site_id, r in site_id) if(s != r) <( s, r )>;
63         capacity : 1;
64         type: buffer;
65     }
66 #endif
67     place mutex {
68         dom : epsilon;
69         init : epsilon;
70         capacity : 1;
71         type: shared;
72     }
73
74     transition update_and_send {
75         in {
76             inactive : <( s )>;
77             mutex : epsilon;

```

```

78 #ifdef UNUSED
79     unused    : for(r in site_id) if(s != r) <( s, r )>;
80 #endif
81     }
82     out {
83         waiting : <( s )>;
84         sent    : for(r in site_id) if(s != r) <( s, r )>;
85     }
86 }
87 transition receive_acks {
88     in {
89         waiting : <( s )>;
90         acks    : for(r in site_id) if(s != r) <( s, r )>;
91     }
92     out {
93         inactive : <( s )>;
94         mutex    : epsilon;
95 #ifdef UNUSED
96     unused    : for(r in site_id) if(s != r) <( s, r )>;
97 #endif
98     }
99 }
100 transition receive_message {
101     in {
102         inactive : <( r )>;
103         sent     : <( s, r )>;
104     }
105     out {
106         performing : <( r )>;
107         received   : <( s, r )>;
108     }
109 }
110 transition send_ack {
111     in {
112         performing : <( r )>;
113         received   : <( s, r )>;
114     }
115     out {
116         inactive : <( r )>;
117         acks     : <( s, r )>;
118     }
119 }
120
121 /*
122 * state propositions
123 */
124 proposition site_waiting: waiting 'card > 0;
125 }

```

Listing 2: Helena file of the distributed database system properties (file examples/dbm.prop.lna)

```

1 /*
2 * a site waiting for answer will eventually leave this state
3 */
4 ltl property bounded_wait:
5     ([ (site_waiting => <> (not site_waiting)));

```