УДК 004.451

**О.С. САВЕНКО, С.В. МОСТОВИЙ**

***Хмельницький національний університет, Украина***

# АЛГОРИТМ ПРОГНОЗУВАННЯ СТАНУ ПРОЦЕСІВ В ПЕРСОНАЛЬНОМУ КОМП'ЮТЕРІ

*В роботі проведений аналіз життєвого циклу процесу, виділено граничний стан, що передує стану взаємоблокування. Розроблено алгоритм виявлення процесів, що знаходяться у граничному стані і можуть потрапити у стан взаємоблокування, та алгоритм прогнозування входження процесів у стан взаємоблокування в персональному комп'ютері (ПК). Також проведено оцінку часової складності розробленого алгоритму прогнозування стану процесу в ПК.*

***Ключові слова:*** *процес, взаємоблокування, стан процесу, алгоритм прогнозування стану процесу.*

## Вступ

При виконанні задач можуть виникати такі ситуації, при яких два і більше процеси весь час знаходяться в стані блокування, очікуючи спільних ресурсів. Під процесом в даному дослідженні розумітимемо програму під час її виконання на ПК [1]. Про такі процеси говорять, що вони знаходяться в стані взаємного блокування. В даний час розглядаються можливі компромісні рішення з погляду додаткових витрат на включення засобів боротьби із взаємоблокуваннями до складу операційних систем (ОС) і очікуваної від цього ефективності. В деяких випадках витрати, які необхідно здійснити для того, щоб зробити систему вільною від взаємоблокувань, є дуже високими. Проте, в системах реального часу виникнення ситуації взаємоблокування може призвести до катастрофічних наслідків.

## 1. Постановка задачі

Актуальною задачею при організації та використанні багатозадачності є виникнення стану взаємоблокування запущених на виконання процесів [2 - 4].

Для вирішення задачі взаємоблокування процесів існує ряд методів та алгоритмів. Вони поділяються на наступні групи [2 - 5]:

1) ігнорування проблеми взаємоблокування у випадку дуже низької ймовірності виникнення взаємоблокування;

2) методи, що запобігають взаємоблокуванням шляхом створення передумов для невиконання однієї з чотирьох умов настання взаємоблокування;

3) методи виявлення взаємоблокувань, які полягають у тому, щоб завжди задовільняти запити на ресурси, коли це можливо, і періодично перевіряти систему на наявність взаємоблокувань та вирішувати проблему у випадку її настання;

4) методи уникнення взаємоблокування, що полягають у тому, щоб не задовільняти запит на ресурс, якщо його виділення може потенційно спричинити взаємоблокування.

Проте згадані вище методи та відповідні їм алгоритми мають ряд недоліків [6], що робить неможливим їх використання у сучасних ОС для ПК.

Для усунення суттєвих недоліків відомих методів та алгоритмів вирішення проблеми взаємоблокування розроблено метод прогнозування стану процесу, що враховує життєвий шлях процесу [6, 7].

## 2. Життєвий шлях процесу

Згідно [2 - 5] у стан взаємоблокування можутьпотрапляти процеси, що взаємодіють між собою у багатозадачних системах. Окрім стану взаємоблокування процеси протягом свого життєвого циклу можуть знаходитись і в інших станах, а саме стан “створений”, стан “очікуючий”, стан “виконуваний”, стан “заблокований”, стан “завершений” [8].

У стан взаємоблокування процеси потрапляють, як правило, із стану “заблокований” або стану “очікуючий”. Отже серед множини процесів можна виділити підмножину потенційних процесів, що наближаються до стану взаємоблокування. Перед входженням у стан взаємоблокування процес буде знаходитись у певному граничному стані, після якого ймовірність переходу у стан взаємоблокування буде високою. Віднесемо до робочого стану такі стани процесу: стан “створений”, стан “виконуваний”, стан “завершений”. До граничного стану віднесемо такі стани процесу: стан “заблокований”, стан “очікуючий”.

