FORMALIZED MATHEMATICS
Vol. 23, No. 4, Pages 325–331, 2015
DOI: 10.1515/forma-2015-0026

DE GRUYTER OPEN
degruyter.com/view/j/forma

Event-Based Proof of the Mutual Exclusion Property of Peterson's Algorithm

Ievgen Ivanov
Taras Shevchenko National University
Kyiv, Ukraine

Mykola Nikitchenko
Taras Shevchenko National University
Kyiv, Ukraine

Uri Abraham
Ben-Gurion University
Beer-Sheva, Israel

Summary. Proving properties of distributed algorithms is still a highly challenging problem and various approaches that have been proposed to tackle it [1] can be roughly divided into state-based and event-based proofs. Informally speaking, state-based approaches define the behavior of a distributed algorithm as a set of sequences of memory states during its executions, while event-based approaches treat the behaviors by means of events which are produced by the executions of an algorithm. Of course, combined approaches are also possible.

Analysis of the literature [1], [7], [12], [9], [13], [14], [15] shows that state-based approaches are more widely used than event-based approaches for proving properties of algorithms, and the difficulties in the event-based approach are often emphasized. We believe, however, that there is a certain naturalness and intuitive content in event-based proofs of correctness of distributed algorithms that makes this approach worthwhile. Besides, state-based proofs of correctness of distributed algorithms are usually applicable only to discrete-time models of distributed systems and cannot be easily adapted to the continuous time case which is important in the domain of cyber-physical systems. On the other hand, event-based proofs can be readily applied to continuous-time / hybrid models of distributed systems.

In the paper [2] we presented a compositional approach to reasoning about behavior of distributed systems in terms of events. Compositionality here means (informally) that semantics and properties of a program is determined by semantics of processes and process communication mechanisms. We demonstrated the proposed approach on a proof of the mutual exclusion property of the Peterson's algorithm [11]. We have also demonstrated an application of this approach for

ISSN 1426–2630(Print), 1898-9934(Online)

proving the mutual exclusion property in the setting of continuous-time models of cyber-physical systems in [8].

Using Mizar [3], in this paper we give a formal proof of the mutual exclusion property of the Peterson's algorithm in Mizar on the basis of the event-based approach proposed in [2]. Firstly, we define an event-based model of a shared-memory distributed system as a multi-sorted algebraic structure in which sorts are events, processes, locations (i.e. addresses in the shared memory), traces (of the system). The operations of this structure include a binary precedence relation $\leqslant$ on the set of events which turns it into a linear preorder (events are considered simultaneous, if $e_1 \leqslant e_2$ and $e_2 \leqslant e_1$), special predicates which check if an event occurs in a given process or trace, predicates which check if an event causes the system to read from or write to a given memory location, and a special partial function "`val of`" on events which gives the value associated with a memory read or write event (i.e. a value which is written or is read in this event) [2]. Then we define several natural consistency requirements (axioms) for this structure which must hold in every distributed system, e.g. each event occurs in some process, etc. (details are given in [2]).

After this we formulate and prove the main theorem about the mutual exclusion property of the Peterson's algorithm in an arbitrary consistent algebraic structure of events. Informally, the main theorem states that if a system consists of two processes, and in some trace there occur two events e_1 and e_2 in different processes and each of these events is preceded by a series of three special events (in the same process) guaranteed by execution of the Peterson's algorithm (setting the flag of the current process, writing the identifier of the opposite process to the "turn" shared variable, and reading zero from the flag of the opposite process or reading the identifier of the current process from the "turn" variable), and moreover, if neither process writes to the flag of the opposite process or writes its own identifier to the "turn" variable, then either the events e_1 and e_2 coincide, or they are not simultaneous (mutual exclusion property).

MSC: 68M14 68W15 68N30 03B35

Keywords: distributed system; parallel computing; algorithm; verification; mathematical model

MML identifier: `PETERSON`, version: `8.1.04 5.33.1254`

The notation and terminology used in this paper have been introduced in the following articles: [4], [5], [16], [18], [19], [10], [17], and [6].

1. PRELIMINARIES

We consider values⟨true, false⟩ which extend 1-sorted structures and are systems

⟨a carrier, a true, a false⟩

where the carrier is a set, the true is an element of the carrier, the false is an element of the carrier.

