Mutual Exclusion in DSM Systems: A Protocol with Vectors of Timestamps; Demonstration of Control Flow by Cause/Effect Structures

Summary

A protocol based on vectors of timestamps and devised for mutual exclusion in Distributed Shared Memory (DSM) systems is presented. Its exampled behaviour is shown as a token game in the Cause/Effect (C/E) structures model.

Key words: distributed systems, distributed shared memory, mutual exclusion.

Jel codes: L86

1. Introduction

Exclusive access to resources in parallel programming systems has found various solutions originating in aforesaid works by Dekker, unpublished but presented in (Dijkstra; 2002; Lamport 1978; 1979; Ricart, Agrawala 1981; Saxena, Rai 2003) and a number of others. With coming of real multicomputer distributed systems without central memory and clock, where cooperation or competition of computers takes place only by message passing, the problem became essentially more complicated than in case of time-sharing systems or multiprocessors with shared physical memory. This concerns especially systems with no aid of central server: the computers „negotiate” by exchanging messages through the network and only one is entitled to access a resource at a time. In this paper, a fully distributed protocol with vectors of global timestamps is presented and a flow of control during its concurrent execution by computers is shown as sequence of its states. To this end an algebraic system called Cause/Effect Structures (Czaja 1988; 2002) has been used. The protocol is intended for systems with distributed shared memory (DSM), where the local memory of each computer is uniformly accessible for all computers: DSM is treated as a union of local memories. Each computer stores the vector of global timestamps of current requests for critical section, which are being issued by the connected computers. We do not discuss in details problems of timestamp advancement and mechanism of vector clocks (cf, for instance (Saxena, Raj 2003; Kuz et al. 2016). It is assumed that such mechanisms provide current values of timestamps for the protocol described here.

The successive sections concern the following issues: ordered sets and global timestamps definition, a new distributed mutual exclusion protocol, demonstration of control flow
during its concurrent execution by computers - a token game in a c/e structure model. In Appendix, a brief outline of the algebra of c/e structures is presented.

Figure 1
The schematic structure of multicomputer system with Distributed Shared Memory

2. Ordered sets

A set $Z$ is partially ordered iff its elements are related by relation $\subseteq \subseteq Z \times Z$ satisfying:

For every $x \in Z, y \in Z, z \in Z$:

1. $x \subseteq x$ (reflexivity)
2. if $x \subseteq y$ and $y \subseteq x$ then $x = y$ (antisymmetry)
3. if $x \subseteq y$ and $y \subseteq z$ then $x \subseteq z$ (transitivity)

Moreover, if apart from (1), (2), (3):

4. $x \subseteq y$ or $y \subseteq x$ (connectivity)

then $Z$ is linearly (totally) ordered.

As usually, $x \subset y$ iff $x \subseteq y$ and $x \neq y$

3. Global timestamps revisited

Let $E(S)$ denote a set of events that may occur during activity of a distributed system $S$ and let us define a partial order relation combining two kinds of precedence of events: occurring in the same process and of message sending and reception. For $x, y \in E(S)$ two auxiliary binary relations \[ \xrightarrow{\text{process}} \] and \[ \xrightarrow{\text{message}} \] are admitted as primary notions with the meaning:

- if $x$ precedes $y$ in the same process or if $x = y$ then $x \xrightarrow{\text{process}} y$
- if $x$ is a sending message by a certain process and $y$ is a reception of this message by another process then $x \xrightarrow{\text{message}} y$
A (weak) precedence \( \sim \subseteq E(S) \times E(S) \) is the least relation satisfying:
- if \( x \xrightarrow{\text{process}} y \) or \( x \xrightarrow{\text{message}} y \) then \( x \sim y \)
- if \( x \sim y \) and \( y \sim z \) then \( x \sim z \)

Events \( x, y \) are independent (concurrent) iff neither \( x \sim y \) nor \( y \sim x \), written \( x || y \)

Relation \( \sim \) is a modified precedence introduced by Lamport (1978) but due to the reflexivity of \( \sim \), relation \( \sim \) is a partial order – contrarily to the Lamport’s version.