Представимо шлях, яким процес потрапляє у стан взаємоблокування, у вигляді наступної схеми (рис. 1).

© О.С. Савенко, С.В. Мостовий

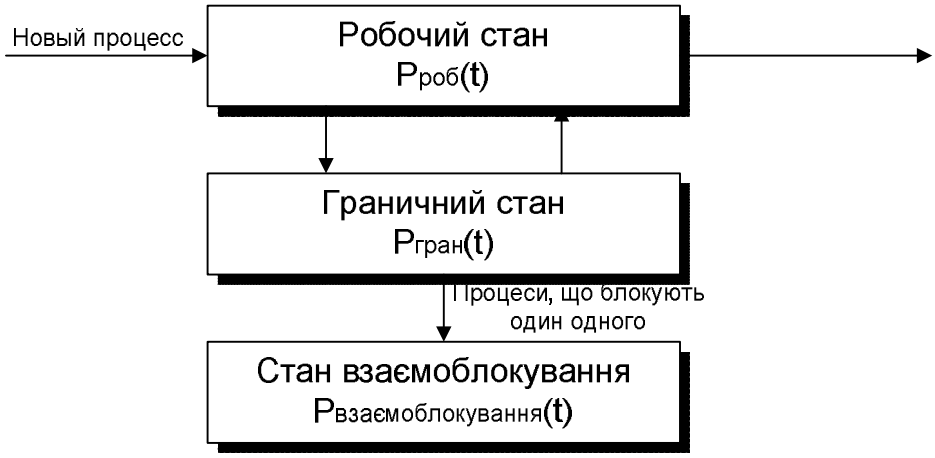


Рис. 1. Схема переходу процесів у стан взаємоблокування

Як видно із схеми, виникнення взаємоблокування процесів можливе лише для частини процесів, що знаходяться у граничному стані. При переході процесів до граничного стану відбувається зміна їхніх параметрів.

Розглянемо життєвий шлях процесу. В момент створення (стан "створений") процес знаходиться у робочому стані, йому надана частина ресурсів системи. Позначимо цей стан процесу, як $P_{роб}(t)$. В певний момент часу процесу для подальшого виконання необхідний додатковий ресурс системи, який на даний час є недоступний. Відбувається перехід процесу до стану "заблокований", тобто процес потрапляє у граничний стан. Позначимо цей стан процесу, як $P_{гран}(t)$, а перехід до даного стану, як $P_{роб}(t) \xrightarrow{\text{потреба ресурсу } s_i} P_{гран}(t)$. При задоволенні потреб процесу у ресурсах він повертається у робочий стан, тобто здійснює перехід $P_{гран}(t) \xrightarrow{\text{надання ресурсу } s_i} P_{роб}(t)$. Коли необхідний ресурс отримати неможливо по причині його використання іншим процесом, то процес залишається у граничному стані. Якщо ж другий процес в свою чергу очікує ресурс, що зайнятий першим процесом, то вони обидва потрапляють у стан взаємоблокування. Позначимо цей стан процесу, як $P_{взаємоблокування}(t)$, а перехід до даного стану, як

$$\begin{matrix} P_{гран}(t) \\ P_{взаємоблокування}(t) \end{matrix} \xrightarrow{\text{очікування ресурсу } s_i,\ \text{утримування ресурсу } s_j}.$$

## 3. Алгоритм прогнозування стану процесу

Як видно із рис.1, прогнозування стану процесу включає два етапи: визначення множини процесів, що можуть потрапити у стан взаємоблокування; виділення із множини потенційних процесів групи процесів, що потраплять у стан взаємоблокування. Таким чином, алгоритм прогнозування стану процесу буде містити дві частини.

