інженерія. – 2007. – № 3 (10). – С. 131–139.

6. Шиян А.А. Теоретико-ігровий аналіз раціональної поведінки людини та прийняття рішень в управлінні соціально-економічними системами : [монографія] / Шиян А.А. – Вінниця : УНІВЕРСУМ-Вінниця, 2009. – 404 с.

7. Новиков Д.А. Рефлексивные игры / Д.А. Новиков, А.Г. Чхартишвили. – М. : СИНТЕГ, 2003. – 149 с.

8. Чхартишвили А.Г. Теоретико-игровые модели информационного управления / Чхартишвили А.Г. – М. : ЗАО «ПМСОФТ», 2004. – 227 с.

9. Шиян А.А. Математичне моделювання спільної економічної діяльності людей / А.А. Шиян // Наукові праці ВНТУ. – 2008. – № 2. – 7 с. [Едектронний ресурс]. – Режим доступу : http://www.nbuv.gov.ua/e-journals/VNTU/2008-2/2008-2.files/uk/08aasaop_uk.pdf .

10. Шиян А.А. Метод обрахунку ефективності спільної економічної діяльності людей на основі теоретико-ігрового моделювання / А.А. Шиян, Л.О. Нікіфорова // Вестник национального технического университета «ХПИ». – 2008. – № 54 (2). – С. 10–13.

11. Bolton P. Contract Theory / P. Bolton, M. Dewatripont. – Cambridge: MIT Press, 2005. – 724 p.

Надійшла 14.6.2012 р.
Рецензент: д.т.н. Штовба С.Д.

УДК 004.451

С.В. МОСТОВИЙ
Хмельницький національний університет

ВЗАЄМОБЛОКУВАННЯ ПРОЦЕСІВ В КОМП'ЮТЕРНИХ СИСТЕМАХ

В роботі проведений аналіз життєвого циклу процесу, виділено граничний стан, що передує стану взаємоблокування. Розроблено алгоритм прогнозування потрапляння процесів у стан взаємоблокування.

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 forecasting of deadlock of processes is developed.

Ключові слова: процес, стан процесу, взаємоблокування, прогнозування взаємоблокування.

Вступ

Вагома частка взаємоблокувань процесів припадає на взаємоблокування процесів, що виконуються в комп'ютерних системах (КС).

Під комп'ютерною системою будемо розуміти сукупність програмно-апартних засобів, призначених для розв’язання поставленої задачі, а саме персональні комп'ютери (ПК) та відповідне програмне забезпечення (як системне, так і прикладне).

В результаті дослідження відомих методів та алгоритмів уникнення взаємоблокувань процесів в операційних системах (ОС) [1, 2] виявлено, що вони не в повному обсязі розв’язують поставлену задачу. Не всі алгоритми придатні до реалізації у сучасних операційних системах [3, 4], оскільки мають ряд недоліків (блокування роботи ОС, наявність циклів активного очікування, складність програмної реалізації для багатьох процесів, необхідність використання спеціалізованої команди процесора) і є складними для реалізації. Тому більшість сучасних ОС не містять ефективних засобів для вирішення проблеми взаємоблокування процесів [5].

Проте масштабність задач, які розв’язуються за допомогою персонального комп’ютера, зростає і це вимагає розв’язання задачі уникнення взаємоблокування процесів. Тому задача розробки нових методів та засобів, які б дозволили прогнозувати входження процесів у стан взаємоблокування є актуальною. Для розробки таких методів та засобів необхідно дослідити життєвий цикл процесів КС та ивзначити при яких умовах може відбутись взаємоблокування процесів.

Основна частина

Життєвий цикл процесів комп'ютерної системи

Багатозадачність (англ. multitasking) – властивість операційної системи або середовища програмування забезпечувати можливість паралельної (або псевдопаралельної) обробки декількох процесів [6]. Дійсна багатозадачність операційної системи можлива тільки в розподілених обчислювальних системах.

Процес – це система дій, що реалізує певну функцію в обчислювальній системі й оформлена так, що керуюча програма обчислювальної системи може перерозподіляти ресурси цієї системи з метою забезпечення багатозадачності [6]. Позначимо множину виконуваних процесів, як $A = \{a_i\}_{i=1}^{y}$, де у – кількість процесів.

Ресурс обчислювальної системи – засіб обчислювальної системи, що може бути виділений процесу обробки даних на певний інтервал часу [6]. Позначимо множину наявних ресурсів ОС, як $RE = \{re_j\}_{j=1}^{x}$, де x – кількість видів ресурсів.