Since neither common (global) clock nor common (global) memory is in asynchronous distributed systems, and partially ordered events occur in real (global) time, then their precedence (\( \sim \)) should imply similar precedence between time instants of their occurrences. So, an injection mapping \( C : E(S) \to R \) (\( R \) - the set of real numbers), called a logical clock should be defined, satisfying implication \( x \sim y \Rightarrow C(x) \leq C(y) \), where values \( C(x), C(y) \) are logical (not real) time instants of events \( x, y \). To avoid absurd relationship of events to their time instants visible from outside of the system (when message reception precedes its dispatch), a compensation of processors’ local clocks is necessary. So, if a sender sends a message together with its local time of dispatch and a receiver gets it earlier according to its local time, then the receiver must put forward its clock (a time-keeping register) to the time a little later than received from the sender. This procedure ensures the implication \( x \sim y \Rightarrow C(x) \leq C(y) \). Obviously the reverse implication does not hold for some \( x, y \) if \( x || y \). The logical clock \( C \) measures the compensated time. But it may happen that \( C(x) = C(y) \) and \( x \neq y \) for some concurrent \( x, y \), so the mapping \( C \) is not one-to-one function, thus \( C \) does not establish unique representation of events by their timestamps. However if the notion of timestamp is enhanced by a number of process in which the timestamped event occurs, then events can be uniquely represented by the richer timestamps, called global. So, let the processes be numbered and let \( nr(p_i) \) be a unique number of process \( p_i \) in which the event \( x \) occurs (an event may occur in exactly one process, thus it identifies the process). A pair \( \langle C(x), nr(p_i) \rangle \) is called a global timestamp of event \( x \) and let \( \preceq \) denote a relation between global timestamps defined as \( \langle C(x), nr(p_i) \rangle \preceq \langle C(y), nr(p_j) \rangle \) iff \( C(x) < C(y) \) or if \( C(x) = C(y) \) then \( nr(p_i) \leq nr(p_j) \). Obviously \( \preceq \) is linear, the so-called lexicographic order and the one-to-one injection mapping \( G : E(S) \to R \times N \) (\( N \) - the set of natural numbers) has been established by \( G(x) = \langle C(x), nr(p_i) \rangle \), because \( G(x) = G(y) \Rightarrow x = y \) for all \( x, y \). Therefore, \( G \) establishes unique representation of events by their global timestamps. Again, the implication \( x \sim y \Rightarrow G(x) \preceq G(y) \) holds but not the reverse one.

Fig. 2 exemplifies some relationships between events and their global timestamps during a certain system run. Black and grey circles are events of send and receive message respectively.
MUTUAL EXCLUSION IN DSM SYSTEMS: A PROTOCOL...

4. Distributed mutual exclusion – a protocol with vectors of global timestamps

The global timestamps are used in a number of implementations of mechanisms in distributed systems. Let us say sometimes „timestamps” instead of „global timestamps” – for brevity. As an example, consider a new protocol implementing distributed mutual exclusion. Let us admit the following assumptions:

1. sequential computers work in parallel asynchronously and are numbered 1, 2, ..., n;
2. writing and reading to/from memory is governed by the memory manager of each computer;
3. each request to the protocol for the critical section delivers a current global timestamp;
4. computer of number \( i \) keeps vector \( \overline{r}_i = [r_{i1}, r_{i2}, ..., r_{in}] \) of variables \( r_{ik} \) allocated in its physical memory; it stores its current timestamp in the component \( r_{ii} \) when requesting for the critical section, then fetches values of components \( r_{kk} \) \( (k \neq i) \) from remaining computers and stores them in variables \( r_{ik} \) of its vector \( \overline{r}_i \). Fig. 3 depicts location of vectors of timestamps in the local memories;
5. initially all variables \( r_{ij} \) contain \( \infty \) with \( \infty > x \) for any number \( x \);
6. by \( \text{min}(\overline{r}_i) \) is denoted the least value of the components in \( \overline{r}_i \);

A computer of number \( i = 1, 2, ..., n \), using the protocol depicted as a transition graph in Fig. 4 for exclusive access to a protected resource, passes through the following states:

