WYŻSZA SZKOŁA ROLNICZO - PEDAGOGICZNA
w Siedlcach

Wydział Chemiczno - Matematyczny
Kierunek Informatyka

Zbigniew Niedźwiedź

Opracowanie pakietu programów umożliwiających generowanie zredukowanych przestrzeni stanów zachowujących własności wyrażalne w DESL

Praca dyplomowa
napisana pod kierunkiem
doc. dr hab. Wojciecha Penczka

Lublin 1997/98

1 Spis treści

2 Wstęp

Sieci Petriego są przydatnym narzędziem do modelowania i analizy asynchronicznych systemów współbieżnych. Znalazły one szerokie zastosowanie w modelowaniu różnych problemów z zakresu systemów operacyjnych, programowania współbieżnego, rozproszonych baz danych, systemów czasu rzeczywistego itp.

Podstawowe ograniczenia zastosowań sieci Petriego wynikają z faktu, że modele sieciowe systemów rzeczywistych są na ogół bardzo rozbudowane i zarówno ich poprawne konstruowanie, jak i sprawna analiza są praktycznie możliwe tylko wtedy, gdy są dostępne odpowiednie narzędzia automatycznie wykonujące te czynności [5].

Model Checking jest jedną z najlepszych metod automatycznej weryfikacji własności programu, dających odpowiedź na pytanie: czy program spełnia założenia (podane jako formuła w języku logiki temporalnej). Dowodem na siłę tej metody niech będzie znalezienie błędu w specyfikacji procesora Pentium właśnie za pomocą tej metody. Za jej pomocą sprawdzane są również układy scalone, protokoły komunikacyjne [8], itd.

Problemem przy automatycznej weryfikacji programów jest tzw. eksplozja stanów, powodująca, że uzyskanie pełnej przestrzeni stanów dla nietrywialnych programów jest praktycznie niemożliwe. Przy wielu niezależnych operacjach wielkość tej przestrzeni rośnie wykładniczo.

Przy pewnych ograniczeniach języka formuł można sprawdzić wiele własności przy wykorzystaniu części stanów. Niezbędne jest zatem zastosowanie heurystyki do redukcji przestrzeni stanów podczas

generowania tej przestrzeni. Uzyskuje się wtedy redukcje przestrzeni np. z wykładniczej do wielomianowej. Jednej z metod generowania zredukowanej przestrzeni stanów poświęcona jest [1], będąca podstawą niniejszej pracy.

Nie jest to jedyna metoda redukcji generowanych przestrzeni stanów. Podobne są opisane w [3] i [4], zaś inne w [9] i [10].

3 Cel pracy

Celem pracy jest uzyskanie wyników doświadczalnych (opracowanie programu) dla efektywnej redukcji generowanej przestrzeni stanów zachowujących własności wyrażalne w DESL (logika interpretowana na porządkach częściowych). Metoda osiągnięcia takiego celu została opisana w [1], [2] i [3], ale nie było dotąd implementacji tych algorytmów oraz wyników doświadczalnych.

4 Teoria

Podstawą tej części pracy jest [1] i [2]. Znajdują się tam dowody poprawności użytych twierdzeń. Sporo informacji na temat sieci można znaleźć również w [5].

1 Sieci Petriego

1. Sieć

Uporządkowaną trójkę $N = (S, T, F)$, gdzie S jest zbiorem wierzchołków, T zbiorem zdarzeń a F relacją przepływu (przejścia) nazywamy siecią, gdy spełnione są następujące warunki:

$S \cap T = \emptyset, S \cup T \neq \emptyset$

$F \subseteq (S \times T) \cup (T \times S)$

$(\forall s \in S)(\exists t \in T) : (s,t) \in F \vee (t,s) \in F$

$(\forall t \in T)(\exists s \in S) : (s,t) \in F \vee (t,s) \in F$

Jest to struktura umożliwiająca modelowanie programów współbieżnych o skończonej liczbie stanów, reprezentowana przez graf zorientowany bez wierzchołków izolowanych.

2. Zbiór poprzedników (prekondycja)

$pre(e) = \{s \in S | (s,e) \in F\}$

3. Zbiór następników (postkondycja)

$post(e) = \{s \in S | (e,s) \in F\}$

4. Konfiguracja

Podzbiór $c \subseteq S$.

5. Zajście (odpalenie) zdarzenia

Zachodzi, gdy zdarzenie $e \in T, c \subseteq S$ jest umożliwione, tj. wtw $pre(e) \subseteq c \wedge post(e) \cap c = \emptyset$.

6. Konfiguracja osiągalna

Wykonanie zdarzenia e w konfiguracji c oznaczamy $c[e > c'$. Konfiguracja c' jest osiągalna z konfiguracji c wtw, gdy istnieje skończona sekwencja zdarzeń $e_1, ..., e_{n+1}$ i konfiguracji $c_1, ..., c_n$ takich, że $c[e_1 > c_1[e_2 > ...[e_n > c_n[e_{n+1} > c'$, przy czym $c' = (c \backslash pre(e)) \cup post(e)$.

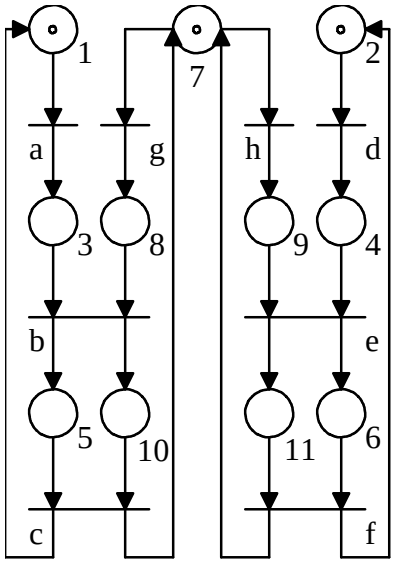
7. Sieć markowana

Sieć Petriego z określoną konfiguracją początkową $MN = (S, T, F, c_0)$, gdzie (S, T, F) - sieć, a c_0 - konfiguracja początkowa.

8. Graf konfiguracji

Graf $GK = (V, E)$, gdzie V jest zbiorem konfiguracji osiągalnych z c_0, $E \subseteq V \times T \times V$; $c \xrightarrow{e} c' \Leftrightarrow (\exists e \in E); c[e > c'$.

9. Przykład - sieć Petriego dla programu MUTEX



Sieć dla programu Mutex

W powyższym przykładzie $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$, $T = \{a, b, c, d, e, f, g, h\}$, $c_0 = \{1, 7, 2\}$.

2 Struktura zdarzeń

Niech S będzie przeliczalnym zbiorem i niech < będzie binarną relacją nad S. Odwrotność < jest oznaczona przez $<^{-1}$. Dla elementu $s \in S$ zbiór $\downarrow_<(s)$ zawiera elementy $s' \in S$ takie, że $s' < s$ oraz $\uparrow_<(s) = \{s' \in S | s < s'\}$.

Element $s \in S$ jest <-początkowy jeśli $\downarrow_<(s)$ jest pusty. Relacja < jest preskończona jeżeli $\downarrow_<(s)$ jest skończona dla wszystkich elementów $s \in S$.

Niech R^* będzie zwrotnie-przechodnim domknięciem relacji R^*. Relacja < jest zredukowana jeśli dla każdego $s < s', (s, s') \notin (< \setminus (s, s'))^*$. Jeżeli < jest zredukowana to < jest niezwrotna i jeśli $s < s'$ i $s' < s''$ to $s \not< s''$.

Struktura zdarzeń reprezentuje system współbieżny poprzez branie wystąpień akcji jako punktu startowego. Każde wystąpienie akcji jest modelowane jako osobne zdarzenie. Pozwalają to uzyskać dwie relacje: odpowiednio (natychmiastowa) przyczynowość i (natychmiastowy) konflikt między zdarzeniami.

Czwórka $(E, E_0, \angle, \#_m)$ jest (dyskretną) strukturą zdarzeń (pierwotnych), jeśli są spełnione następujące warunki:

- E jest przeliczalnym zbiorem, nazywanym zbiorem zdarzeń,
- $\angle \subseteq E \times E$ jest niezwrotną i zredukowaną relacją, nazywaną relacją natychmiastowej przyczynowości,
- $\leq = \angle^*$ jest pre-skończonym częściowym porządkiem, nazywanym relacją przyczynowości,
- $\#_m \subseteq E \times E$ jest niezwrotną i symetryczną relacją, nazywaną natychmiastową relacją konfliktu,
- $\leq \cap \# = \emptyset$ dla $\# = (\leq^{-1} \circ \#_m \circ \leq)$, nazywaną relacją konfliktu,
- $E_0 \subseteq E$ jest zbiorem $\angle$-początkowych elementów E.

Z definicji wynika, że $\# \circ \leq \subseteq \#$. Ten warunek jest nazywany zachowaniem konfliktu. Zauważmy, że relacja konfliktu # jest generowana przez relację natychmiastowego konfliktu $\#_m$ i relację zależności $\leq$. Dla struktury zdarzeń dającej semantykę dla programów współbieżnych, relacja $\#_m$ będzie modelowana jako minimalna relacja generująca relację konfliktu $\#_m$

3 System śladów

Przez alfabet współbieżny rozumiemy uporządkowaną parę (Σ, I), gdzie Σ jest skończonym zbiorem symboli (nazw operacji) i $I \subseteq \Sigma \times \Sigma$ jest symetryczną i niezwrotną relacją binarną w Σ (relacja niezależności). Niech (Σ, I) będzie niezależnym alfabetem. Definiujemy $\equiv$ jako najmniejszą zgodność w (standardowym) łańcuchu $(\Sigma^*, \circ, e)$ taką, że $(a, b) \in I \Rightarrow ab \equiv ba$ dla wszystkich $a, b \in \Sigma$, np. $w \equiv w'$, jeśli jest skończona sekwencja łańcuchów $w_1, ..., w_n$ takich, że $w_1 = w, w_n = w'$ i dla każdego $i < n$, $w_i = uabv$, $w_{i+1} = uabv$, dla pewnych $(a, b) \in I$ i $u, v \in \Sigma^*$.

Klasy równoważności ≡ są nazywane śladami nad (Σ, I). Ślad generowany przez łańcuch w jest oznaczany przez $[w]$. Używamy następującej notacji: $[\Sigma^*] = \{[w] | w \in \Sigma^*\}$. Konkatenacja śladów $[w], [v]$, oznaczona $[w][v]$ jest zdefiniowane jako $[wv]$.

Niech T będzie zbiorem śladów nad (Σ, I). Relacja następstwa $\rightarrow$ w T jest zdefiniowana następująco: $[w_1] \rightarrow [w_2]$, jeśli jest $a \in \Sigma$ takie, że $[w_1][a] = [w_2]$. Relacja prefiksu $\leq$ w T jest zdefiniowana jako zwrotne i przechodnie domknięcie relacji następstwa, np. $\leq = (\rightarrow)^*$. Przez $<$ rozumiemy $\leq - id_T$. Niech $\tau \in T$ i $Q \subseteq T$. Używamy następującej notacji: $\downarrow_{\leq} Q = U_{\tau \in Q} \downarrow_{\leq} \tau$. Mówimy, że podzbiór Q od T dominuje nad innym podzbiorem R od T., jeśli $R \subseteq \downarrow_{\leq} Q$. Dwa ślady są zgodne, jeśli jest ślad T dominujący obydwa lub niezgodne w przeciwnym wypadku. Mówimy, że niezgodne ślady są w konflikcie.

Zbiór R śladów nazywa się właściwym, jeśli jego dwa zgodne ślady są zdominowane przez ślad w R, i kierowanym, gdy dowolne dwa ślady są zdominowane przez ślad w R. Zbiór śladów Q nazywany jest zamknięciem prefiksu, jeśli $Q = \downarrow_{\leq} Q$.