Згідно [6] до моделі прогнозування стану процесів входять наступні величини:

$$M = \langle A, S, D, P, R \rangle,$$

де A –- множина сигнатур процесів, що виконуються на ПК у даний момент;

S – впорядкована послідовність характеристик комп'ютерної системи (загальний обсяг оперативної пам'яті, зовнішньої пам'яті, обсяг вільної в даний момент пам'яті, кількість периферійних пристроїв);

D – підмножина сигнатур процесів, що знаходяться у стані, наближеному до стану взаємоблокування (у граничному стані);

P – множина правил, на основі яких визначається група процесів, що потраплять у стан взаємоблокування;

R – вектор ймовірностей переходу у стан взаємоблокування процесів із підмножини D.

Алгоритм 1. Виявлення потенційних процесів, що можуть потрапити у стан взаємоблокування (виявлення процесів, що знаходяться у граничному стані).

1.1. Якщо $a_k \in A$ потребує ресурс $s_i \in S$, то перехід до 1.2.

1.2. Якщо $a_k \in A$ знаходиться в стані $P_{роб}(t)$, то перехід до 1.3, інакше перехід до 1.4.

1.3. Виконати для $a_k \in A$ перехід із робочого стану до граничного $P_{роб}(t) \xrightarrow{\text{потреба ресурсу } s_i} P_{гран}(t)$ та включити $a_k \in A$ до $D \subset A$. Перехід до 1.4.

1.4. Якщо $a_k \in A$ надано у використання ресурс $s_i \in S$, то перехід до 1.5, інакше перехід до 1.6.

1.5. Виконати для $a_k \in A$ перехід із граничного стану до робочого $P_{гран}(t) \xrightarrow{\text{надання ресурсу } s_i} P_{роб}(t)$ та виключити $a_k \in A$ із $D \subset A$. Перехід до 1.6.

1.6. Якщо перевірено всю множину А, то перехід до 1.7, інакше перехід до 1.1.

1.7. Кінець алгоритму 1.

Для визначення процесів, що потраплять у стан взаємоблокування, використовується система прогнозування стану процесів, в основі якої лежить система нечіткого логічного висновку (СНЛВ) [7].

На рис.2 показана структура системи нечіткого логічного висновку.

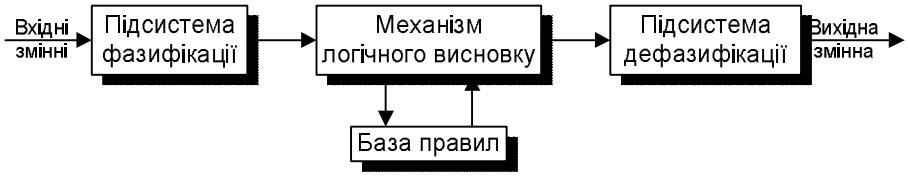


Рис. 2. Загальна схема СНЛВ

Підсистема фазифікації призначена для визначення ступеня приналежності вхідних значень $x_g \in X$, $X = D \cup S$, $g = \overline{1,h}$ до нечітких множин входу, що являють собою лінгвістичні змінні з відповідної лінгвістичної шкали

$$T_{x_g} = \left\{ T_{x_g}^1, T_{x_g}^2, ..., T_{x_g}^{m_{x_g}} \right\},$$

де $m_{x_g}$ – кількість лінгвістичних змінних у шкалі для g-го входу. Необхідність у фазифікації зумовлена тим, що у СНЛВ використовуються лінгвістичні правила [9].

База правил, що містить лінгвістичні правила, є основою механізму логічного висновку. Механізм логічного висновку здійснює відображення вхідних нечітких множин $T_{x_g}$ за допомогою кожного правила у вихідну $T_y$ з набору вихідних лінгвістичних змінних $T_y = \left\{ T_y^1, T_y^2, ..., T_y^{m_y} \right\}$.