- \( W \) – execution of local (not critical) section
- \( B \) – import of current timestamps stored in variables \( r_{kk} \) of remaining computers; execution of \( n-1 \) assignments \( r_{ik} := r_{kk} \) \( (k \neq i) \); test of condition \( r_{ii} > \text{min}(\overline{r}_i) \)
- \( Y \) – refusal to perform critical section (waiting state)
- \( R \) – execution of critical section
- \( G \) – release of critical section

Their set:
\[ \Omega = \{ W, B, Y, R, G \} \]
Figure 3
Structure of distributed system of \( n \) computers with vectors \((i = 1, 2, \ldots, n)\) of timestamps allocated in local memories

![Diagram of distributed system structure](image)

Figure 4
The distributed mutual exclusion protocol performed by computer of number \( i \) in the cycle from request for critical section till release

\[
\begin{align*}
B & : r_{ii} := \text{global timestamp} \\
& \text{fetch values of all } r_{kk} \text{ from all computers } k \neq i \text{ and store them in } r_{ik} \\
& \text{test: } r_{ii} > \min(ri) \\
Y & : \text{wait until } r_{ii} = \min(ri) \\
W & : \text{perform local section} \\
G & : r_{ii} := \infty; \text{send } \infty \text{ to all } r_{ki} \\
R & : \text{perform critical section}
\end{align*}
\]

Denotations:
- Let \( Q_i \rightarrow Q'_i \) mean: computer of number \( i \) passes from a state \( Q_i \in \Omega \) to the next state \( Q'_i \in \Omega \) in the transition graph depicted in Fig. 4. Note that transitions \( B \rightarrow R \) and \( Y \rightarrow R \) are possible if and only if \( r_{ii} = \min(\overline{ri}) \), thus due to steady growth of global timestamp as a strictly increasing function of time, at most one computer may perform critical section at a time. A formal proof is given further.
Set of global states $\Omega'' = \Omega \times \Omega \times \ldots \times \Omega$ (the $i$th component corresponds to computer number $i$) satisfying: if $\bar{Q} = [Q_1, Q_2, \ldots, Q_n] \in \Omega''$ then $\exists i, k: (i \neq k \land Q_i = R \land Q_k = R)$.

- Initial state: $Q_{init} = [W, W, \ldots, W]$ with $r_{ij} = \infty$ for every computer $i = 1, 2, \ldots, n$.

- For $\bar{Q} = [Q_1, Q_2, \ldots, Q_n] \in \Omega''$ and $\bar{Q}' = [Q_1', Q_2', \ldots, Q_n'] \in \Omega''$ let $\bar{Q} \Rightarrow \bar{Q}'$ mean: there exists a computer number $i$ such that $Q_i \rightarrow Q_i'$ and if $\rightarrow (Q_k \rightarrow Q_k')$ then $Q_k = Q_k'$. $\bar{Q}'$ is the next global state following $\bar{Q}$.

It follows from the transition graph in Fig. 4 that for any computer of number $i$:

1. Storing a timestamp in register $r_{ij}$ proceeds only in the state $B$ of computer of number $i$; $r_{ii}$ retains this value until the transition $R \rightarrow G$ takes place.
2. Storing $\infty$ in register $r_{ij}$ and sending to $r_{ki}$ of remaining computers proceeds only in the state $G$. Thus, from point 1 follows that $r_{ii}$ decreases its value only in the state $B$.
3. Global states are exactly those reachable from the initial state $Q_{init} = [W, W, \ldots, W]$.
4. Because computation of $\text{min}(\vec{r}_i)$ in the state $B$ of computer of number $i$ takes place on completion of fetching values of $r_{kk}$ from remaining computers, the order of entering computers into the critical section does not depend on the transmission latency. This is the FIFO order (sooner requested, sooner served) due to the steady growth of the global timestamps.

Figure 5 presents an exemplary run of a four computer system with Distributed Shared Memory. This is the succession of global states:


Global timestamps, i.e., pairs of numbers, are coded by single numbers – for brevity.