Uporządkowana para $(T, \rightarrow)$ jest systemem śladów nad (Σ, I), jeżeli T jest zamknięciem prefiksu i właściwym śladem języka nad (Σ, I) oraz $\rightarrow$ jest relacją prefiksu w T.

Dla każdego $w \in \Sigma^*$ niech $last(w) = a$, jeśli $w = w'a$. Dla każdego śladu $\tau \in [\Sigma^*]$ $Max(\tau) = \{last(w) | [w] = \tau\}$. Ślad τ jest nazywa się pierwotnym, gdy $|Max(\tau)| = 1$, np. gdy jest dokładnie jedna operacja wykonana ostatnio w τ.

4 Współbieżne programy o skończonej ilości stanów i ich semantyka

Programy są reprezentowane przez K-sekwencyjnych agentów komunikujących się przez wykonywanie wspólnych operacji.

Program jest strukturą P z następującymi elementami:

1. K jest skończonym zbiorem, nazywanym zbiorem procesów,
2. dla każdego procesu $i \in K$, skończony niepusty zbiór S_i, nazywany zbiorem stanów lokalnych procesu i,

3. dla każdego procesu $i \in K, s_i^0 \in S_i$ jest wyróżnionym stanem, nazywanym stanem początkowym procesu i,
4. dla każdego niepustego zbioru procesów $X \subseteq K, \Gamma_X \subseteq (\Pi_{i \in X} S_i)^2$ jest relacją tranzycji

Dla $s \in \Pi_{i \in X} S_i$ i $Y \subseteq X$ niech $s|_Y$ oznacza rzut s na elementy Y. Jeżeli $Y = \{i\}$, to napiszemy $s|_i$ zamiast $s|_{\{i\}}$.

Niektóre z rezultatów model checkingu będą zachodziły tylko dla ograniczonych klas programów w których tranzycje w konflikcie mające co najmniej jeden wspólny stan początkowy należą do tych samych sekwencyjnych agentów. Ta klasa programów jest nazywana programami z wolnym wyborem (free choice) i formalnie jest zdefiniowana następująco:

Program P jest programem z wolnym wyborem, jeśli dla jego każdych dwóch tranzycji $t = (s, s_1) \in \Gamma_X, t' = (s', s_1') \in \Gamma_Y$, albo $X \cap Y = \emptyset$, albo jeśli $s|_i = s'|_i$ dla pewnego $i \in X \cap Y$, to $X = Y$ i $s|_i = s'|_i$, dla wszystkich $i \in X$.

Relacja tranzycji Γ_X modeluje zdarzenia w których wszystkie procesy w X biorą udział. Dla każdej tranzycji $t = (s, s_1)$, $proc(t)$ oznacza zbiór procesów wplątanych w wykonanie t, np. $proc(t) = X$, jeżeli $t \in \Gamma_X$; $in(t) = s$, oraz $out(t) = s_1$. Niech $\Sigma = \bigcup_{X \subseteq K} \Gamma_X$. Relacja niezależności $I \subseteq \Sigma \times \Sigma$ tranzycji jest zdefiniowana następująco: $(t, t') \in I$, jeśli $proc(t) \cap proc(t') = \emptyset$. Relacja przyczynowości $D = \Sigma \times \Sigma \setminus (I \cup id_\Sigma)$. Relacja natychmiastowej przyczynowości $D_m = \{(t, t') \in D | in(t)|_i = in(t')|_i$, dla pewnych $i \in proc(t) \cap proc(t')\}$. Programy z wolnym wyborem mają następującą własność: jeżeli dwie tranzycje $(t, t') \in D_m$, to $proc(t) = proc(t')$ i $in(t)|_i = in(t')|_i$, dla każdego $i \in proc(t)$.

Niech $GS = \Pi_{i \in K} S_i$ będzie zbiorem stanów globalnych z P. Tranzycja $t = (s, s') \in \Gamma_X$ jest nazywana umożliwioną z globalnego stanu $g \in GS$ (oznaczonego $t \in enabled(g)$), jeżeli $g|_X = s$. Dla programów z wolnym wyborem, dla każdych dwóch tranzycji $(t, t') \in D_m$, jeśli jeden z nich jest umożliwiony ze stanu globalnego, wtedy drugi też jest umożliwiony z tego stanu.

Dla dwóch stanów globalnych $g, g' \in GS, g \xrightarrow{t} g'$, jeżeli dla pewnego $Y \subseteq K, t = (g|_Y, g'|_Y) \in \Gamma_Y$ i $g|_{K \setminus Y} = g'|_{K \setminus Y}$. Sekwencja wykonania $w = t_0 ... t_n \in \Sigma^*$ z P jest skończoną sekwencją tranzycji takich, że istnieje

sekwencja stanów globalnych $\xi = g_0 g_1 g_2 ... g_n$ z P z $g_0 = (s_1^0, ..., s_K^0)$, i $g_i \xrightarrow{t_i} g_{i+1}$, dla każdego $i < n$. Niech $g_0[w > g_n$ oznacza ten stan g_n, który jest osiągnięty po wykonaniu sekwencji operacji w ze stanu g_0.

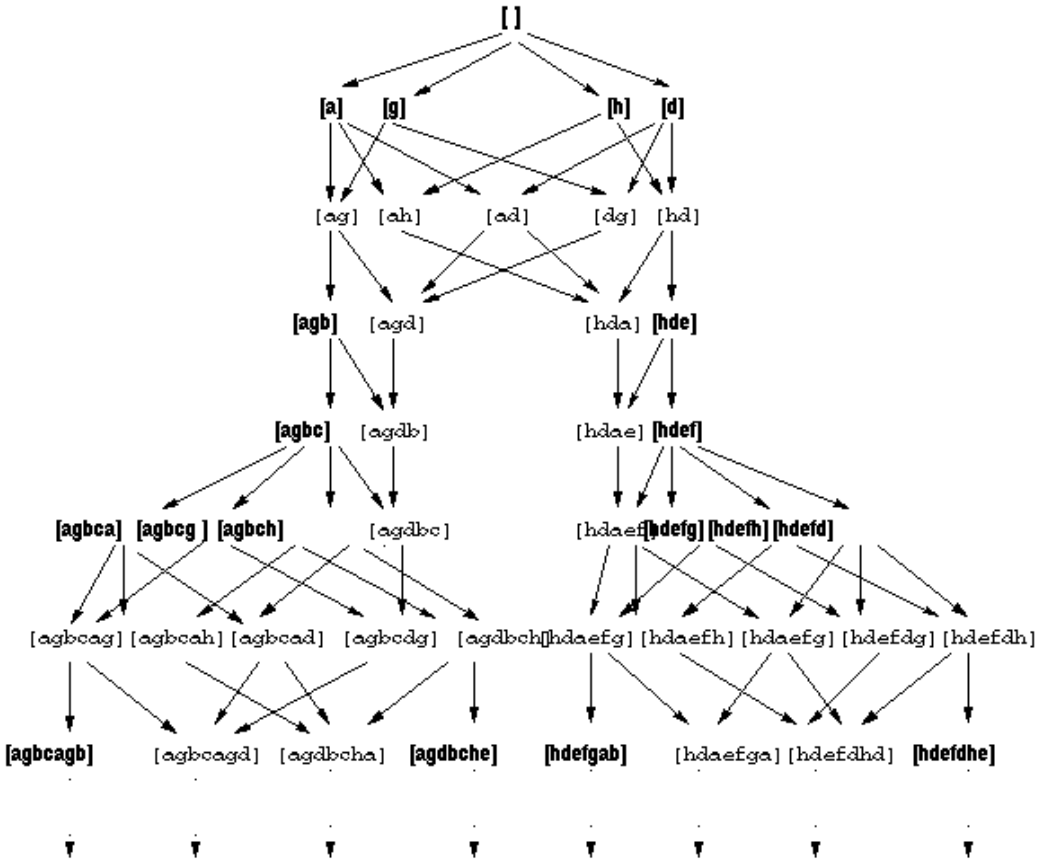
Przykład. Program MUTEX pokazany poprzednio składa się z trzech procesów, których stany lokalne są oznaczone okręgami, natomiast tranzycje poziomymi kreskami, np. $b = ((3,8),(5,10))$. Program zapewnia wzajemne wykluczanie w dostępie do stanów lokalnych 5 i 6 będących sekcjami krytycznymi. $S_1 = \{1,3,5\}$, $S_2 = \{2,4,6\}$, $S_3 = \{8,9,10,11\}$ oraz $s_1^0 = 1$, $s_2^0 = 2$, $s_3^0 = 7$.

5 Semantyka systemu śladów programu

Interpretowany system śladów $TS_P = (T, \rightarrow, I)$ nad (Σ, I) i zbiór formuł atomowych AP jest semantyką śladów programu P gdy są spełnione następujące warunki:

- $[w] \in T$, gdzie w jest sekwencją wykonania P,
- $\rightarrow$ jest relacją następstwa śladu w T,
- $I : T \rightarrow 2^{AP}$ jest funkcją interpretującą taką, że dla każdego $[w], [w'] \in T$ jeśli $g_0[w > g$ i $g_0[w' > g$, to $I([w]) = I([w'])$.

Funkcja interpretująca nie rozróżnia śladów wiodących do tego samego śladu globalnego.



System śladów semantyki programu MUTEX

6 Struktura zdarzeń programu

Każdy system śladów definiuje odpowiadającą mu strukturę zdarzeń, gdzie zdarzenia są zdefiniowane jako klasy równoważności łańcuchów tranzycji. Dla systemów śladów, które są semantykami programów zdefiniowanych powyżej, można identyfikować zdarzenia ze śladami pierwotnymi i oglądać odpowiadającą mu strukturę zdarzeń jako część struktury poprzedniego systemu śladów.

Niech $TS_P = (T, \rightarrow, I)$ będzie semantyką śladów programu P. Interpretowana struktura zdarzeń $ES_P = (E, E_0, \angle^E, \#_m^E, V^E)$ jest strukturą zdarzeń semantyki programu P, gdzie

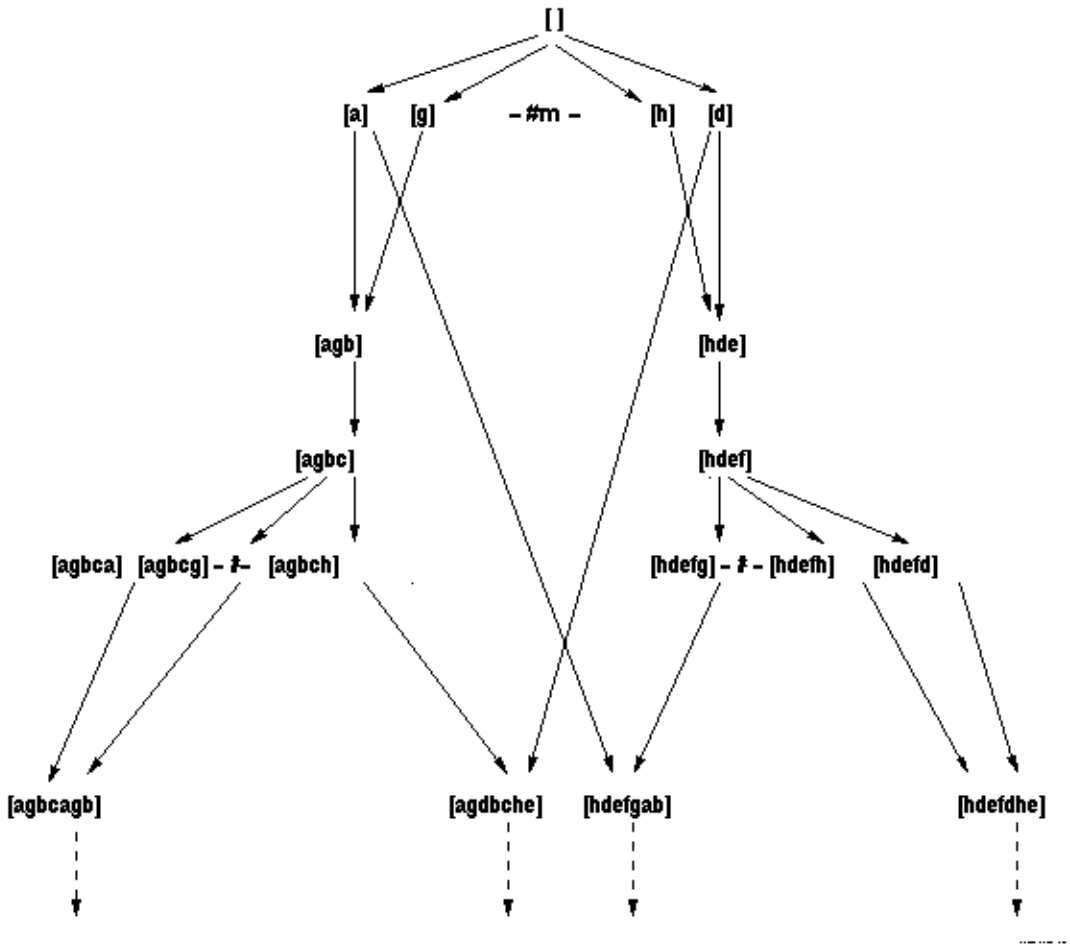
1. $E = \{e \in T \mid |Max(e)| = 1\}$,
2. $E_0 = \{[e]\}$,