До ресурсів ОС віднесемо наявну пам'ять, процесори, пристрої вводу/виводу, а також дані,

необхідні для роботи процесів (файли в пам'яті та на зовнішніх носіях, результати обчислень інших процесів).

Кожен процес від моменту створення до моменту завершення проходить через ряд станів (рис. 1). Проте така діаграма станів не відображають стан взаємоблокування процесів.

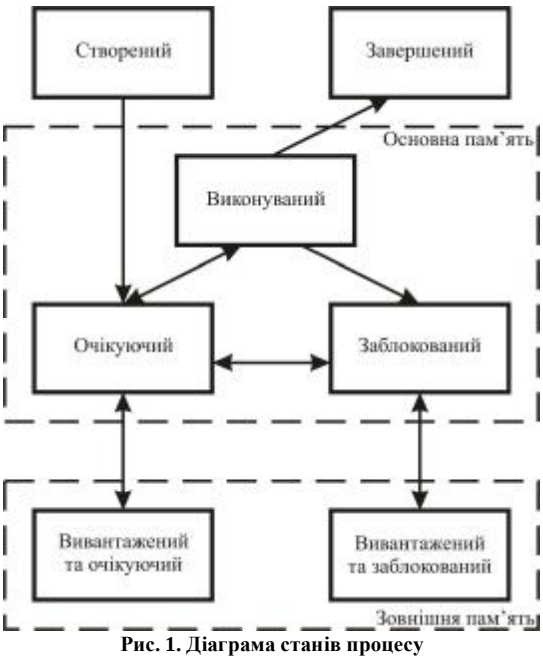

Рис. 1. Діаграма станів процесу

Під сигнатурою процесу будемо розуміти сукупність його характеристик, яка однозначно ідентифікує стан процесу в ОС в певний момент часу t:

$$a_i(t) \to (a_{i_1}^t, a_{i_2}^t, ..., a_{i_z}^t), \quad (1)$$

де $a_i(t) \in A$ – поточний процес, $a_{i_1}^t, ..., a_{i_z}^t$ – характеристики процесу в поточний момент часу (параметри та ресурси, які використовує процес в даний момент).

До характеристик процесу, що формують сигнатуру, віднесемо наступні: ідентифікатор процесу, ідентифікатор батьківського процесу, ідентифікатор користувача, якому належить процес, пріоритет процесу, квоти процесу (кількість пам'яті і процесорний час доступні процесу), дескриптори відкритих процесом файлів [6].

Оскільки до складу сигнатури процесу входять унікальні характеристики, то в один і той самий момент часу в системі не існує двох абсолютно однакових сигнатур.

Отже, життєвий цикл процесу можна подати у вигляді послідовності станів, через які проходить цей процес. Перехід зі стану в стан відбувається через зміну певних параметрів, якими характеризується процес. Зміна параметрів процесу відбувається з ряду причин: дії ОС, дії інших процесів, виконання власного програмного коду. Стан кожного окремого процесу буде впливати на стан КС в цілому.

Позначимо стан процесу через w, причину зміни параметрів через r, а перехід із стану в стан через $w_i \xrightarrow{\text{зміна параметрів з причини } r_j} w_{i+1}$. Тоді життєвий цикл процесу буде мати вигляд (2):

$$a_i : w_0 \xrightarrow{r_j} w_1 \xrightarrow{r_j} ... \xrightarrow{r_j} w_{k-1} \xrightarrow{r_j} w_k, \quad (2)$$

де $w_0 \in W$ – початковий стан процесу (стан "створений"), $w_k \in W$ – кінцевий стан процесу (стан "завершений"), W – множина програмних станів процесу (рис.1), $r_j \in R$ – множина можливих причин зміни параметрів процесу (j=1,2,3…).

Враховуючи визначення сигнатури та (1) і (2), життєвий цикл кожного процесу можна подати у вигляді:

$$a_i : (a_{i_1}^{t_0}, a_{i_2}^{t_0}, ..., a_{i_z}^{t_0}) \xrightarrow{r_j} (a_{i_1}^{t_1}, a_{i_2}^{t_1}, ..., a_{i_z}^{t_1}) \xrightarrow{r_j} (a_{i_1}^{t_{k-1}}, a_{i_2}^{t_{k-1}}, ..., a_{i_z}^{t_{k-1}}) \xrightarrow{r_j} (a_{i_1}^{t_k}, a_{i_2}^{t_k}, ..., a_{i_z}^{t_k}). \quad (3)$$