Correctness of the protocol in Fig. 4, which is to assure mutually exclusive execution of critical section in distributed systems, enjoys almost straightforward verification:

**Proposition 5**: in no global state two distinct computers can perform critical section.

Proof. Let on the contrary, in a global state $\bar{Q} = [Q_1, Q_2, \ldots, Q_n] \in \Omega''$ computers of number $i$ and $k$ perform critical section. Then $r_{ii} = \text{min}(\vec{r}_i)$ and $r_{kk} = \text{min}(\vec{r}_k)$ in the local states $Q_i$ and $Q_k$ of the computers. By definition of the global timestamps $r_{ii} \neq r_{kk}$ because events of request for critical section are distinct, so, their global timestamps (i.e. values of $r_{ii}$ and $r_{kk}$) are also distinct – due to the one-to-one function $G$ (Section 3). But because of actions in the state $B$ of the protocol $r_{ik} = r_{kk}$ and $r_{ki} = r_{ii}$ hold. Since $r_{ii}$ and $r_{kk}$ are minimal in vectors $\vec{r}_i$ and $\vec{r}_k$ respectively, so, $r_{ii} \leq r_{ik}$ and $r_{kk} \leq r_{ki}$, therefore $r_{ii} \leq r_{kk}$ and $r_{kk} \leq r_{ii}$ which implies $r_{ii} = r_{kk}$ (by antisymmetry of $\leq$ - see Section 2 for definition of the order $\leq$ between global timestamps) – a contradiction!
Figure 5
Exemplary run of a system with four computers using protocol depicted in Fig. 4. Background of the computers corresponds to their local states as pictured in the protocol.

\[ \text{state 1 } [W,W,W,W] \]

\[ \text{state 2 } [B,W,B,W] \]

\[ \text{state 3 } [Y,W,R,W] \]
MUTUAL EXCLUSION IN DSM SYSTEMS: A PROTOCOL...

State 4: $[Y, W, R, B]$

State 5: $[Y, B, R, Y]$

State 6: $[Y, Y, G, Y]$
MUTUAL EXCLUSION IN DSM SYSTEMS: A PROTOCOL...

state 10
\([W,Y,W,G]\)

state 11
\([W,R,W,W]\)

state 12
\([W,G,W,W]\)
More remarks on the protocol in Fig. 4 are appropriate.

1. **Consumption of time.** In the state B, computer i requesting for critical section broadcasts message „send me value of your $r_{kk}$” to all $n-1$ remaining ($k \neq i$) computers, and waits for delivery. In the worst case the message reaches all destinations one after one as well as responses arrive one after one, which takes $2(n-1)$ transmissions. Next, in the state G, on release of critical section, the computer broadcasts $\infty$ to all $r_{ki}$ of all $n-1$ remaining computers, which takes, in the worst case, $n-1$ transmissions.

2. **Failure.** If a faulty computer $k$ permanently delivers incorrect timestamp in $r_{kk}$ to remaining computers that fetch it in the state B of the protocol, then their behaviour depends on this value. If, for instance, $r_{kk}$ is small enough to make $\min(r_{ii})$ less than $r_{ii}$ then computer $i$ enters the waiting state $Y$ and will remain there forever: a starvation! But if computer $k$ delivers to computer $i$ value of $r_{kk}$ satisfying equality $r_{ii} = \min(r_{ii})$, and after a while it delivers to computer $j$ value of $r_{kk}$ satisfying equality $r_{ij} = \min(r_{ij})$, then computer $j$ may enter the critical section before computer $i$ leaves it: violation of mutual exclusion! To solve this problem, the protocol in Fig. 4 would require suitable supplementation.

3. **Elimination of busy waiting.** To make computers perform some computation instead of useless permanent checking $r_{kk} = \min(r_{kk})$ in the state $Y$, a computer which leaves critical section (i.e. in the state $G$) might fetch values of $r_{kk}$ from remaining computers and send permission to enter critical section to the computer with minimal of these values. This would, however, increase number of transmissions.