3. $e \angle^E e'$ jeśli $e \to^* e'$ i $e \to^* e'' \to^* e'$ implikuje $e = e''$ lub $e' = e''$, dla każdego $e'' \in E$,
4. $e \#_m^E e'$, jeżeli e, e' są niezgodne i dla każdego $\tau \in T$ takiego, że $\tau \to e$ ślady τ, e' są zgodne w T i dla każdego $\tau' \in T$ takiego, że $\tau' \to e'$ ślady e, τ' są zgodne w T.
5. $V^E(e) = I(e)$, dla $e \in E$.

Warunek 3. odnosi się do faktu, że relacja $\angle^E$ jest zredukowana. Warunek 4. określa, że $\#_m^E$ jest minimalną relacją generującą relację konfliktu w E.

Lemat. Niech P będzie programem z wolnym wyborem i niech ES_P będzie strukturą zdarzeń semantyki P. Wtedy zachodzi następujący warunek:

- $e \#_m^E e'$ jeśli $(\exists \tau \in T)\tau \to e, \tau \to e'$ i $(Max(e), Max(e')) \in D_m$.



Struktura zdarzeń dla programu MUTEX

7 Skończona reprezentacja systemu śladów pierwotnych

System śladów jak również struktura zdarzeń pierwotnych są przeważnie obiektami nieskończonymi i jako takie nie są wygodne dla model checkingu. Dlatego stosuje się struktury TS_P, które identyfikują wszystkie ślady które nie mogą być rozróżnione przez formuły interesującej logiki.

Kiedy jest się zainteresowanym model checkingiem formuł CTL, wtedy skończona reprezentacja jest otrzymywana przez użycie relacji równoważności $\sim_{CTL}$ na śladach, która rozpoznaje ślady prowadzące do tych samych stanów globalnych, zdefiniowanej następująco: $(\forall [w], [w'] \in T)(\forall g \in GS)[w] \sim_{CTL} [w']$ jeżeli $(g_0[w' > g)$.

Gdy jest się zainteresowanym model checkingiem formuł CTL_P, wtedy skończona reprezentacja TS_P jest otrzymywana przez użycie bardziej ograniczającej relacji równoważności.

Dla logik struktur zdarzeń sytuacja jest trochę inna, gdyż skończona reprezentacja powinna wymagać tylko równoważności klas śladów pierwotnych. Najprostszym rozwiązaniem byłoby, gdyby mogła być użyta podstruktura TS_P z $\sim_{CTL}$. Nie jest to niestety możliwe, gdyż mogą być dwa ślady pierwotne prowadzące do tego samego stanu globalnego, które mają różne zależności w przyszłości i dlatego mogą być rozróżnione przez formuły DESL odnoszące się do relacji zależności. W śladzie semantyki MUTEXu (patrz rysunek) ślady: [], [agbc], [hdef] prowadzą do tego samego stanu globalnego (1, 2, 7), ale ich zależne następniki prowadzą do innych stanów globalnych (patrz rysunek wyżej).

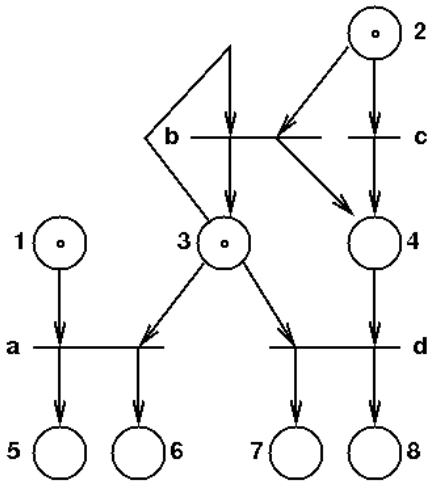
Rozwiązaniem jest odpowiednie zaostrzenie relacji równoważności $\sim_{CTL}$. Z przyczyn technicznych, nowa relacja równoważności jest zdefiniowana dla wszystkich śladów. Dwa ślady są ES-równoważne, jeśli prowadzą do tych samych śladów globalnych i ich zbiory maksymalnych

tranzycji są tożsame. Nowa relacja równoważności jest formalnie zdefiniowana następująco:

- $f : T \stackrel{def}{\rightarrow} \Pi_{i \in K} S_i \times (2^{\Sigma} \cup \{t_0\})$, gdzie $t_0 \notin \Sigma$ jest sztuczną tranzycją z $proc(t_0) = K$, $f([e]) = (g_0, t_0)$, $f([w]) = (g, X)$ jeśli $g_0[w > g$ oraz $X = Max([w])$ dla $[w] \neq [e]$ i
- $(\forall [w], [w'] \in T)[w] \sim_{ES} [w']$ jeżeli $f([w]) = f([w'])$.

Napiszemy $(g, t_1 ... t_n)$ dla $(g, \{t_1, ..., t_n\})$ i podstawimy $State(\tau)$ za g, gdzie $f(\tau) = (g, X)$.

Jak zostanie pokazane niżej, $\sim_{ES}$ identyfikuje wszystkie ślady pierwotne, które nie mogą być rozróżnione przez formuły DESL bez operatorów konfliktu. Niestety, dla nieograniczonych programów $\sim_{ES}$ -równoważne ślady pierwotne mogą być rozróżnione przez modalności konfliktu. Aby to zobaczyć, spójrzmy na poniższy rysunek.



Program Pr

Stany początkowe trzech procesów z Pr są zaznaczone kropkami. Ślady [a], [b], [c], [ac], [ba], [bd], [cd] należą do śladu semantyki Pr. Wszystkie poza [ac] są pierwotne. Można zauważyć, że $f([bd]) = f([cd]) = ((1, 7, 8), d)$, ale $[cd] \#_m^E [a]$, natomiast $[bd] \#_m^E [ba]$. Ponieważ $f([a]) \neq f([ba])$, ślady pierwotne [bd] i [cd] mogą być rozróżnione przez formułę konfliktu, odnosząc się do $f([a])$ lub $f([ba])$.

Są dwie możliwe drogi obejścia tego problemu:

1. Zdefiniowanie bardziej restrykcyjnej relacji równoważności na T, lub
2. Ograniczenie klasy programów tak, aby $\sim_{ES}$ zachowywała wszystkie formuły.

Pierwsze rozwiązanie prowadzi niechybnie do relacji równoważności, której koszt rośnie wykładniczo ze wzrostem liczby stanów globalnych, więc nie zajmujemy się dalej tym rozwiązaniem.

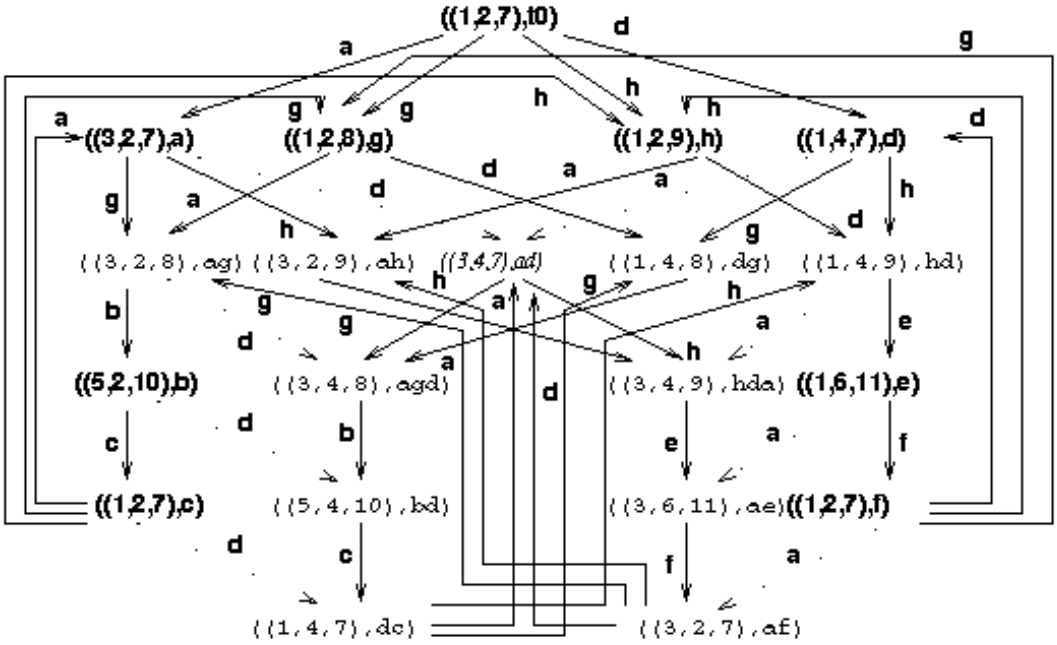
Drugie rozwiązanie wymaga takiego ograniczenia programów, które daje możliwość zdefiniowania natychmiastowej relacji konfliktu w kategoriach zachowanych przez relację zależności $\sim_{ES}$. Jest to sytuacja programów z wolnym wyborem.

Definiujemy strukturę TS_P przez $\sim_{ES}$ jako trójkę $F_{TS} = (W, \rightarrow, w_0)$, gdzie:

- $W = \{[\tau]_{\sim_{ES}}, \tau_1' \in T\}$ jest zbiorem stanów,
- $\rightarrow \subseteq W \times \Sigma \times W$ jest relacją tranzycji taką, że $[\tau]_{\sim_{ES}} \xrightarrow{t} [\tau']_{\sim_{ES}}$, jeśli są ślady $\tau_1 \in [\tau]_{\sim_{ES}}, \tau_1' \in [\tau']_{\sim_{ES}}$ i $t \in \Sigma$ takie, że $\tau_1[t] = \tau_1'$,
- $w_0 = [e]_{\sim_{ES}}$.

Definiujemy strukturę ES_P przez $\sim_{ES}$ jako czwórkę $F_{ES} = (E_F, \angle, \#_m, e_0)$ gdzie:

- $E_F = \{[e]_{\sim_{ES}} | e \in E\}$ jest zbiorem stanów,
- $\angle \subseteq E_F \times E_F$ jest relacją taką, że $[e]_{\sim_{ES}} \angle [e']_{\sim_{ES}}$, jeśli są ślady pierwotne $e_1 \in [e]_{\sim_{ES}}, e_1' \in [e']_{\sim_{ES}}$ takie, że $e \angle^E e'$,
- $\#_m \subseteq E_F \times E_F$ jest relacją taką, że $[e]_{\sim_{ES}} \#_m [e']_{\sim_{ES}}$, jeśli są ślady pierwotne $e_1 \in [e]_{\sim_{ES}}, e_1' \in [e']_{\sim_{ES}}$ takie, że $e_1 \#_m^E e_1'$,
- $e_0 = [e]_{\sim_{ES}}$.