У стан взаємоблокування можуть потрапляти процеси, що взаємодіють між собою у багатозадачних ОС в певний момент часу. До потрапляння у стан взаємоблокування процеси протягом свого життєвого циклу знаходяться в інших станах, а саме стан "створений", стан "очікуючий", стан "виконуваний", стан "заблокований", стан "завершений" (рис. 2). У стан взаємоблокування процеси потрапляють, як правило, із стану "заблокований" або стану "очікуючий". Отже, у даний момент часу серед множини процесів можна виділити підмножину процесів, які можуть в наступний момент часу потрапити до стану взаємоблокування. Перед входженням у стан взаємоблокування процес буде знаходитись у певному "граничному" стані [7], після якого ймовірність переходу у стан взаємоблокування буде високою. Взаємоблокування процесів призводить до часткової або повної втрати функційної здатності ОС. Тому вважатимемо стан взаємоблокування процесів неробочим станом ОС, а інші стани процесів – робочим станом ОС.

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. 3).

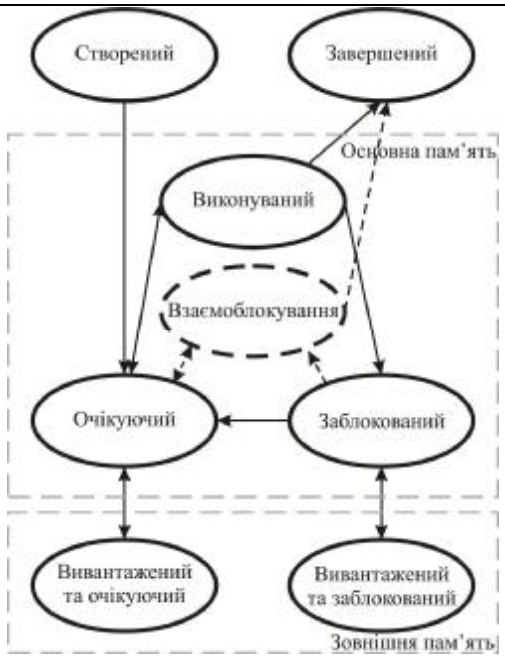

Рис. 2. Діаграма станів процесу, що включає стан взаємоблокування

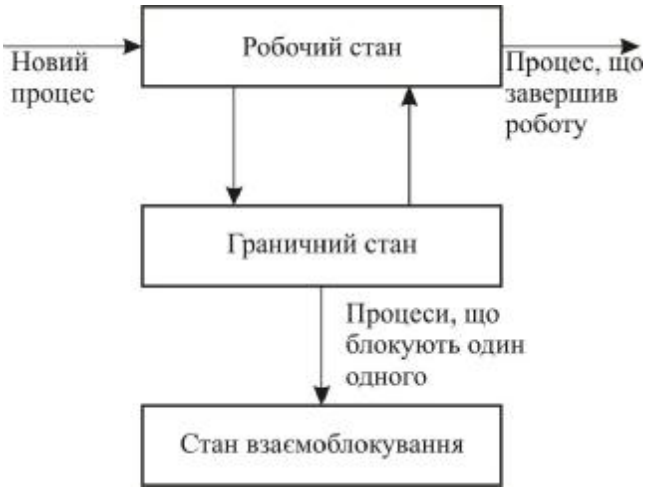

Рис. 3. Схема переходу процесів у стан взаємоблокування

Як видно зі 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 цикл процесу. В момент створення (стан "створений") процес знаходиться у робочому стані., йому надана частина ресурсів системи. Позначимо цей стан процесу, як $s_{\text{роб}}(t)$. В певний момент часу процесу для подальшого виконання необхідний додатковий ресурс системи, який на даний час є недоступним. Відбувається перехід процесу до стану "заблокований", тобто процес потрапляє у граничний стан. Позначимо цей стан процесу, як $s_{\text{гран}}(t)$, а перехід до даного стану, як $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$. При задоволенні потреб процесу у ресурсах він повертається у робочий стан, тобто здійснює перехід $s_{\text{гран}}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{\text{роб}}(t)$. 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 $s_{\text{взаємоблокування}}(t)$, а перехід до даного стану, як $s_{\text{гран}}(t) \xrightarrow[\text{утримування ресурсу } re_j]{\text{очікування ресурсу } re_i} s_{\text{взаємоблокування}}(t)$.