Правила із множина правил $P = \left\{ P_j \right\}, j = \overline{1,n}$, що міститься в базі правил, подані у наступному форматі [9,10]:

$$P_j = \text{"якщо } x_1 \in T_{x_1} \text{ і } x_2 \in T_{x_2} \text{ ... і } x_h \in T_{x_h}, \text{ то } y_j \in T_y \text{"} \quad (1)$$

Вихідні нечіткі множини $y_j$ кожного правила об'єднуються в одну нечітку множину висновку $\tilde{y}$. Після цього підсистема дефазифікації здійснить відображення нечіткої множини висновку $\tilde{y}$ у чітке число $\bar{y}$, яке буде результатом СНЛВ для заданих вхідних значень $x_g$.

Алгоритм 2. Виявлення процесів, що потраплять у стан взаємоблокування:

2.1. Проводимо нормування показників за наступною формулою:

$$\bar{y} = 1 - \frac{P_d + P_m}{P_d * P_m}, P_d \neq 0, \quad (2)$$

де $P_n$ – черговий нормований показник;

$P_d$ – поточне значення показника в конкретній системі;

$P_m$ – максимальне значення показника в конкретній системі.

2.2. Для кожного $x_g \in X$, $X = D \cup S$, $g = \overline{1,h}$ визначаємо ступінь належності до нечітких множин входу (ступені істинності $\mu_g^j(x_g)$).

2.3. На основі ступенів істинності передумов $\mu_g^j(x_g)$ для кожного правила $P_j$, $j = \overline{1,n}$ розраховуємо ступінь його виконання $\alpha_j$ за формулою (3).

$$\alpha_j = \min\left( \mu_1^j(x_1), \mu_2^j(x_2), ..., \mu_h^j(x_h) \right). \quad (3)$$

2.4. На основі ступеню виконання $\alpha_j$ для кожного правила $P_j$, $j = \overline{1,n}$ розраховуємо результат його виконання.

2.5. На основі результату виконання кожного правила $P_j$, $j = \overline{1,n}$ визначаємо вихідну нечітку множину з усіченою функцією приналежності $\ddot{\mu}^j(y)$ за формулою (4).

$$\ddot{\mu}^j(y) = \min\left( \alpha_j, \mu^j(y) \right). \quad (4)$$

2.6. Вихідні нечіткі множини $\ddot{\mu}^j(y)$ згідно (5) агрегуємо в нечітку множину висновку $\tilde{y}$, що має функцію приналежності (6).

$$\tilde{y} = \max\left( \mu^j(y) \right), j = \overline{1,r}, \quad (5)$$

$$\mu_{\tilde{y}} = \max\left( \ddot{\mu}^1(y), \ddot{\mu}^2(y), ..., \ddot{\mu}^r(y) \right). \quad (6)$$

2.7. Приводимо до чіткості нечітку множину $\tilde{y}$ за допомогою процедури дефазифікації центроїдним методом (7)

$$\bar{y} = \frac{\int_{C_l}^{C_t} x \cdot f_{\tilde{y}}(x)\,dx}{\int_{C_l}^{C_t} f_{\tilde{y}}(x)\,dx}. \qquad (7)$$

2.8. Встановлюємо для $R_k$ значення $\bar{y}$.

2.9. Повторюємо кроки 2.1-2.8 k разів.

2.10. Кінець алгоритму 2.

## 4. Ефективність алгоритму прогнозування стану процесу

Оцінка ефективності алгоритму включає як якісний так і кількісні показники. Якісним показником є те, чи вирішує алгоритм поставлену задачу, чи ні. Кількісними показниками є час виконання та ємність алгоритму. В даному випадку критичними показниками є час, за який буде виконуватись прогнозування настання ситуації взаємоблокування, та задіяні при цьому ресурси системи. Отже, критерієм для оцінки ефективності алгоритму буде час.

Згідно загальноприйнятих підходів основним показником часової ефективності є часова складність (ЧС) – порядок складності алгоритму O(f).