Skończona reprezentacja programu MUTEX

8 Język DESL

Tematem pracy jest generowanie przestrzeni stanów. Ma ona jednak służyć sprawdzaniu na niej formuł języka DESL (Discrete Event Structure Logic).

1. Składnia i semantyka DESL

Niech $AP = \{p_1, p_2, ...\}$ będzie przeliczalnym zbiorem formuł atomowych. Użyte zostaną łączniki logiczne $\neg$ i $\wedge$, jak również modalności $\Box$ (zawsze w przyczynowej przyszłości), $\otimes$ (wszystkie przyczynowe następniki), $\Box_{\#}$ (wszystkie w konflikcie) i $\otimes_{\#}$ (wszystkie w natychmiastowym konflikcie). Zbiór formuł jest budowany indukcyjnie:

- każdy element AP jest formułą
- jeśli α i β są formułami, to są nimi również $\neg\alpha$ i $\alpha \wedge \beta$
- jeśli α jest formułą, to są nimi również $\Box\alpha$ i $\Box_{\#}\alpha$
- jeśli α jest formułą, to są nimi również $\otimes\alpha$ i $\otimes_{\#}\alpha$.

Zdefiniowane są następujące łączniki logiczne i modalności:

- $\alpha \vee \beta \stackrel{def}{=} \neg(\neg\alpha \wedge \neg\beta)$ (standard)

- $\alpha \Rightarrow \beta \stackrel{def}{=} \neg\alpha \vee \beta$ (standard)
- $\Diamond\alpha \stackrel{def}{=} \neg\Box\neg\alpha$ ($\Diamond$ - istnieje w przyczynowej przyszłości)
- $\Diamond_{\#}\,\alpha \stackrel{def}{=} \neg\Box_{\#}\neg\alpha$ ($\Diamond_{\#}$ - istnieje w konflikcie)
- $O\alpha \stackrel{def}{=} \neg \otimes \neg\alpha$ (O - istnieje przyczynowy następnik)
- $O_{\#}\alpha \stackrel{def}{=} \neg\otimes_{\#} \neg\alpha$ ($O_{\#}$ - istnieje w bezpośrednim konflikcie)

Uporządkowana para $M = (F, V)$ jest modelem, gdzie $F = (E, E_0, \angle, \#_m)$ jest przestrzenią zdarzeń i $V : E \to 2^{AP}$ jest funkcją wartościującą.

Niech M będzie modelem, $e \in E$ będzie stanem a α będzie formułą. $M, e| = \alpha$ oznacza, że formuła α jest prawdziwa w stanie e modelu M (M jest opuszczone). Notacja jest zdefiniowana indukcyjnie:

- $e| = p$ jeśli $p \in V(e)$, dla $p \in AP$,
- $e| = \neg\alpha$ jeśli nie $e| = \alpha$,
- $e| = \alpha \wedge \beta$ jeśli $e| = \alpha$ i $e| = \beta$
- $e| = \Box\alpha$, jeśli $(\forall e' \in E)(e \angle^* e' \Rightarrow e'| = \alpha)$,
- $e| = \Box_{\#}\alpha$, jeśli $(\forall e' \in E)(e \#_m \circ \angle^* e' \Rightarrow e'| = \alpha)$,
- $e| = \otimes\alpha$ jeśli $(\forall e' \in E)(e \angle e' \Rightarrow e'| = \alpha)$,
- $e| = \otimes_{\#}\alpha$ jeśli $(\forall e' \in E)(e \#_m e' \Rightarrow e'| = \alpha)$.

9 Algorytm DFS (przeszukiwanie grafu w głąb)

```
procedure DFS; {za [7]}
begin
   for v:=1 to n do
   begin
      visited[v]:=false
      ustaw wskaźnik current[v] na pierwszy wierzchołek na liście L[v]
   end;
   visit(p); visited[p]:=true;
   if current[p]<>nil then
   begin
      S:=[]; push (S, p);
      while S<>∅ do
      {stos S zawiera wszystkie nieodwiedzone do tej pory wierzchołki,
      z których wychodzą nieodwiedzone jeszcze krawędzie}
      begin
         v:=front(S)
         niech w będzie wierzchołkiem wskazywanym przez current[v]
         przesuń wskaźnik current[v] do następnego wierzchołka
         na liście L[v]
         if current[v]=nil then pop(S);
         if not visited[w] then
```

```
            begin
                visit(w);
                visited[w]:=true;
                if current[w]<>nil then push(S, w)
            end
        end
    end
end;
```

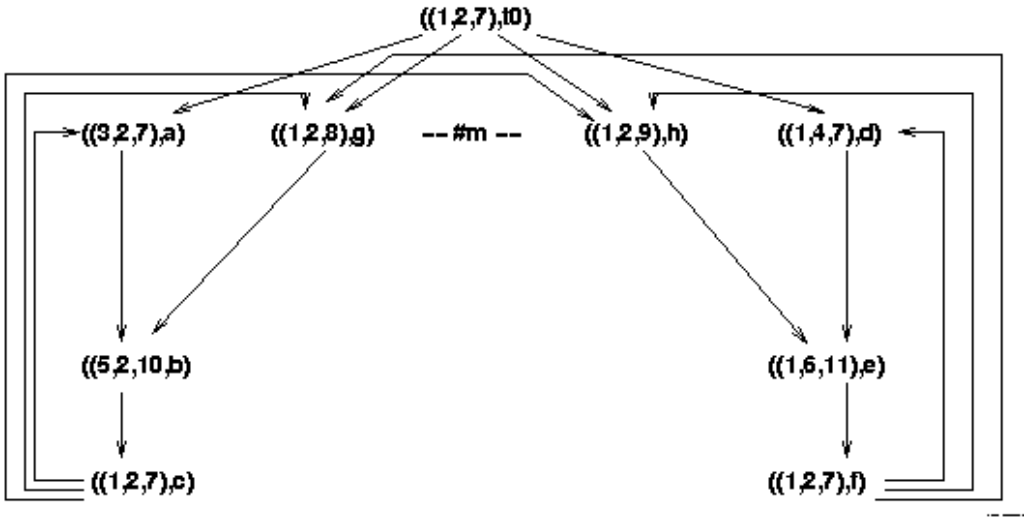
10 Generowanie przestrzeni stanów M

Podstawą jest algorytm DFS w wersji rekurencyjnej. k0 - konfiguracja początkowa, PS - przestrzeń stanów

```
procedure przeszukaj(k);
begin
    for each enabled in k transition t do
        begin
            l:=k+t;
            if l nie istnieje w PS then
                begin
                    PS:=PS+[l];
                    przeszukaj(l);
                end;
        end;
end;
PS:=[k0];
przeszukaj(k0);
```

11 Prosta redukcja przestrzeni stanów

Najprostszą (ale nieefektywną) metodą otrzymania zredukowanej przestrzeni stanów jest wygenerowanie całej przestrzeni wraz ze śladami, a następnie usunięcie z niej wszystkich markingów ze śladami globalnymi (tzn. z ilością elementów w śladzie>1). Jest to metoda bardzo kosztowna (zarówno pod względem pamięci, jak i czasu wykonania), ale dająca najmniejszą możliwą przestrzeń. W wielu wypadkach jest to jednak zadanie niewykonalne.



Program MUTEX po redukcji idealnej

12 Efektywne generowanie przestrzeni stanów pierwotnych

Zwiększenie efektywności powyższego algorytmu należy przeprowadzić w tym kierunku, aby już w czasie generowania tej przestrzeni eliminować niektóre zdarzenia. Tak więc korzystając z heurystyki ograniczamy zbiór operacji umożliwionych, które analizujemy. Niestety, nie jest otrzymamy w ten sposób najmniejszej możliwej przestrzeni (gdyż generujemy po jednej akcji; jeśli między dwoma śladami pierwotnymi jest ślad niepierwotny, to też trzeba go wygenerować). Metoda polega na tym, że z danego stanu wybieramy podzbiór zdarzeń umożliwionych taki, aby omijać (w miarę możliwości) stany niepierwotne. Następny ślad pierwotny musi być z tego samego procesu co aktualny. Wybieramy wszystkie takie operacje, że na drodze startującej z danej operacji znajdzie się operacja z tego procesu. Nie ominie się żadnej drogi (równoważnej), którą można dojść do stanu, w którym można wykonać operację z tego procesu. Jeśli jest droga między miejscami danego procesu jest poza tym procesem, to musi być też równoważna droga w tym procesie. Heurystyka: jeżeli mamy

akcję umożliwioną z tego samego procesu J, co ostatni ślad pierwotny, to ją bierzemy; bierzemy operacje które są w konflikcie; lub bierzemy wszystkie (patrz dalej: procedura WyliczZdarzeniaUmożliwione).

5 Program

1 Wymagania sprzętowe

Komputer zgodny z IBM PC, procesor minimum 286, minimum 2 MB RAM (1 MB Extended) - im więcej tym lepiej (tryb Protected). Dla komputerów z 640 KB RAM jest program w trybie Real - można obrabiać wtedy małe sieci. Zalecana myszka i dysk twardy.

2 Instalacja

Aby zainstalować program należy skopiować z dyskietki na dysk twardy plik lic14ins.exe. Najlepiej umieścić go w osobnym katalogu. Po uruchomieniu lic14ins, w katalogu rozpakuje się wersja wykonywalna programu.

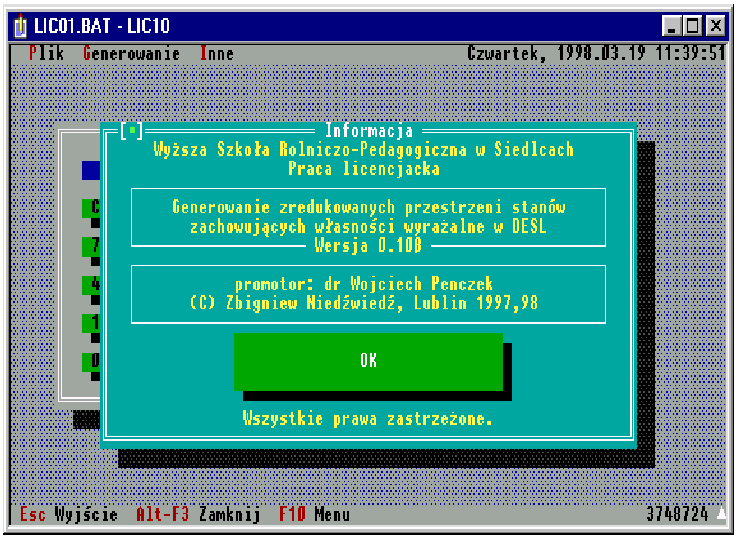
3 Polskie litery

Wewnętrznie (źródło, pliki danych i wydruków) - Mazovia; ekran - Latin2. Jeśli w systemie nie ma polskich liter w standardzie Latin2, należy przed uruchomieniem programu uruchomić vgapllat.com (dla karty graficznej VGA). Funkcja wydruku zamienia polskie litery na ich łacińskie odpowiedniki.