5. **Control flow during a run of the protocol – C/E structure specification**

Flow of control during activity of the distributed mutual exclusion protocol depicted in Fig. 4 may be illustrated as a flow of tokens in a cause/effect (c/e) structure that specifies the control flow. C/E structures (Czaja 1988; 2002), is an algebraic calculus devised to specify and analyse parallel processes. Basic notions and properties of the calculus is outlined in Appendix. Pictorially, a c/e structure is a graph in which nodes (places) are named. Every name of a node is endowed with a superscript and subscript, the terms called formal polynomials, whose arguments are names of predecessors (in the superscript) and successors (in the subscript) of this node. Operators connecting arguments are „+” and „•”, called addition and multiplication, where „+” means nondeterministic choice and „•” simultaneity of receiving (in case of superscript) and sending (in case of subscript) tokens. Fig. 6 shows some transformations of an exemplary c/e structure marked with tokens: a flow of token initially residing in the place $a$. The token moves to $b$, then it „splits” and moves to $c$ and $d$ simultaneously, then one from $c$ to $e$ then back to $a$, while the other remains in $d$ forever. Alternative transformation is from $a$ to $b$ then to $e$ than back to $a$. Polynomial $\theta$ (empty) means „no successors”. Tokens flow in accordance with the rules determined by semantics of c/e structures given in Appendix.
The c/e structure in Fig. 6 may be expressed as the so-called arrow expression:

$$(a \rightarrow b) + (b \rightarrow c) + (b \rightarrow d) + (b \rightarrow e) + (c \rightarrow e) + (e \rightarrow d) + (e \rightarrow a)$$

in accordance with algebraic composition rules and notational conventions given in Appendix.

Fig. 7(a) shows a c/e structure corresponding to the protocol depicted in Fig. 4 and executed by computer number $i$. The arrows are annotated with actions to be performed when tokens traverse these arrows. The annotations (grey) are comments: they do not belong to the c/e structure’s formalism. Fig. 7(b) shows a supplemenation to (a) with a fragment responsible for directing token to the place $G[i]$ when computers other than that of number $i$ release place R (critical section).

The c/e structure in Fig. 6(b) may be expressed as the arrow expression

$$PR[i] = (W[i] \rightarrow B[i]) + (B[i] \rightarrow R) \cdot (B[i] \rightarrow y[i]) + (B[i] \rightarrow Y[i]) + (Y[i] \rightarrow G[i]) + (Y[i] \rightarrow R) \cdot (Y[i] \rightarrow y[i]) + (G[i] \rightarrow W[i]) + (R \rightarrow G[i]).$$

Therefore, flow of control in the protocol performed in parallel by $n$ computers, may be expressed as the following expression:

$$PR = \sum_{i=1}^{n} PR[i]$$

where $\Sigma$ denotes repeated summation of c/e structures.

Figure 8 shows some transformations of the marked c/e structure $PR$: an exemplary flow of control in the protocol used by two computers competing for critical section.
Figure 8
Transformations of marked c/e structure corresponding to the protocol depicted in Fig. 4 and executed by two computers
6. Appendix

Outline of Cause-Effect structures – basic notions

Let $X$ be a non-empty denumerable set. Its elements, called nodes, are counterparts of places in Petri nets. Let $\theta \notin X$ be a symbol called neutral. It will play part of neutral element for operations on formal polynomials (terms). The nodes, symbol $\theta$, operators $+$, $\cdot$, called addition and multiplication respectively, and parentheses are symbols out of which polynomials are formed as follows. Each node and symbol $\theta$ is a polynomial; if $K$ and $L$ are polynomials then $(K+L)$ and $(K\cdot L)$ are too; no other polynomials exist. Let us say “polynomials over $X$”. Their set is denoted by $\mathbb{F}[X]$. Assume stronger binding of $\cdot$ than $+$; this allows for dropping some parentheses. Addition and multiplication of polynomials is defined as follows:

- $(K+L)$
- $(K\cdot L)$

It is required that the system $\langle \mathbb{F}[X],+,\cdot,\theta \rangle$ obeys the following equality axioms for all $K,L,M \in \mathbb{F}[X]$, $x \in X$:

\[
\begin{align*}
(+ & ) \quad \theta + K = K + \theta = K \\
(\cdot & ) \quad \theta \cdot K = K \cdot \theta = K \\
(++) & \quad K + K = K \\
(\ast & ) \quad x \cdot x = x \\
(+++) & \quad K + L = L + K \\
(\ast \ast & ) \quad K \cdot L = L \cdot K \\
(++++) & \quad K + (L + M) = (K + L) + M \\
(\ast \ast \ast & ) \quad K \cdot (L \cdot M) = (K \cdot L) \cdot M \\
(\ast \ast \ast \ast & ) \quad \text{If } L \neq \theta \Rightarrow M \neq \theta \text{ then } K \cdot (L + M) = K \cdot L + K \cdot M
\end{align*}
\]

Algebraic system which obeys these axioms will be referred to as a near semi ring of formal polynomials.