Для алгоритму 1 ЧС становить O(k), де k - кількість наявних в системі процесів, оскільки k разів повторюються однакові лінійні дії.

Для алгоритму 2 ЧС становить $O(j \cdot \log(n))$, де n – кількість процесів, що знаходяться у граничному стані, j – кількість правил, що містяться у базі правил.

На рис. 3 наведена ЧС алгоритму прогнозування стану процесу для різної кількості процесів та правил. Як видно із рис. 3, складність алгоритму зростає нелінійно при збільшенні кількості процесів у системі, оскільки, на відміну від відомих алгоритмів вирішення задачі взаємоблокування, розроблений алгоритм прогнозування стану процесів використовує компоненти нечіткої логіки, що робить його гнучким у використанні.

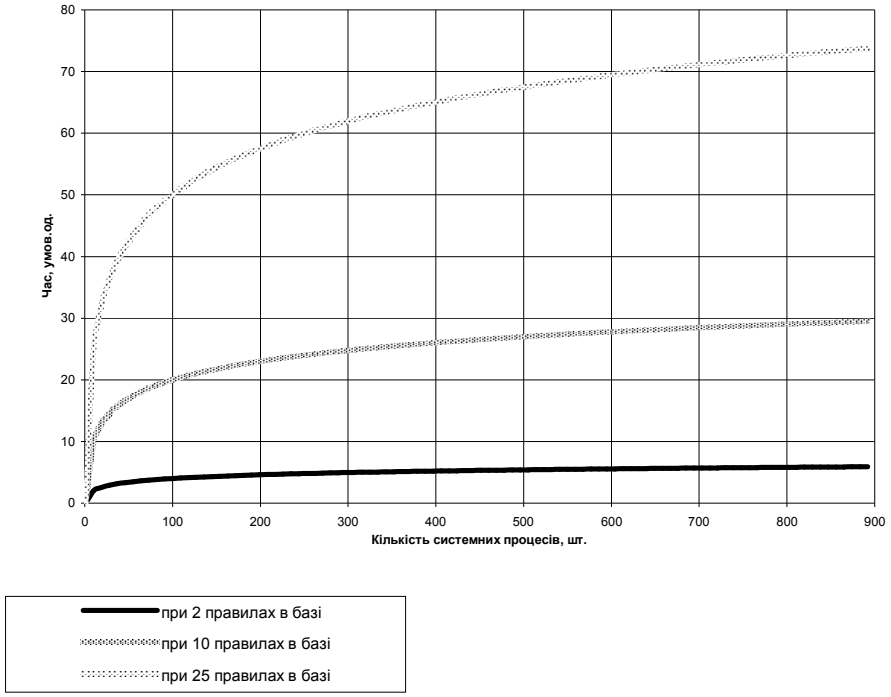


Рис. 3. Часова складність алгоритму прогнозування стану процесу

## Висновок

В роботі розроблено алгоритм прогнозування стану процесів в персональному комп'ютері, який, на відміну від відомих методів та алгоритмів, використовує компоненти нечіткої логіки. Розроблений алгоритм складається із двох частин: виявлення процесів, що знаходяться у граничному стані і можуть потрапити у стан взаємоблокування, та прогнозування входження процесів у стан взаємоблокування. Це дозволяє на першому етапі відсіяти частину процесів, які гарантовано не потраплять у стан взаємоблокування, та зменшити кількість процесів, які будуть аналізуватись на предмет потрапляння до стану взаємоблокування.

Такий підхід дозволяє виявляти два та більше процесів, що потрапляють у стан взаємоблокування і при цьому не робить алгоритм громіздким, що дозволяє використовувати його у сучасних операційних системах.