4 Obsługa programu

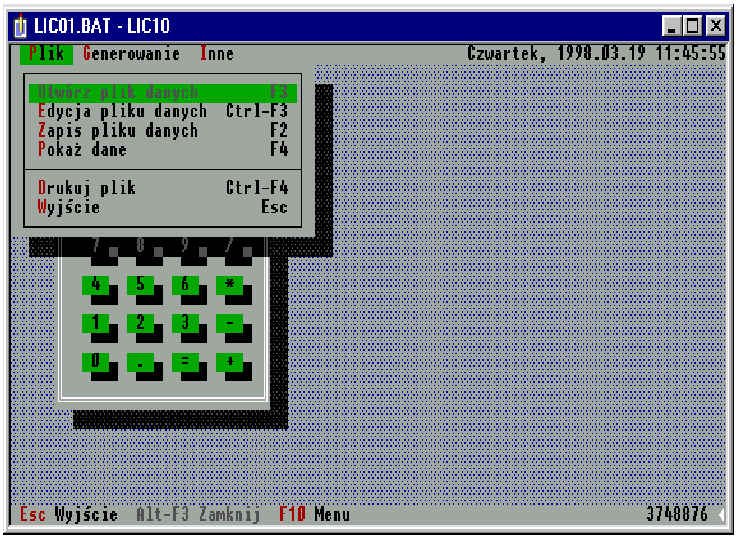
Wygląd programu jest efektem użycia pakietu Turbo Vision firmy Borland. Górna linia programu zawiera opuszczane menu (z lewej strony) oraz aktualną datę/czas (z prawej strony). Dolna linia zawiera opis najczęściej używanych klawiszy funkcyjnych oraz pole wskazujące wielkość dostępnej dla programu pamięci operacyjnej.

Po uruchomieniu programu ukazuje się okienko informacyjne. Aby je zamknąć, należy kliknąć myszką na przycisk OK (lub przycisk w lewym górnym rogu okienka) albo nacisnąć Enter, spację lub Escape. Okienko to można też wywołać ponownie (patrz menu *Inne*).

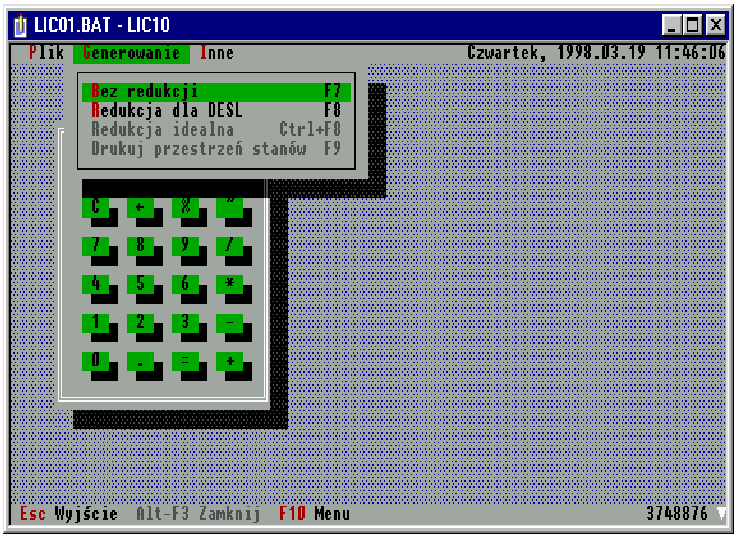


Okienko informacyjne (po uruchomieniu programu)

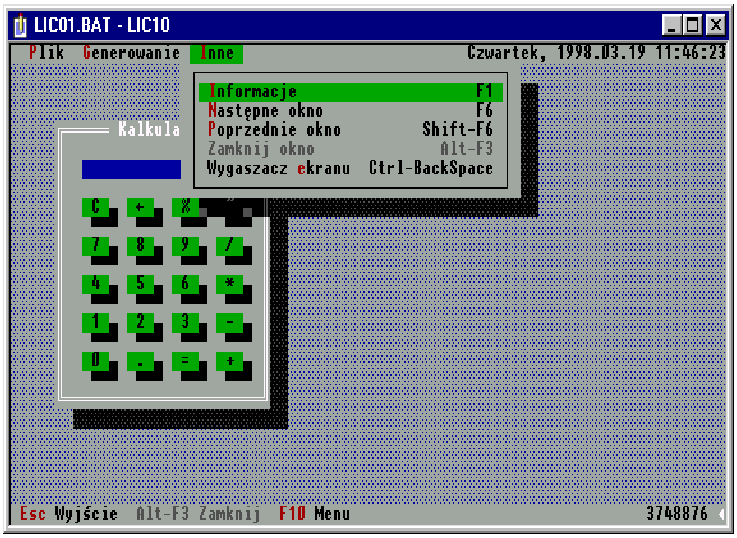
Obsługa menu może się odbywać zarówno przy pomocy klawiatury, jak i myszki. Operacje niedostępne w danym momencie są oznaczone na szaro. Oto trzy główne pod-menu:



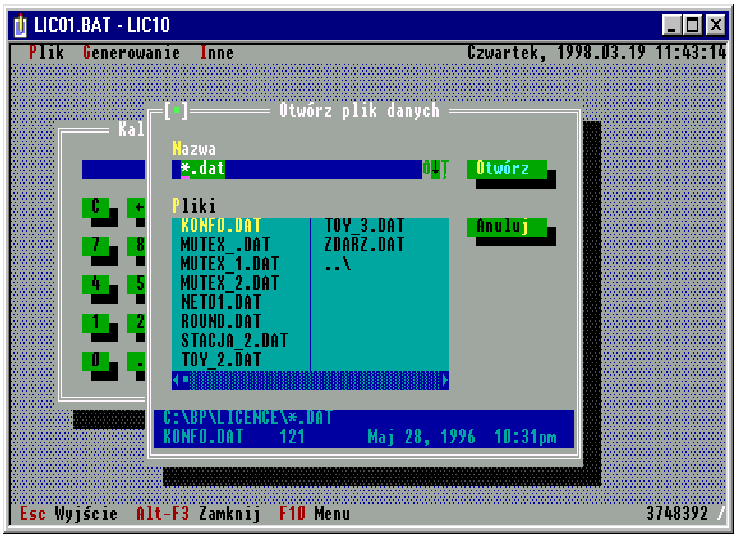
Menu Plik



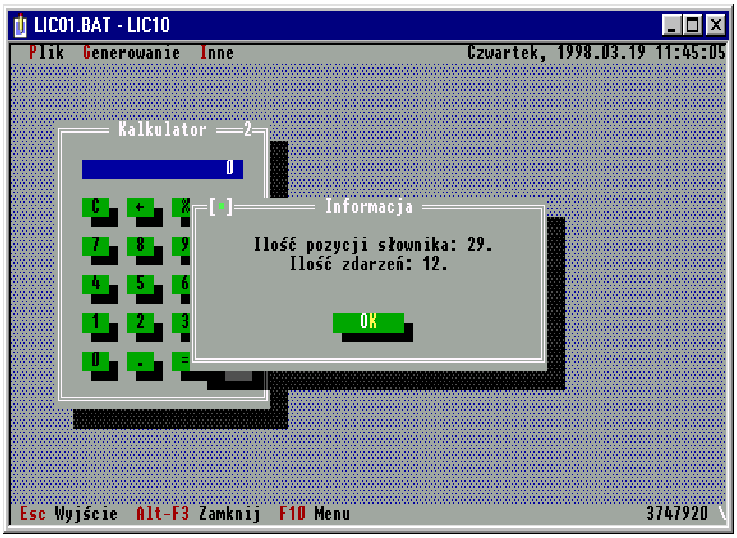
Menu Generowanie



Menu Inne

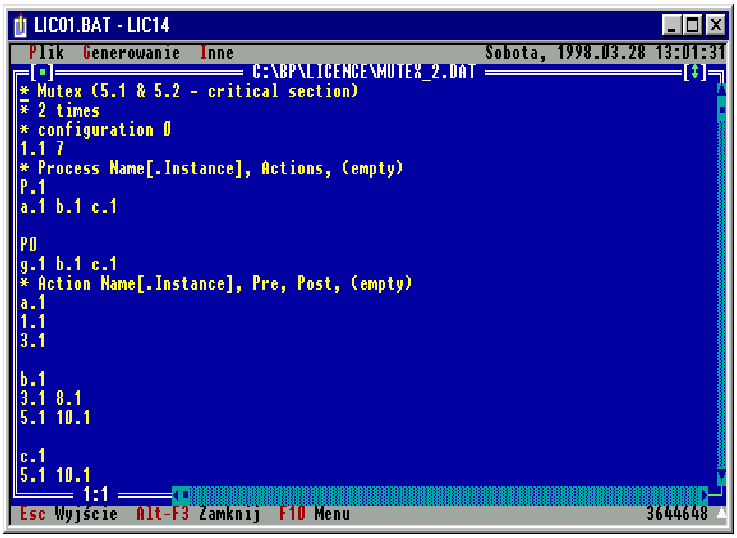


Otwieranie pliku danych



Komunikat o załadowaniu sieci

Po zamknięciu okienka informacyjnego program proponuje załadowanie pliku danych. Po wybraniu pliku i zaakceptowaniu go program wczytuje dane, ewentualnie skalując procesy (patrz podrozdział *Skalowanie*). Po wczytaniu wyświetla komunikat o ilości wykorzystanych pozycji słownika i ilości zdarzeń w systemie.

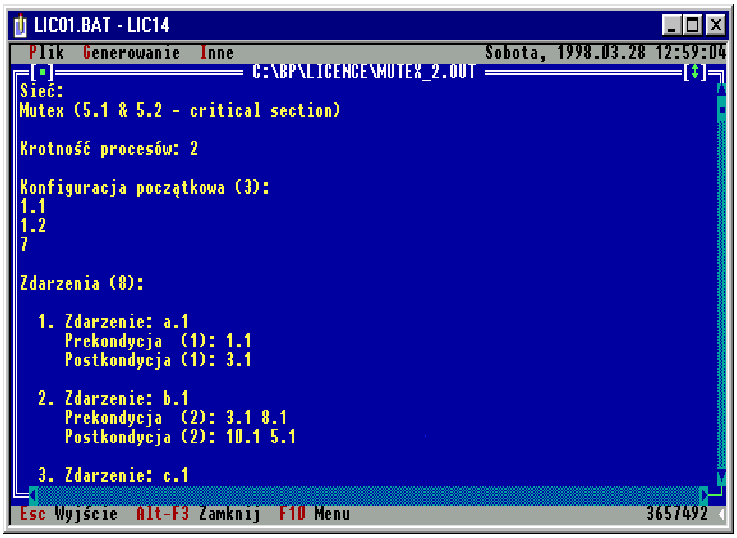


Edycja pliku danych

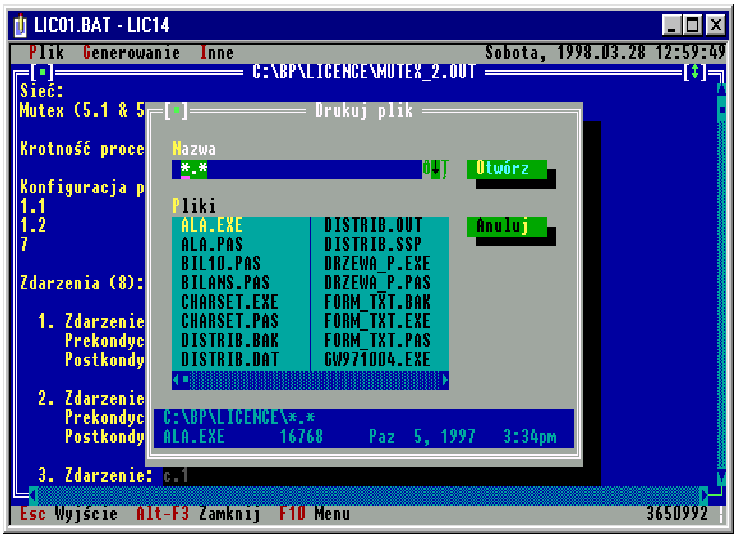
Plik danych można modyfikować dowolnym edytorem odczytującym i zapisującym kod ASCII, np. Edit z MS-DOS, notatnik z MS-Windows, edytor programu Norton Commander. W celu ograniczenia konieczności korzystania z innych programów, w program został wbudowany prosty edytor plików tekstowych oraz kalkulator.