A cause-effect structure (c/e structure) over $X$ is a pair $U = (C,E)$ of functions:

- $C : X \rightarrow \mathbb{F}[X]$ (cause function; nodes occurring in $C(x)$ are causes of $x$)
- $E : X \rightarrow \mathbb{F}[X]$ (effect function; nodes occurring in $E(x)$ are effects of $x$)

such that $x$ occurs in the polynomial $C(y)$ iff $y$ occurs in $E(x)$. Carrier of $U$ is the set $\text{car}(U) = \{x \in X \mid C(x) \neq \theta \lor E(x) \neq \theta\}$. $U$ is finite iff $|\text{car}(U)| < \infty$. The set of all c/e structures over $X$ is denoted by $\text{CE}[X]$. Since $X$ is fixed, write simply $\text{CE}$ - wherever this makes no confusion.

A representation of a c/e structure $U = (C,E)$ as a set of annotated nodes is $\{x_{C(x)}^{\text{car}(U)}, x_{E(x)}^{\text{car}(U)} \mid x \in \text{car}(U)\}$. $U$ is also presented as a directed graph with $\text{car}(U)$ as set of nodes labelled with objects of the form $x_{C(x)}^{\text{car}(U)}$ ($x \in \text{car}(U)$) and there is an edge (arrow) from $x$ to $y$ iff $y$ occurs in the polynomial $E(x)$. Note that in this representation, edges, although useful for the appearance of system models, are redundant: interconnection of nodes may be inferred from polynomials $C(x)$, $E(x)$. Since functions $C,E$ are total, any c/e structure comprises all the nodes from $X$, also the isolated ones (with $C(x) = E(x) = \theta$), invisible in the graphical representation. The isolated nodes make the distributivity law $(+\cdot)$ conditional.
Addition and multiplication of c/e structures

For c/e structures \( U = (C_U, E_U), V = (C_V, E_V) \) define:
\[
U + V = (C_{U+V}, E_{U+V}) = (C_U + C_V, E_U + E_V)
\]
where
\[
(C_U + C_V)(x) = C_U(x) + C_V(x) \quad \text{and} \quad (E_U + E_V)(x) = E_U(x) + E_V(x)
\]
\[
U \cdot V = (C_{U \cdot V}, E_{U \cdot V}) = (C_U \cdot C_V, E_U \cdot E_V)
\]
where
\[
(C_U \cdot C_V)(x) = C_U(x) \cdot C_V(x) \quad \text{and} \quad (E_U \cdot E_V)(x) = E_U(x) \cdot E_V(x)
\]

\( U \) is a \textit{monomial} c/e structure iff each polynomial \( C_U(x) \) and \( E_U(x) \) is a monomial, i.e. does not comprise “+”. C/e structure \( \{x^\theta_y, y^\theta_x\} \) is an arrow, denoted as \( x \rightarrow y \). The pair \((\theta, \theta)\) is a c/e structure if \( \theta \) is understood as a constant function \( \theta(x) = \theta \) for each \( x \in X \). From definition of addition and multiplication of c/e structures follows that \((\theta, \theta)\) is neutral for + and \( \cdot \). For brevity let us write \( \theta \) instead of \((\theta, \theta)\).