Let A be a value⟨true, false⟩. We say that A is consistent if and only if

(Def. 1) the true of $A \neq$ the false of A.

Let us observe that there exists a value⟨true, false⟩ which is consistent.

A value with bool is a consistent value⟨true, false⟩. Let A be a relational structure. We say that A is strongly connected if and only if

(Def. 2) the internal relation of A is strongly connected in the carrier of A.

Let us observe that there exists a relational structure which is non empty, reflexive, transitive, and strongly connected.

A linear preorder is a reflexive, transitive, strongly connected relational structure. Let V be a value with bool. We consider events structures over V and are systems

⟨events, processes, locations, traces,

a proc-E, a trace-E, a read-E, a write-E, a val⟩

where the events constitute a non empty linear preorder, the processes constitute a non empty set, the locations constitute a non empty set, the traces constitute a non empty set, the proc-E is a function from the processes into $2^{(\text{the carrier of the events})}$, the trace-E is a function from the traces into $2^{(\text{the carrier of the events})}$, the read-E is a function from the locations into $2^{(\text{the carrier of the events})}$, the write-E is a function from the locations into $2^{(\text{the carrier of the events})}$, the val is a partial function from the carrier of the events to the carrier of V.

Let S be an events structure over V.

A process of S is an element of the processes of S.

An event of S is an element of the carrier of the events of S.

An event set of S is a subset of the carrier of the events of S.

A location of S is an element of the locations of S.

A trace of S is an element of the traces of S. From now on V denotes a value with bool, a, a_1, a_2 denote elements of the carrier of V, S denotes an events structure over V, p, p_1, p_2 denote processes of S, x, x_1, x_2 denote locations of S, t denotes traces of S, e, e_0, e_1, e_2, e_3 denote events of S, and E denotes an event set of S.

Let us consider V, S, e, and x. We say that e reads x if and only if

(Def. 3) $e \in$ (the read-E of S)(x).

We say that e writes to x if and only if

(Def. 4) $e \in$ (the write-E of S)(x).

Let us consider E. We say that E reads x if and only if

(Def. 5) there exists e such that $e \in E$ and e reads x.

We say that E writes to x if and only if

(Def. 6) there exists e such that $e \in E$ and e writes to x.

Let us consider e and t. We say that $e \in t$ if and only if

(Def. 7) $e \in$ (the trace-E of S)(t).

Let us consider p. We say that $e \in p$ if and only if

(Def. 8) $e \in$ (the proc-E of S)(p).

The value associated with event e is defined by the term

(Def. 9) (the val of S)(e).

Let us consider p and t. We say that $e \in p, t$ if and only if

(Def. 10) $e \in p$ and $e \in t$.

Let us consider x and a. We say that e writes to x the value a if and only if

(Def. 11) e writes to x and the value associated with event $e = a$.

We say that e reads from x the value a if and only if

(Def. 12) e reads x and the value associated with event $e = a$.

We say that S is process-complete if and only if

(Def. 13) for every t and e such that $e \in t$ there exists p such that $e \in p$.

We say that S is process-ordered if and only if

(Def. 14) for every p, e_1, and e_2 such that e_1, $e_2 \in p$ holds if $e_1 \leqslant e_2 \leqslant e_1$, then $e_1 = e_2$.

We say that S is rw-ordered if and only if

(Def. 15) for every x, e_1, and e_2 such that (e_1 reads x or e_1 writes to x) and (e_2 reads x or e_2 writes to x) holds if $e_1 \leqslant e_2 \leqslant e_1$, then $e_1 = e_2$.

We say that S is rw-consistent if and only if

(Def. 16) for every t, x, e, and a such that $e \in t$ and e reads x and the value associated with event $e = a$ there exists e_0 such that $e_0 \in t$ and $e_0 < e$ and e_0 writes to x and the value associated with event $e_0 = a$ and for every e_1 such that $e_1 \in t$ and $e_1 \leqslant e$ and e_1 writes to x holds $e_1 \leqslant e_0$.

We say that S is rw-exclusive if and only if

(Def. 17) for every e, x_1, and x_2, it is not true that e reads x_1 and e writes to x_2.