Таким чином, враховуючи (3), життєвий цикл процесу буде мати один з наступних виглядів:

1) Процес створений, йому надано всі необхідні для завершення роботи ресурси, він виконується і завершується (процес весь час знаходиться в робочому стані $s_{\text{роб}}(t)$), тобто:

$$a_i : s_0 \xrightarrow{r_j} s_1 \xrightarrow{r_j} s_k, \tag{4}$$

де $s_0 \in s_{\text{роб}}$, $s_1 \in s_{\text{роб}}$, $s_k \in s_{\text{роб}}$.

2) Процес створений, йому надано частину ресурсів, необхідних для завершення роботи, він потребує додаткових ресурсів, через деякий час отримує їх, виконується і завершується (процес спочатку знаходиться в робочому стані $s_{\text{роб}}(t)$, потім переходить $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$ в граничний стан $s_{\text{гран}}(t)$, потім $s_{\text{гран}}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{\text{роб}}(t)$ знову в робочий стан $s_{\text{роб}}(t)$), тобто:

$$a_i : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_l \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_k, \tag{5}$$

де $s_0 \in s_{\text{роб}}$, $s_l \in s_{\text{гран}}$, $s_k \in s_{\text{роб}}$.

3) Процес створений, йому надано частину ресурсів, необхідних для завершення роботи, він потребує додаткових ресурсів, які утримує інший процес, який в свою чергу потребує ресурсів першого процесу (процес спочатку знаходиться в робочому стані $s_{\text{роб}}(t)$, потім переходить

$s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$ в граничний стан $s_{\text{гран}}(t)$, із якого відбувається перехід $s_{\text{гран}}(t) \xrightarrow[\text{утримування ресурсу } re_j]{\text{очікування ресурсу } re_i} s_{\text{взаємоблокування}}(t)$ до стану взаємоблокування).

$$\begin{cases} a_i : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_l \xrightarrow{r_j} s_x \\ a_m : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_n \xrightarrow{r_j} s_x \end{cases}, \tag{6}$$

де $s_0 \in s_{\text{роб}}$, $s_l \in s_{\text{гран}}$, $s_n \in s_{\text{гран}}$, $s_x \in s_{\text{взаємоблокування}}$.

Із розглянутих вище можливих варіантів поведінки процесів критичним для ОС є останній, за якого процеси потрапляють у стан взаємоблокування

Алгоритм прогнозування потрапляння процесів у стан взаємоблокування

Як видно з рис. 3, 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 з [8] до моделі прогнозування стану процесів входять наступні величини:

$$M = \langle A,S,D,P,R \rangle, \tag{7}$$

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 в стан взаємоблокування процесів з підмножини D.

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

1. Якщо $a_k \in A$ потребує ресурс $r_i \in R$, то перехід до 2.

2. Якщо $a_k \in A$ знаходиться в стані $s_{\text{роб}}(t)$, то перехід до 3, інакше перехід до 4.

3. Виконати для $a_k \in A$ перехід з робочого стану до граничного $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$ та включити $a_k \in A$ до $D \subset A$. Перехід до 4.

4. Якщо $a_k \in A$ надано у використання ресурс $re_i \in RE$, то перехід до 5, інакше перехід до 6.

5. Виконати для $a_k \in A$ перехід з граничного стану до робочого $s_{\text{гран}}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{\text{роб}}(t)$ та виключити $a_k \in A$ із $D \subset A$. Перехід до 6.

6. Якщо перевірено всю множину А, то перехід до 7, інакше перехід до 1.

7. Кінець алгоритму.

Для визначення процесів, що потраплять у стан взаємоблокування, використовується система прогнозування стану процесів, в основі якої лежить система нечіткого логічного висновку (СНЛВ). На рис. 4 показана структура СНЛВ.

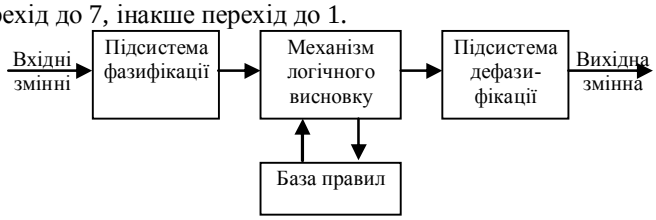

Рис. 4. Загальна схема СНЛВ

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 $T_{x_g} = \left\{ T_{x_g}^1, T_{x_g}^2, \ldots, T_{x_g}^{m_{x_g}} \right\}$, де m_{x_g} – кількість лінгвістичних змінних у шкалі для g-го входу. Необхідність у фазифікації зумовлена тим, що у СНЛВ використовуються лінгвістичні правила.