Jeśli w lewym dolnym rogu okienka edycyjnego pojawi się gwiazdka - znaczy to, że dane uległy modyfikacji. Niezbędne jest wtedy użycie funkcji Zapisz - program nie pozwoli wyjść bez zapisu.

Po załadowaniu pliku danych można obejrzeć wyniki tej czynności. Mogą się one różnić od dostarczonych informacji ze względu na m.in. skalowanie.

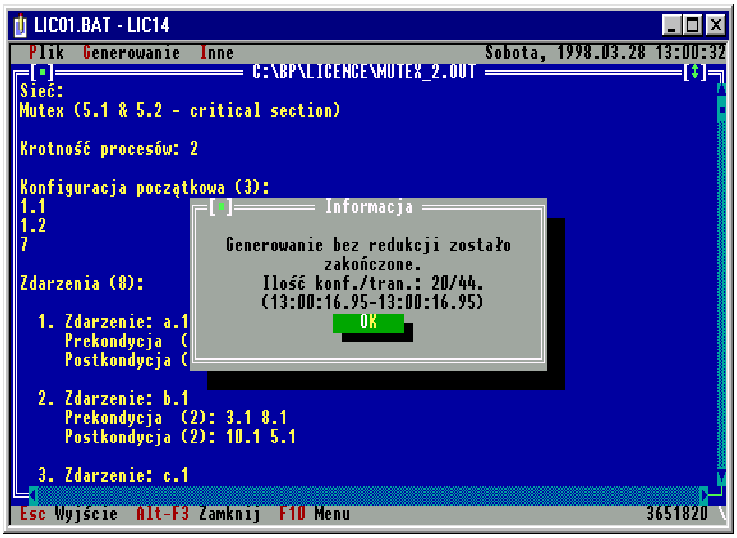


Wyświetlenie danych wejściowych



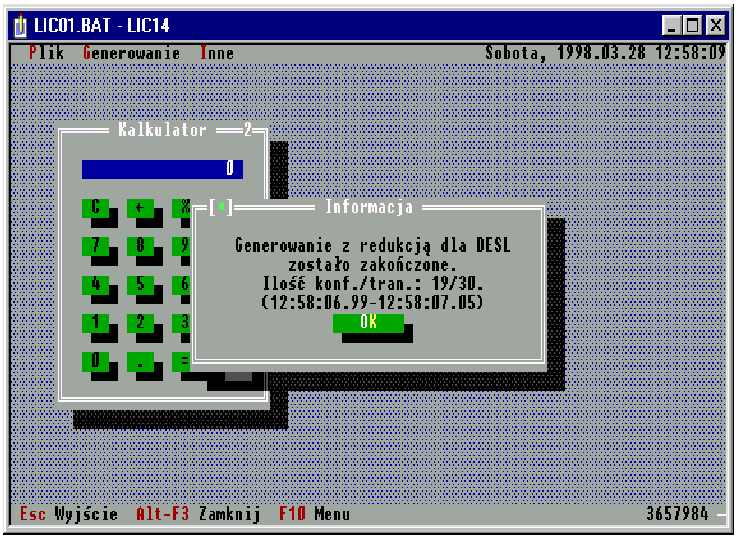
Wybór pliku do drukowania

Program umożliwia wydrukowanie dowolnego pliku tekstowego na drukarce podłączonej do portu LPT1. Aby uniknąć problemu polskich liter zawartość pliku jest konwertowana na alfabet łaciński.



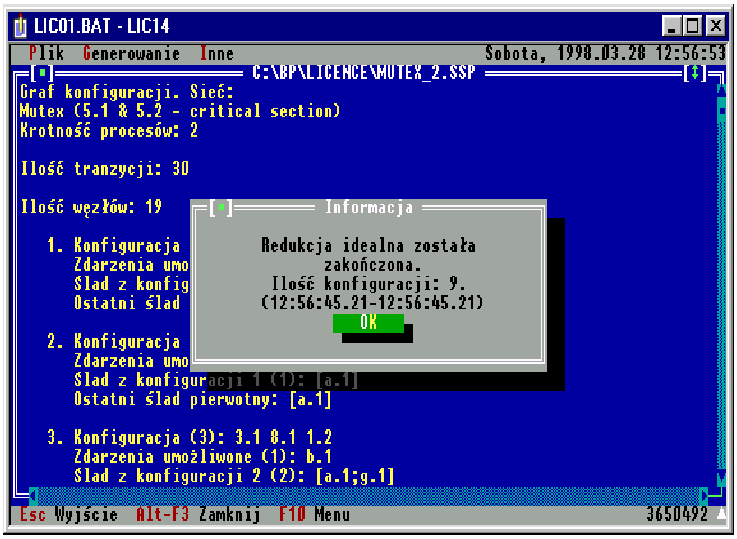
Komunikat o zakończeniu generowania przestrzeni stanów (bez redukcji)

Po załadowaniu sieci aktywne są pozycje menu dotyczące generowania przestrzeni stanów. Uwaga! Niektóre duże sieci mogą być generowane długo nawet na stosunkowo szybkim sprzęcie (patrz *Zestawienie wyników*). O trwającym procesie generowania świadczy zmniejszająca się ilość wolnej pamięci operacyjnej (prawy dolny róg ekranu).

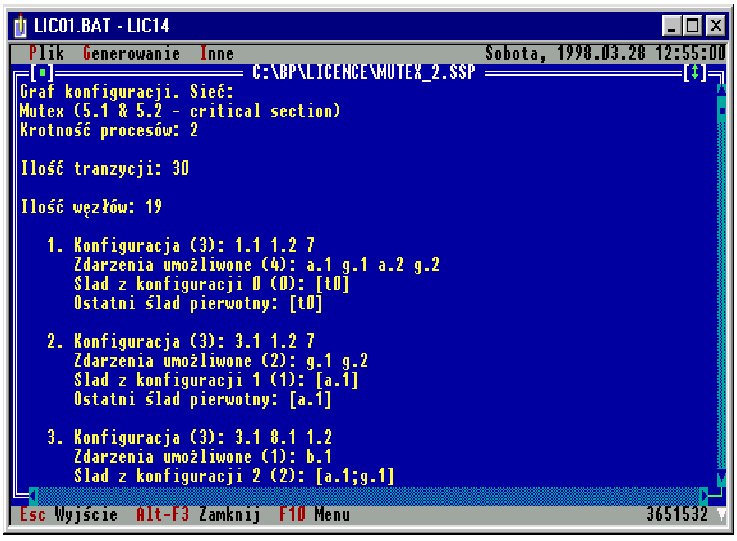


Komunikat o zakończeniu generowania przestrzeni stanów (z redukcją)

Po zakończeniu generowania można dokonać dodatkowej, idealnej redukcji. Komunikaty informujące o zakończeniu generowania/redukcji zawierają również czas rozpoczęcia/zakończenia tej czynności (w formacie godzina:minuta:sekunda.setna_część_sekundy).

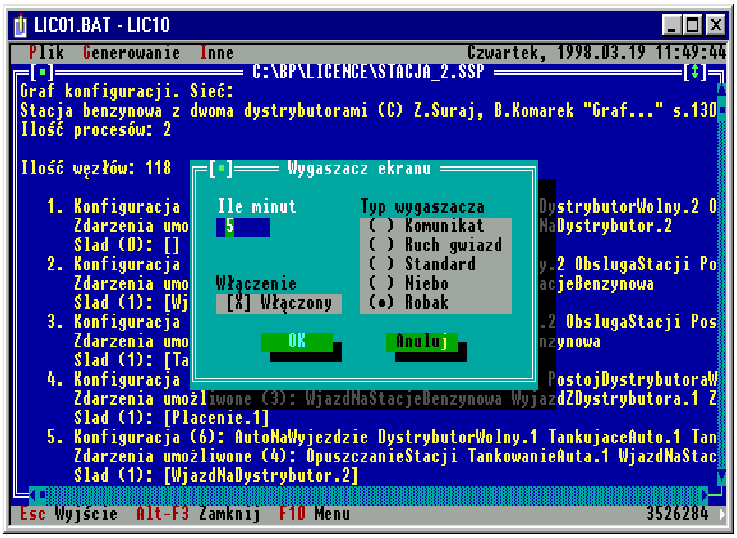


Komunikat o zakończeniu redukcji idealnej



Wyświetlenie przestrzeni stanów

Po wygenerowaniu przestrzeni stanów można ją obejrzeć (o ile jest na to wystarczająca ilość pamięci operacyjnej). W najwyższej linii okna przeglądania widnieje pełna ścieżka dostępu do przeglądanego pliku.



Konfiguracja wygaszacza ekranu

Aby oszczędzić monitor, warto go wygasić (przynajmniej częściowo). Funkcję tę pełni wygaszacz ekranu, standardowo skonfigurowany na 5 minut nieaktywności użytkownika (tzn. klawiatury i myszki) i pokazywanie wijącego się po ekranie robaka. Ustawienia te podlegają modyfikacji (np. w celu ograniczenia wpływu wygaszacza na czas generowania przestrzeni stanów bądź przy alergii na robaki).

5 Dane wejściowe - format

Wewnątrz pliku danych jest zawarty krótki opis jego zawartości. Do słownika pobierane są elementy z linii nie rozpoczynających się gwiazdką. Pierwsza linijka zawiera nazwę i krótki opis sieci. W drugiej linii zawarta jest informacja o krotności sieci (potrzebne do skalowania). Trzecia jest nagłówkiem konfiguracji początkowej, w czwartej zaś sama konfiguracja. Piąta jest nagłówkiem procesów. Format procesu (trzy linie): nazwa procesu, lista zdarzeń należących do procesu, pusta linia. Definiowanie procesów kończy nagłówek operacji. Format operacji (cztery linie) jest następujący: nazwa zdarzenia, prekondycja, postkondycja, pusta linia.

Definicję sieci kończy napis "* end". Przykładowe sieci znajdują się w załączniku.

6 Skalowalność

Jeżeli dany element sieci ma rozszerzenie (tzn. kropkę i numer, np. "a.1"), to przy generowaniu drugiego procesu przyjmie rozszerzenie zwiększone o numer_procesu-1 modulo liczba_procesów (tzn. np. "a.2"). Jeżeli element nie ma rozszerzenia, to albo jest to element wspólny dla wszystkich procesów, albo cały proces jest jedynym i wtedy nie ma to znaczenia. Długość rozszerzenia (np. "a.1" lub "a.01") jest dobierana automatycznie w zależności od ilości procesów. Dwa symbole specjalne "gen()" i "null" to odpowiednio źródło znaczników i "czarna dziura". Opracowanie formatu danych uwzględniającego skalowalność oraz samego skalowania procesów stanowi wkład własny autora.

7 Reprezentacja danych

Większość danych jest reprezentowana w programie za pomocą kolekcji (rozszerzenie języka Pascal z pakietu Turbo Vision) obiektów. Ich zaletą jest łatwość operowania na dużej (do 16380 elementów) ilości danych. W zależności od potrzeb, część kolekcji jest posortowana (np. słownik).

Kolekcja (jak każdy obiekt) posiada pola i metody. Oto najważniejsze:

1. Init - utworzenie kolekcji
2. Count - liczba elementów kolekcji
3. At(n) - wskaźnik n-tego elementu kolekcji
4. Compare - przy kolekcji posortowanej funkcja porównująca klucze
5. Insert - dołączenie nowego elementu

6. Delete - usunięcie elementu
7. Search - wyszukanie elementu

8 Najważniejsze obiekty

TIntCollection = object(TSortedCollection) - kolekcja danych typu Int

TSlownik = object(TStringCollection) - słownik nazw elementów sieci

TKonf1 = object(TIntCollection) - pojedyncza konfiguracja

TKonfig = object(TCollection) - kolekcja konfiguracji

TZdarz1 = object(TObject) - pojedyncze zdarzenie

TZdarzenia = object(TSortedCollection) - zbiór zdarzeń

TActions = object(TIntCollection) - elementy procesu

TProc1 = object(TObject) - pojedynczy proces

TProcesy = object(TSortedCollection) - zbiór procesów

TWezly = object(TIntCollection) - lista węzłów

TWezel = object(TObject) - element kolekcji konfiguracji

TGrafKonf = object(TCollection) - kolekcja konfiguracji i śladów

TMyApp = object(TAppPl) - aplikacja sterowana zdarzeniami

9 Algorytmy

Wczytywanie danych (procedura OpenData) w przypadku skalowania procesów polega na utworzeniu słownika a następnie kilkukrotnym przejrzeniu pliku danych.