Evidently \( U + V \in \text{CE} \) and \( U \cdot V \in \text{CE} \) that is, in the resulting c/e structures, \( x \) occurs in \( C_{U+V}(y) \) iff \( y \) occurs in \( E_{U+V}(x) \) and the same for \( U \cdot V \). Thus, addition and multiplication of c/e structures yield correct c/e structures. The algebraic system \( \langle \text{CE}[X], +, \cdot, \theta \rangle \) is a near semi ring similar to \( \langle \text{F}[X], +, \cdot, \theta \rangle \) as states the following:

\textbf{Proposition:} For all \( U, V, W \in \text{CE}[X], x, y \in X \) the following properties hold in the algebraic system \( \langle \text{CE}[X], +, \cdot, \theta \rangle : \)

\[
\begin{align*}
(+ & \quad \theta + U = U + \theta = U \quad \text{(i)} \quad \theta \cdot U = U \cdot \theta = U \\
(++) & \quad U + U = U \quad \text{(ii)} \quad (x \rightarrow y) \cdot (x \rightarrow y) = x \rightarrow y \\
(+++ & \quad U + V = V + U \quad \text{(iii)} \quad U \cdot V = V \cdot U \\
U + (V + W) & = (U + V) + W \quad \text{(iv)} \quad U \cdot (V \cdot W) = (U \cdot V) \cdot W \\
(+) & \quad \text{If } C_U(x) \neq \theta \iff C_V(x) \neq \theta \quad \text{and} \quad E_U(x) \neq \theta \iff E_V(x) \neq \theta \quad \text{then} \quad U \cdot (V + W) = U \cdot V + U \cdot W
\end{align*}
\]

The equations follow directly from definition of c/e structures and definitions of adding and multiplying c/e structures.

Notice that the operations on c/e structures make possible to combine small c/e structures into large parallel system models.

For \( U \in \text{CE} \), define a partial order in \( \text{CE} \) by \( U \leq V \iff V = U + V \). If \( U \leq V \) then \( U \) is a substructure of \( V \); \( \text{SUB}[V] = \{ U \mid U \leq V \} \) is the set of all substructures of \( V \). For \( A \subseteq \text{CE} : V \in A \) is minimal (w.r.t. \( \leq \)) in \( A \) iff \( \forall W \in A : (W \leq V \Rightarrow W = V) \).

The crucial notion for behaviour of c/e structures is \textit{firing component}, a counterpart of transition in Petri nets. It is, however, not a primitive notion but derived from the definition of c/e structures, and is introduced regardless of any particular c/e structure:
A minimal in CE\{\theta\} c/e structure \( Q = (C_Q, E_Q) \) is a firing component iff \( Q \) is a monomial c/e structure and \( C_Q(x) = \theta \Leftrightarrow E_Q(x) \neq \theta \) for any \( x \in \text{car}(Q) \). The set of all firing components is denoted by FC, thus the set of all firing components of \( U \in \text{CE} \) is
\[
\text{FC}[U] = \text{SUB}[U] \cap \text{FC}.
\]

Following the standard Petri net notation (Petri 1966; Reisig 1985), let for \( Q \in \text{FC} \):

\[
\bullet Q = \{ x \in \text{car}(Q) \mid C_Q(x) = \theta \} \quad \text{(pre-set of } Q \text{)}
\]

\[
Q^* = \{ x \in \text{car}(Q) \mid E_Q(x) = \theta \} \quad \text{(post-set of } Q \text{)}
\]

\[
\bullet Q^* = \bullet Q \cup Q^* \quad \text{(neighbourhood of } Q \text{)}
\]

The state of c/e structure is a counterpart of marking in 1-safe Petri nets. Note however that it is not bound up to any c/e structure:

A state is a subset of the set of nodes: \( s \subseteq X \). The set of all states: \( S = 2^X \), the powerset of \( X \). A node \( x \) is active in the state \( s \) if and only if \( x \in s \) and passive otherwise. After Petri nets phrasing we say “\( x \) holds a token” when \( x \) is active.

Semantics of c/e structures is a counterpart of simple firing rule in 1-safe Petri nets:

For \( Q \in \text{FC}[U] \) and \( s, t \in S \), let \( [Q] \subseteq S \times S \) be a binary relation defined as:

\[
(s,t) \in [Q] \text{ iff } Q \subseteq s \text{ and } Q^* \cap s = \emptyset \text{ and } t = (s \setminus Q) \cup Q^*
\]

(say: \( Q \) transforms state \( s \) into \( t \)). Semantics \([U]\) of \( U \in \text{CE} \) is:

\[
[U] = \bigcup_{Q \in \text{FC}[U]} [Q]
\]

\([U]^*\) is its reflexive and transitive closure, that is, \((s,t) \in [U]^*\) iff \( s = t \) or there exists a sequence of states \( s_0, s_1, \ldots, s_n \) with \( s = s_0, t = s_n \) and \((s_j, s_{j+1}) \in [U]\) for \( j = 0, 1, \ldots, n-1 \). Say that \( t \) is reachable from \( s \) in semantics \([\bullet]\) and the sequence \( s_0, s_1, \ldots, s_n \) is called a computation in \( U \).