## Література

*1. ДСТУ 2941-94. – Системи оброблення інформації. Розроблення систем. Терміни та визначення.*

*2. Coffman E.G. System deadlocks / E.G. Coffman, M.J. Elphick, A. Shoshani. // Computing Surveys. – June 1971. – Vol.3, No.2. – P. 67 - 78.*

*3. Kaveh N. Deadlock detection in distribution object systems / N. Kaveh, W. Emmerich // Software Engineering Notes. – September 2001. – Vol.26, No.5. – P. 44 - 51.*

*4. Bensalem S. Confirmation of deadlock potentials detected by runtime analysis / S. Bensalem, J.-C. Fernandez, K. Havelund, L. Mounier // International Symposium on Software Testing and Analysis – 2006. – P. 41 - 50.*

*5. Isloor S.S. The Deadlock Problem: An Overview / S.S. Isloor, T.A. Marsland // Computer. – 1980. – №9, vol. 13. – P. 58-78.*

*6. Савенко О.С. Модель прогнозування стану процесів в комп'ютерній системі / О.С. Савенко, С.В. Мостовий // Радіоелектронні і комп'ютерні системи. – 2008. - №5 (32). – С. 109-115/*

*7. Система прогнозування стану процесів в персональному комп'ютері: збірник праць VIII міжнародної конференції, 14-17 травня 2008 р., Київ / відп. ред. С.В. Сирота. – К.: Просвіта, 2008. – С. 308-314/*

*8. Савенко О.С. Дослідження та аналіз блокування процесів в комп'ютерній системі / О.С. Савенко, Ю.П. Кльоц, С.В. Мостовий // Вісник ХНУ. – 2007. – № 3, т.1. – С. 248-251.*

*9. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д.Рутковская, М. Пилиньский, Л. Рутковский; перевод с польского И.Д.Рудинского. – М: Горячая линия – Телеком, 2006. – 452 с.*

*10. Леоненков А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH / Леоненков А.В. – СПб: БХВ-Петербург, 2005. – 736 с.*

*Поступила в редакцію 12.02.2009*

**Рецензент:** д-р техн. наук, проф., завідувач кафедри системного програмування, В.М. Локазюк, Хмельницький національний університет, Хмельницький, Україна.

### АЛГОРИТМ ПРОГНОЗИРОВАНИЯ СОСТОЯНИЯ ПРОЦЕССОВ В ПЕРСОНАЛЬНОМ КОМПЬЮТЕРЕ

***О.С. Савенко, С.В. Мостовой***

В работе проведен анализ жизненного цикла процесса, выделено граничное состояние, которое предшествует состоянию взаимоблокировки. Разработан алгоритм выявления процессов, которые находятся в граничном состоянии и могут попасть в состояние взаимоблокировки, и алгоритм прогнозирования вхождения процессов в состояние взаимоблокировки в персональном компьютере (ПК). Также проведена оценка временной сложности разработанного алгоритма прогнозирования состояния процесса в ПК.

**Ключевые слова**: процесс, взаимоблокировка, состояние процесса, алгоритм прогнозирования состояния процесса.

### ALGORITHM OF FORECASTING OF A STATUS OF PROCESSES IN THE PERSONAL COMPUTER

***O.S. Savenko, S.V. Mostovoy***

In work the analysis of life cycle of process is conducted, the boundary status is chosen which will precede to a status of deadlock. The algorithm of revealing of processes is developed which are in a boundary status and can hit in a status of deadlock, and algorithm of forecasting of entry of processes in a status of deadlock in the personal computer (PC). An evaluation of temporary complexity of the developed algorithm of forecasting of a status of process in the PC also is conducted.

**Key words**: process, deadlock, status of process, algorithm of forecasting of a status of process.

**Савенко Олег Станіславович** - канд. техн. наук, доцент, декан факультету комп'ютерних систем та програмування, Хмельницький національний університет, Хмельницький, Україна, e-mail: kism@beta.tup.km.ua.

**Мостовий Сергій Володимирович** - асистент кафедри системного програмування, Хмельницький національний університет, Хмельницький, Україна, e-mail: ks99@datasvit.km.ua.