We say that S is consistent if and only if

(Def. 18) S is process-complete, process-ordered, rw-ordered, rw-consistent, and rw-exclusive.

One can check that there exists an events structure over V which is consistent.

A distributed system with shared memory over a set of values V is a consistent events structure over V.

2. PETERSON'S ALGORITHM

From now on D denotes a distributed system with shared memory over a set of values V, p, p_1, p_2 denote processes of D, x, x_1, x_2, f_1, f_2, t_1 denote locations of D, t denotes traces of D, e, e_0, e_1, e_2, e_3 denote events of D, and E denotes an event set of D.

Let us consider V, D, e_1, and e_2. We say that $e_1 \ll e_2$ if and only if

(Def. 19) $e_1 \leqslant e_2$ and $e_2 \not\leqslant e_1$.

The interval (e_1, e_2) yielding an event set of D is defined by the term

(Def. 20) $\{e : e_1 < e < e_2\}$.

Let us consider p and t. The (e_1, e_2) interval in (p, t) yielding an event set of D is defined by the term

(Def. 21) $\{e : e_1 < e < e_2 \text{ and } e \in p, t\}$.

Now we state the propositions:

(1) The (e_1, e_2) interval in $(p, t) \subseteq$ the interval (e_1, e_2).

(2) (i) $e_1 \leqslant e_2$, or

(ii) $e_2 \leqslant e_1$.

(3) Suppose $e \in p, t$ and $e_1 < e < e_2$. Then $e \in$ the (e_1, e_2) interval in (p, t).

(4) If $e_1 < e_2$, then $e_1 \leqslant e_2$.

(5) If $e_1, e_2 \in p$ and $e_1 < e_2$, then $e_1 \ll e_2$.

(6) If $e_1 \in p, t$ and $e_2 \in p, t$ and $e_1 < e_2$, then $e_1 \ll e_2$.

(7) If $e_1 \ll e_2$, then $e_1 < e_2$.

(8) If $e_1, e_2 \in p$, then $e_1 = e_2$ or $e_1 \ll e_2$ or $e_2 \ll e_1$.

(9) If $e_1 \leqslant e_2 \leqslant e_3$, then $e_1 \leqslant e_3$.

(10) If $e_1 \leqslant e_2 \ll e_3$, then $e_1 \ll e_3$.

(11) If $e_1 \ll e_2 \leqslant e_3$, then $e_1 \ll e_3$.

(12) If $e_1 \ll e_2 \ll e_3$, then $e_1 \ll e_3$.

Let us consider V, D, e_1, and e_2. We say that e_1 and e_2 are simultaneous events if and only if

(Def. 22) $e_1 \leqslant e_2 \leqslant e_1$.

Now we state the proposition:

(13) If e_1 and e_2 are not simultaneous events, then $e_1 \ll e_2$ or $e_2 \ll e_1$.

Let us consider V, D, p, t, e, x_1, x_2, t_1, a_1, and a_2. We say that e is a Peterson critical section with respect to p, x_1, x_2, t_1, a_1, a_2 and t if and only if

(Def. 23) there exists e_1 and there exists e_2 and there exists e_3 such that $e_1 \in p, t$ and $e_2 \in p, t$ and $e_3 \in p, t$ and $e_1 < e_2 < e_3 < e$ and e_1 writes to x_1 the value the true of V and the (e_1, e) interval in (p, t) does not write to x_1 and e_2 writes to t_1 the value a_2 and the (e_2, e) interval in (p, t) does not write to t_1 and (e_3 reads from x_2 the value the false of V or e_3 reads from t_1 the value a_1).

Let E_1 be a set. We say that E_1 are Peterson critical sections in t if and only if

(Def. 24) there exists p_1 and there exists p_2 such that for every process p of D, $p = p_1$ or $p = p_2$ and there exists f_1 and there exists f_2 and there exists t_1 such that for every e such that $e \in p_1, t$ holds e does not write to f_2 and e does not write to t_1 the value the false of V and for every e such that $e \in p_2, t$ holds e does not write to f_1 and e does not write to t_1 the value the true of V and for every e such that $e \in E_1$ holds e is a Peterson critical section with respect to p_1, f_1, f_2, t_1, the false of V, the true of V and t and e is a Peterson critical section with respect to p_2, f_2, f_1, t_1, the true of V, the false of V and t.