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 $T_y = \left\{ T_y^1, T_y^2, \ldots, T_y^{m_y} \right\}$. Правила з множини правил

$P = \{P_j\}$, $j = \overline{1,n}$, що міститься в базі правил, надані у наступному форматі:

$$P_j = "\text{якщо } x_1 \in T_{x_1} \text{ і } x_2 \in T_{x_2} \ldots \text{ і } x_h \in T_{x_h}, \text{то } y_j \in T_y" \quad (8)$$

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

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

1. Проводимо нормування показників за наступною формулою:

$$P_n = 1 - \frac{P_d + P_m}{P_d \cdot P_m}, P_d \neq 0, \quad (9)$$

де P_n – черговий нормований показник;

P_d – поточне значення показника в конкретній системі;

P_m – максимальне значення показника в конкретній системі.

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

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

$$\alpha_j = \min\left(\mu_1^j(x_1), \mu_2^j(x_2), \ldots, \mu_h^j(x_h)\right) \quad (10)$$

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

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

$$\mu^j(y) = \min\left(\alpha_j, \mu^j(y)\right) \quad (11)$$

6. Вихідні нечіткі множини $\mu^j(y)$ згідно (12) агрегуємо в нечітку множину висновку $\tilde{y}$, що має функцію приналежності (13).

$$\tilde{y} = \max\left(\mu^j(y)\right), j = \overline{1,r} \quad (12)$$

$$\mu_{\tilde{y}} = \max\left(\mu^1(y), \mu^2(y), \ldots, \mu^r(y)\right) \quad (13)$$

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

$$\bar{y} = \frac{\int_{C_1}^{C_t} x \cdot f_{\tilde{y}}(x)dx}{\int_{C_1}^{C_t} f_{\tilde{y}}(x)dx} \quad (14)$$

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

9. Повторюємо кроки 1–8 *k* разів.

10. Кінець алгоритму.

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 з загальноприйнятими підходами 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. 5 наведена ЧС алгоритму прогнозування стану процесу для різної кількості процесів та правил. Як видно з рис. 5, 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.

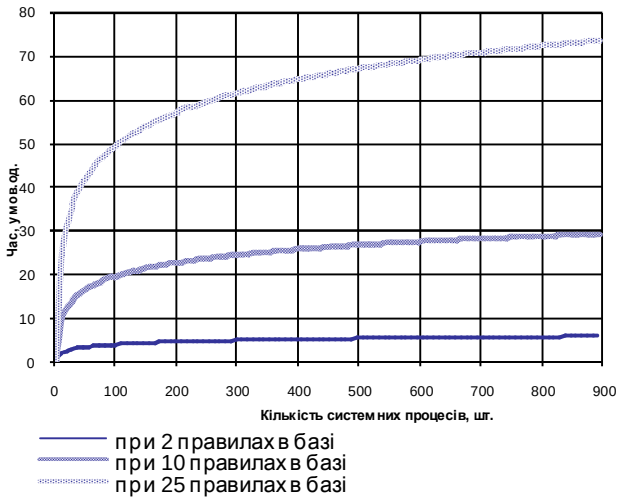

Рис. 5. Часова складність алгоритму прогнозування стану процесу

Висновки

В роботі проведений аналіз життєвого циклу процесу. В результаті дослідження виявлений "граничний" стан, який передує стану взаємоблокування процесів. Це дозволяє в поточний момент часу визначити множину процесів, які можуть потрапити в стан взаємоблокування в наступний момент часу. В подальшому необхідно визначити та дослідити умови та параметри КС, при яких процеси із виділеної множини потраплять у взаємоблокування.

Література

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

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

3. Kaveh N. Deadlock detection in distribution object systems / Nima Kaveh, Wolfgang Emmerich // Software Engineering Notes. – September 2001. – Vol.26, No.5. – Pages 44 – 51.

4. Confirmation of deadlock potentials detected by runtime analysis / Saddek Bensalem, Jean-Claude Fernandez, Klaus Havelund, Laurent Mounier // International Symposium on Software Testing and Analysis – 2006. – Pages 41 – 50.

5. 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

6. Таненбаум Э. Операционные системы: разработка и реализация. Класика CS / Э. Таненбаум, А. Вудхалл. – СПб : Питер, 2006. – 576 с.

7. Поморова О.В. Теоретичні основи, метод та засоби інтелектуального діагностування комп'ютерних систем : [монографія] / Поморова О.В. – Хмельницький : ТОВ "Тріада-М", 2006. – 253 с.

8. 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.

Надійшла 14.6.2012 р.
Рецензент: д.т.н. Шинкарук О.М.