Użyty w programie algorytm DFS różni się od przedstawionego w poprzednim rozdziale szczegółami implementacji. Procedura przeszukaj(k) nazywa się Buduj(n), przy czym n jest indeksem obiektu Wezel w kolekcji, w którym przechowywane są informacje m.in. o konfiguracji i śladzie. Jest w niej sprawdzany warunek ostatniego śladu pierwotnego w procesie. Jeśli tak jest, to dalsze konfiguracje nie są z niego wyliczane.

Redukcja przy generowaniu odbywa się w dwóch miejscach: procedura wyliczająca zdarzenia umożliwione przy danej konfiguracji usuwa niektóre z nich, jeśli parametr Redukcja jest prawdziwy.

```
procedure WyliczZdarzeniaUmozliwione;
begin
   Umożliwione:=[]
   for each Zdarzenie in Program.Zdarzenia do
   if Zdarzenie.Prekondycja<=Konfiguracja and
      Zdarzenie.Postkondycja*Konfiguracja=[]
   then Umożliwione:=Umożliwione+Zdarzenie;
   if Redukcja then {usunięcie niektórych zdarzeń z umożliwionych}
      begin
         Potrzebne:=[];
         for each Zdarzenie in Umożliwione do
            if Zdarzenie in
               ZdarzeniaProcesuOstatniegoŚladuPierwotnego
            then Potrzebne:=Potrzebne+Zdarzenie;
         if Potrzebne=[] then
            begin
               for each Zdarzenie1 in Program.Zdarzenia do
                  if Zdarzenie.Prekondycja in Konfiguracja then {częściowo umożl.}
                     for each Zdarzenie2 in Umożliwione do
                        if Zdarzenie2 in Zdarzenie1.Proces.Zdarzenia
                           then Potrzebne:=Potrzebne+Zdarzenie2
               for each Zdarzenie1 in Potrzebne do
                  for each Zdarzenie2 in Potrzebne do
                     if Zdarzenie1<>Zdarzenie2 and
                        Niezależne(Zdarzenie1, Zdarzenie2)
                     then Potrzebne:=Potrzebne-Zdarzenie2
            end;
         if Potrzebne<>[] then {usunięcie niepotrzebnych}
            for each Zdarzenie in Umożliwione do
               if not Zdarzenie in Potrzebne then
```

```
                    Umożliwione:=Umożliwione-Zdarzenie
        end;
end;
```

10 Ograniczenia

Ilość elementów kolekcji (słownika nazw, zdarzeń, konfiguracji przestrzeni stanów itd.) jest ograniczona do 16380. Dla niektórych sieci ograniczeniem jest też wielkość stosu, ograniczająca głębokość wywołań rekurencyjnych.

6 Wyniki eksperymentalne

1 Zestawienie wyników

sieć	Mutex_2					
ilość skalow. procesów	ilość zdarzeń	generowanie bez redukcji			redukcja idealna	
		generowanie - redukcja efektywna				
		il. węzłów	czas [s]	il. tranzycji	il. węzłów	
2	8	20	0,00	44	9	
		19	0,00	30		
3	12	56	0,06	168	13	
		42	0,00	72		
4	16	144	0,28	554	17	
		89	0,11	160		
5	20	352	2,20	1 600	21	
		184	0,33	342		
6	24	832	15,05	4 416	25	
		381	1,31	728		
7	28	0	0,00	0	29	
		796	5,32	1 560		
8	32	0	0,00	0	33	
		1 677	24,27	3 366		

sieć	Toy_2					
ilość skalow. procesów	ilość zdarzeń	generowanie bez redukcji			redukcja idealna	
		generowanie - redukcja efektywna				
		il. węzłów	czas [s]	il. tranzycji	il. węzłów	
2	2	4	0,00	4	3	
		3	0,00	2		
3	3	8	0,00	12	4	
		4	0,00	3		
4	4	16	0,00	32	5	
		5	0,00	4		
5	5	32	0,00	80	6	
		6	0,00	5		

sieć	Toy_2				
ilość skalow. procesów	ilość zdarzeń	generowanie bez redukcji			redukcja idealna
		generowanie - redukcja efektywna			
		il. węzłów	czas [s]	il. tranzycji	il. węzłów
6	6	64	0,06	192	7
		7	0,00	6	
7	7	128	0,22	448	8
		8	0,00	7	
8	8	256	0,90	1 024	9
		9	0,00	8	
9	9	512	4,84	2 304	10
		10	0,00	9	
10	10	1 024	21,97	5 120	11
		11	0,00	10	
11	11	2 048	104,19	11 264	12
		12	0,00	11	
12	12	4 096	487,74	24 576	13
		13	0,00	12	
13	13	8 192	2 238,93	53 248	14
		14	0,00	13	
25	25	0	0,00	0	26
		26	0,00	25	
50	50	0	0,00	0	51
		51	0,05	50	
100	100	0	0,00	0	101
		101	0,44	100	
200	200	0	0,00	0	201
		201	3,24	200	
400	400	0	0,00	0	401
		401	30,81	400	
1 000	1 000	0	0,00	0	1 001
		1 001	453,85	1 000	

sieć	Toy_3					
ilość skal. procesów	ilość zdarzeń	generowanie bez redukcji			redukcja idealna	
		generowanie - redukcja efektywna				
		il. węzłów	czas [s]	il. tranzycji	il. węzłów	
2	5	10	0,00	15	6	
		9	0,00	11		
3	7	28	0,00	58	8	
		16	0,00	20		
4	9	82	0,11	221	10	
		25	0,00	31		
5	11	244	0,61	816	12	
		36	0,00	44		
6	13	730	5,88	2 923	14	
		49	0,05	59		
7	15	2 188	65,20	10 214	16	
		64	0,05	76		
8	17	6 562	777,25	35 001	18	
		81	0,11	95		

sieć	Toy_4					
ilość skal. procesów	ilość zdarzeń	generowanie bez redukcji			redukcja idealna	
		generowanie - redukcja efektywna				
		il. węzłów	czas [s]	il. tranzycji	il. węzłów	
2	9	26	0,00	43	10	
		17	0,00	19		
3	13	126	0,11	304	14	
		32	0,05	36		
4	17	636	2,62	2 005	18	
		51	0,05	57		
5	21	3 126	85,30	12 506	22	
		74	0,06	82		

sieć	Round				
ilość skalow. procesów	ilość zdarzeń	generowanie bez redukcji			redukcja idealna
		generowanie - redukcja efektywna			
		il. węzłów	czas [s]	il. tranzycji	il. węzłów
2	14	72	0,00	148	25
		72	0,00	148	

2 Uwagi

1. Pomiary przeprowadzono na komputerze zgodnym z IBM PC wyposażonym w procesor Pentium MMX taktowany zegarem 188 MHz.
2. Dokładność zegara w IBM PC wynosi 1/18 sekundy, czyli niecałe 0,06 sekundy.
3. Dla 8192 węzłów niezbędne było udostępnienie dla programu pełnych 15 MB pamięci Extended.
4. Program dla trybu Real jest szybszy (ale obrabia mniejsze sieci).
5. Czas generowania 0 oznacza czas krótszy niż jedno "tyknięcie" zegara (1/18 sekundy).
6. Określenie "bez redukcji" nie oznacza, że generowane są wszystkie możliwe ślady, gdyż ślady zawierają tylko maksymalne niezależne zdarzenia a ponadto nie są generowane powtarzające się konfiguracje.
7. Czas redukcji idealnej był zawsze mniejszy niż 1/18 sekundy.
8. Na dokładność pomiaru mają wpływ procedury wyświetlające aktualny czas, datę ilość wolnej pamięci operacyjnej i ew. wygaszacz ekranu - pracują one podczas generowania przestrzeni stanów.

9. Poprawność generowania zredukowanych przestrzeni stanów w powyższych przykładów łatwo sprawdzić, wykonując redukcję idealną po generowaniu bez redukcji oraz generowaniu z redukcją i porównując wielkość przestrzeni.
10. Ilość tranzycji jest sumą zdarzeń umożliwionych w każdym analizowanym węźle.

7 Wnioski

W sieci Toy_2 występuje spektakularna redukcja, bo program jest trywialny. Poszczególne procesy są od siebie całkowicie niezależne, występuje też łatwy do uchwycenia moment zakończenia danego procesu (co oznacza, że nie będzie już w nim dalej stanów pierwotnych, a więc można go dalej nie analizować). Warto zwrócić uwagę na stopień redukcji - z wykładniczej do liniowej (wielkość przestrzeni) lub wielomianowej (czas generowania). Wydłużenie procesów (Toy_5) daje dalej rewelacyjne wyniki. Zakończenie akcją synchronizującą (Toy_6) nie zmienia tego stanu, ale synchronizacja w środku procesu (Toy_7) nieco zmniejsza wielkość redukcji.

Redukcja występuje wtedy, gdy istnieją duże fragmenty poszczególnych procesów, które pracują od siebie całkowicie niezależnie. Jeśli procesy się często synchronizują, to redukcja jest niewielka lub żadna (sieć Round). Wynika to z samej metody generowania, ponieważ między dwoma stanami lokalnymi mogą występować stany globalne. Ponieważ generujemy po jednym kroku, to musimy (przynajmniej jedną drogą) przejść po tych stanach globalnych.

Warto używać redukcji podczas generowania, ponieważ nie prowadzi ona do zwiększenia czasu generowania przestrzeni stanów. Poza tym warto pamiętać o tym, że użyte w programie określenie "bez redukcji" nie jest w pełni prawdą, ponieważ ślad jest zdefiniowany jako maksymalna operacja w każdym procesie oraz nie są generowane powtarzające się konfiguracje i ślady.

Wydaje się, że warto również po wygenerowaniu zredukowanej przestrzeni zredukować ją idealnie, ponieważ czas tej redukcji nie był

uchwytny na testowym komputerze w żadnym wypadku, nawet przy dużych przestrzeniach.