Now we state the propositions:

(14) Suppose e_1, $e_2 \in t$ and e_1 reads from x the value a_1 and e_2 reads from x the value a_2 and $e_1 \leqslant e_2$ and $a_1 \neq a_2$. Then there exists e such that

(i) $e \in t$, and

(ii) $e_1 \ll e \ll e_2$, and

(iii) e writes to x the value a_2.

The theorem is a consequence of (9) and (2).

(15) Main result: Mutual exclusion property of Peterson's algorithm:

If e_1, $e_2 \in t$ and $\{e_1, e_2\}$ are Peterson critical sections in t, then $e_1 = e_2$ or $e_1 \ll e_2$ or $e_2 \ll e_1$. The theorem is a consequence of (2), (5), (9), (11), (10), and (14).

References

[1] Uri Abraham. *Models for Concurrency.* Gordon and Breach, 1999.

[2] Uri Abraham, Ievgen Ivanov, and Mykola Nikitchenko. Proving behavioral properties of distributed algorithms using their compositional semantics. In *Proceedings of the First International Seminar Specification and Verification of Hybrid Systems, October 10-12, 2011, Taras Shevchenko National University of Kyiv*, pages 9–19, 2011.

[3] Grzegorz Bancerek, Czesław Byliński, Adam Grabowski, Artur Korniłowicz, Roman Matuszewski, Adam Naumowicz, Karol Pąk, and Josef Urban. Mizar: State-of-the-art and beyond. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge, editors, *Intelligent Computer Mathematics*, volume 9150 of *Lecture Notes in Computer Science*, pages 261–279. Springer International Publishing, 2015. ISBN 978-3-319-20614-1. doi:10.1007/978-3-319-20615-8_17.

[4] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**): 55–65, 1990.

[5] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1(**1**):153–164, 1990.

[6] Czesław Byliński. Some basic properties of sets. *Formalized Mathematics*, 1(**1**):47–53, 1990.

[7] K. Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addison Wesley, 1988.

[8] Ievgen Ivanov, Mykola Nikitchenko, and Uri Abraham. On a decidable formal theory for abstract continuous-time dynamical systems. In Vadim Ermolayev, Heinrich C. Mayr, Mykola Nikitchenko, Aleksander Spivakovsky, and Grygoriy Zholtkevych, editors, *Information and Communication Technologies in Education, Research, and Industrial Applications*, volume 469 of *Communications in Computer and Information Science*, pages 78–99. Springer International Publishing, 2014. ISBN 978-3-319-13205-1. doi:10.1007/978-3-319-13206-8_4.

[9] L. Lamport. On interprocess communication. Part I: Basic formalism; Part II: Algorithms. *Distributed Computing*, 1:77–101, 1986.

[10] Beata Padlewska. Families of sets. *Formalized Mathematics*, 1(**1**):147–152, 1990.

[11] G. Peterson. Myths about the mutual exclusion problem. *Information Processing Letters*, 12:1133–1145, 1981.

[12] V. Pratt. Modeling concurrency with partial orders. *International Journal of Parallel Programming*, 15:33–71, 1986.

[13] M. Raynal. A simple taxonomy for distributed mutual exclusion algorithms. *ACM SIGOPS Operating Systems Review*, 25:47–50, 1991.

[14] Tom Ridge. Peterson's algorithm in Isabelle/HOL. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.99.3484`, 2006.

[15] Tom Ridge. Operational reasoning for concurrent Caml programs and weak memory models. In Klaus Schneider and Jens Brandt, editors, *Theorem Proving in Higher Order Logics*, volume 4732 of *Lecture Notes in Computer Science*, pages 278–293. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74590-7. doi:10.1007/978-3-540-74591-4_21.

[16] Wojciech A. Trybulec and Grzegorz Bancerek. Kuratowski – Zorn lemma. *Formalized Mathematics*, 1(**2**):387–393, 1990.

[17] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.

[18] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1 (**1**):73–83, 1990.

[19] Edmund Woronowicz and Anna Zalewska. Properties of binary relations. *Formalized Mathematics*, 1(**1**):85–89, 1990.

Received August 14, 2015