Note that \([U] = \emptyset\) iff \( \text{FC}[U] = \emptyset \). Behaviour of c/e structures in accordance with this semantics may be imagined as a token game: if each node in a firing component’s pre-set holds a token and none in its post-set does, then remove tokens from the pre-set and put them in the post-set.

A few immediate conclusions of above definitions are:

1. \( U_1 \leq V_1 \wedge U_2 \leq V_2 \Rightarrow U_1 + U_2 \leq V_1 + V_2 \) (monotonicity of +)
2. \( U_1 \leq V_1 \wedge U_2 \leq V_2 \Rightarrow U_1 \bullet U_2 \leq V_1 \bullet V_2 \) provided that \((U_1 + V_1) \bullet (U_2 + V_2) = U_1 \bullet U_2 + V_1 \bullet V_2 + U_1 + V_1 \bullet U_2\) (conditional monotonicity of •)
3. \( U \bullet (V + W) \leq U \bullet V + U \bullet W \) but relation \( \leq \) not always may be replaced by equality
4. If \( U \bullet (V + W) = U \bullet V + U \bullet W \) then \( V \leq W \Rightarrow U \bullet V \leq U \bullet W \)
5. \( U \leq V \Rightarrow \text{FC}[U] \subseteq \text{FC}[V] \) but converse implication not always holds
6. \( \text{FC}[U] \cup \text{FC}[V] \subseteq \text{FC}[U + V] \) but the inclusion not always may be replaced by equality
7. \( (\text{CE}, \leq) \), i.e. the set of all c/e structures partially ordered by relation \( \leq \) is a non-distributive lattice with the least element \( \theta \) and with no greatest element.
8. \( \text{FC}[U] \subseteq \text{FC}[V] \Rightarrow [U] \subseteq [V] \) but converse implication not always holds
9. $\llbracket U \rrbracket \cup \llbracket V \rrbracket \subseteq \llbracket U+V \rrbracket$ but the inclusion not always may be replaced by equality
10. $\text{FC}[U] \cup \text{FC}[V] = \text{FC}[U+V] \Rightarrow \llbracket U \rrbracket \cup \llbracket V \rrbracket = \llbracket U+V \rrbracket$ but converse implication not always holds. Note that equation $\llbracket U \rrbracket \cup \llbracket V \rrbracket = \llbracket U+V \rrbracket$ expresses compositionality of summation for c/e structures $U$ and $V$; equation $\text{FC}[U] \cup \text{FC}[V] = \text{FC}[U+V]$ states that no new firing components (except for those in $U$ and $V$) are created in their sum.

References

Kuz I., Chakravarty M.M.T., Heiser G. (2016), A course on distributed systems COMP9243.

Wzajemne wykluczenie w systemach DSM: protokół z wektorami znaczników czasu; pokaz przepływu sterowania przez struktury przyczynowo-skutkowe

Streszczenie

Przedstawiono protokół oparty na wektorach znaczników czasu i zaprojektowany dla wzajemnego wykluczenia w systemach rozproszonej pamięci dzielonej (DSM). Jego przykładowe zachowanie jest ukazane jako gra z zastosowaniem żetonów w modelu struktur przyczynowo-skutkowych.

Słowa kluczowe: systemy rozproszone, rozproszona pamięć dzielona, wzajemne wykluczenie.

Kody JEL: L86

Artykuł nadesłany do redakcji w marcu 2016 roku.
© All rights reserved

Afiliacja:
prof. dr hab. inż. Ludwik Czaja
Akademia Finansów i Biznesu Vistula
Wydział Inżynierski
ul. Stokłosy 3
02-787 Warszawa
tel.: 22 457 23 00
e-mail: l.czaja@vistula.edu.pl