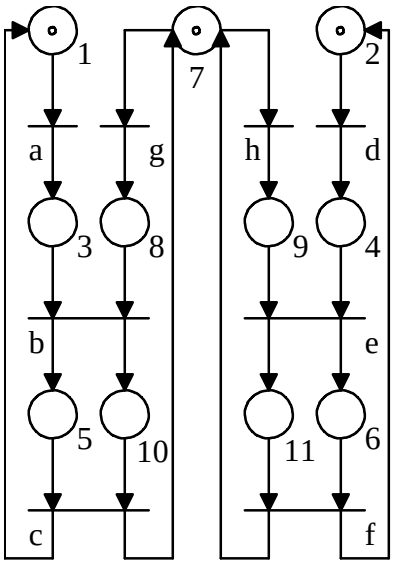
8 Bibliografia

1. Wojciech Penczek, Model-Checking for a Subclass of Event Structures, Proc. TACAS'97, LNCS 1217, 1997
2. Wojciech Penczek, Model-Checking for a Fragment of Event Structure Logic, 820, Wydawnictwo IPI PAN, Warszawa 1996
3. Doron Peled, Partial Order Reduction: Model-Checking using Representatives, Bell Laboratories, Proc. of MFCS'96, LNCS 1113 str. 93-112, 1996
4. Doron Peled, All from One, One for All: on Model-Checking using Representatives, AT&T Bell Laboratories, 5th Conference on Computer Aided Verification, Grecja, LNCS 697 Springer 1993, str. 409-423.
5. Zbigniew Suraj, Bogumił Komarek, GRAF - System graficznej konstrukcji i analizy sieci Petriego, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1994
6. Bronisław Jankowski, Grafy, algorytmy w Pascalu, ZNI "Mikom", Warszawa 1998
7. L. Banachowski, K. Diks, W. Rytter, Algorytmy i struktury danych, WNT Warszawa 1996
8. Jean-Michel Couvreur, Denis Poitrenaud, Model Checking based on Occurence Net Graph, R058b FORTE/PSTV '96
9. Verification Based on Local States, Michaela Huhn, Peter Niebert, Frank Wallner, Proc. of TACAS'98, 1998
10. Put your Model Checker on Diet: Verification on Local States, Michaela Huhn, Peter Niebert, Frank Wallner, Technical Report TUM-I9642, Technische Universität München, grudzień 1996

9 Załączniki

1 Przykładowe sieci Petriego (*.dat)

1. Mutex_ (wersja nieskalowalna)



Program Mutex

```
* Mutex (5 & 6 - critical section) - see Mutex_2
* 1 times
* configuration 0
1 2 7
* Process Name[.Instance], Actions, (empty)
P1
a b c

P2
d e f

P0
g h b e c f
* Action Name[.Instance], Pre, Post, (empty) a 1 3
b
3 8
5 10

c
5 10
1 7
```

```
d
2
4

e
9 4
11 6

f
11 6
2 7

g
7
8

h
7
9
* end
```

2. Mutex_2 (wersja skalowalna)

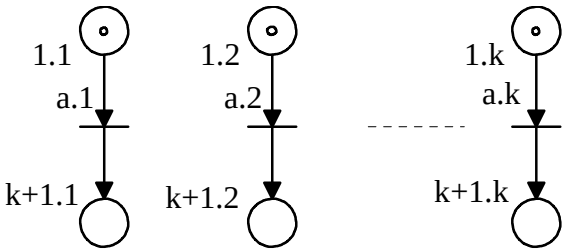
```
* Mutex (5.1 & 5.2 - critical section)
* 2 times
* configuration 0
1.1 7
* Process Name[.Instance], Actions, (empty)
P.1
a.1 b.1 c.1

P0
g.1 b.1 c.1
* Action Name[.Instance], Pre, Post, (empty)
a.1
1.1
3.1

b.1
3.1 8.1
5.1 10.1

c.1
5.1 10.1
1.1 7

g.1
7
```

```
8.1
* end
```

3. Mutex_1

Mutex_1 zawiera to samo, co Mutex_, ale liczby są uzupełnione z przodu zerami, co poprawia sortowanie słownika.

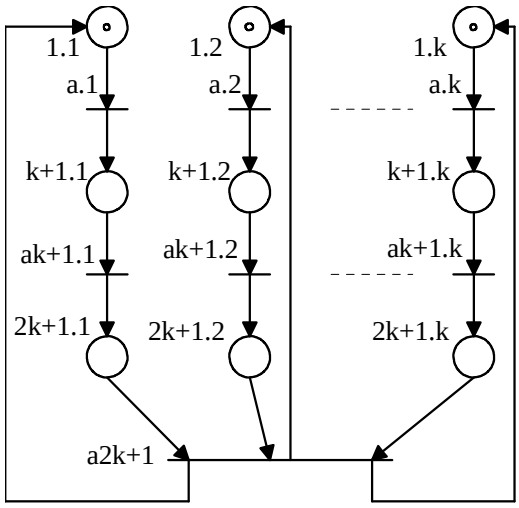
4. Toy_2



Program Toy_2

```
* Toy 2 - global state space = 2^k states, after reduction k+1 states
* 9 times
* configuration 0
1.1
* Process Name[.Instance], Actions, (empty)
P.1
a.1
* Action Name[.Instance], Pre, Post, (empty)
a.1
1.1
k+1.1
* end
```

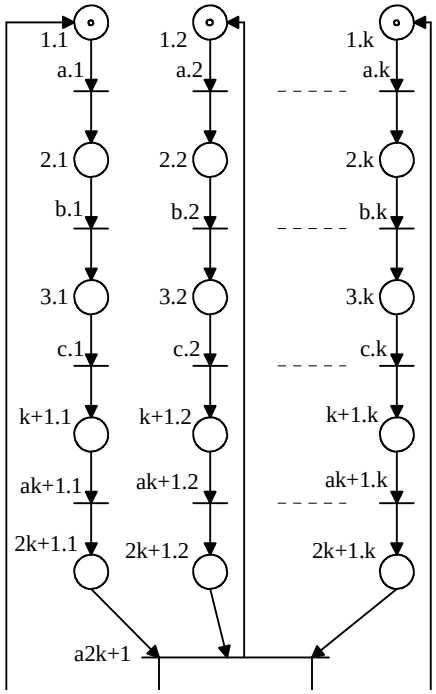
5. Toy_3



Program Toy_3

```
* Toy 3 - global state space = 3^k states, after reduction 2k^2 states
* 8 times
* configuration 0
1.1
* Process Name[.Instance], Actions, (empty)
P.1
a.1 ak+1.1 a2k+1
* Action Name[.Instance], Pre, Post, (empty)
a.1
1.1
k+1.1

ak+1.1
k+1.1
2k+1.1

a2k+1
2k+1.1
1.1
* end
```

6. Toy_4



Program Toy_4

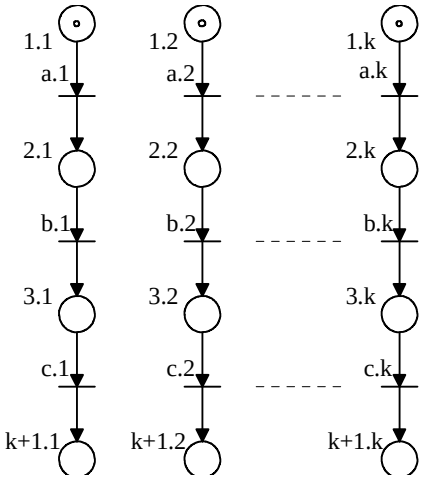
Podobnie jak Toy_3; w każdym procesie wstawione zostały zdarzenia b.1 i c.1 między a.1 i ak+1.1.

```
* Toy 4
* 3 times
* configuration 0
1.1
* Process Name[.Instance], Actions, (empty)
P.1
a.1 b.1 c.1 ak+1.1 a2k+1
* Action Name[.Instance], Pre, Post, (empty)
a.1
1.1
2.1

b.1
2.1
3.1

c.1
3.1
k+1.1

ak+1.1
```

k+1.1
2k+1.1

a2k+1
2k+1.1
1.1
* end

7. Toy_5



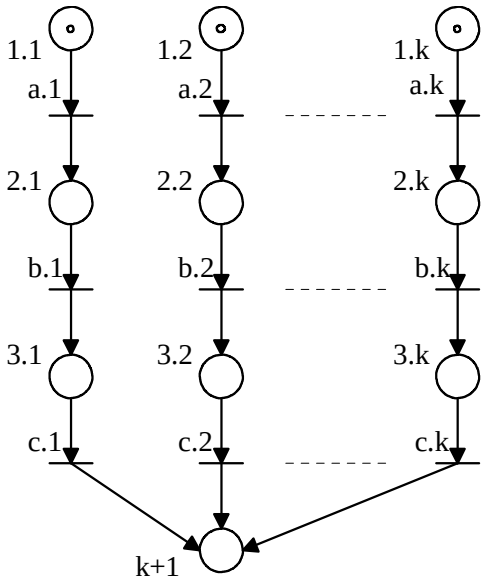
Program Toy_5

* Toy 5 - Toy 2 + 2 states
* 6 times
* configuration 0
1.1
* Process Name[.Instance], Actions, (empty)
P.1
a.1 b.1 c.1
* Action Name[.Instance], Pre, Post, (empty)
a.1
1.1
2.1

b.1
2.1
3.1

c.1
3.1
k+1.1
* end

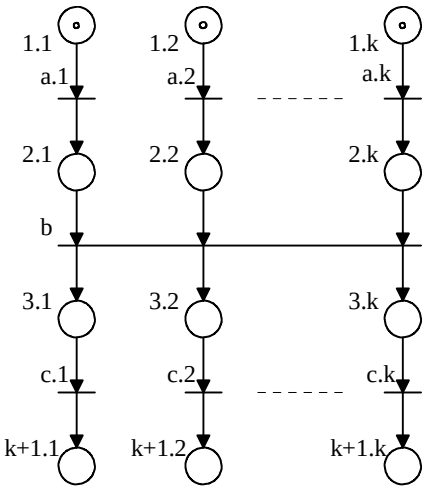
8. Toy_6



Program Toy_6

```
* Toy 6 - Toy 5 + sync. at end
* 8 times
* configuration 0
1.1
* Process Name[.Instance], Actions, (empty)
P.1
a.1 b.1 c.1
* Action Name[.Instance], Pre, Post, (empty)
a.1
1.1
2.1

b.1
2.1
3.1

c.1
3.1
k+1
* end
```

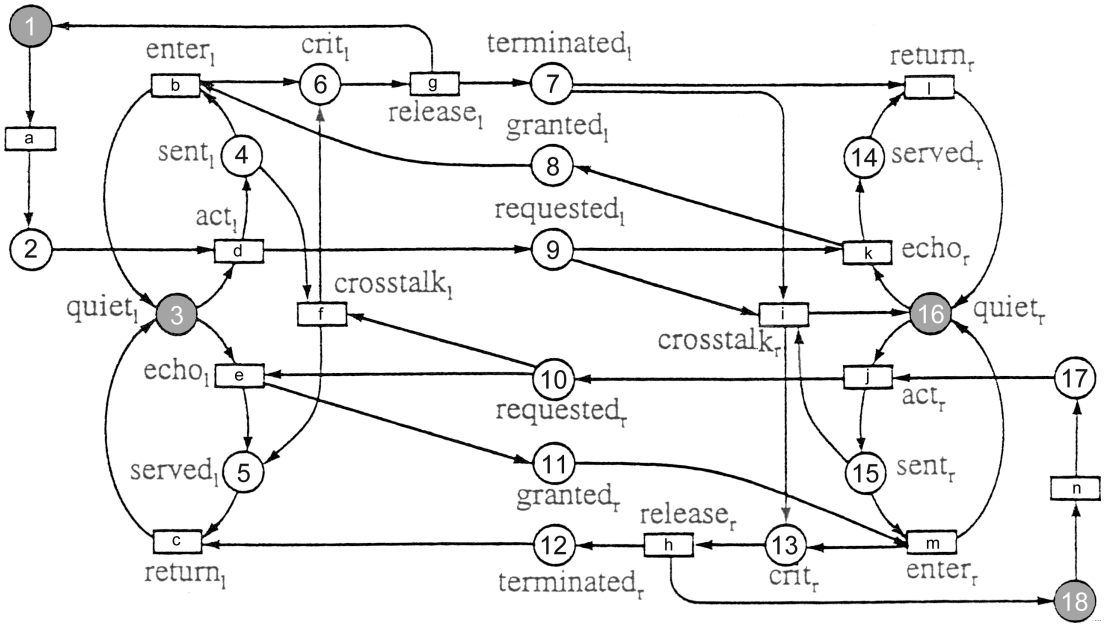
9. Toy_7



Program Toy_7

```
* Toy 7 - Toy 5 + sync at middle
* 3 times
* configuration 0
1.1
* Process Name[.Instance], Actions, (empty)
P.1
a.1 b c.1
* Action Name[.Instance], Pre, Post, (empty)
a.1
1.1
2.1

b
2.1
3.1

c.1
3.1
k+1.1
* end
```

10. Round



Program Round

* Round Based Mutual Exclusion (from Model... J.-M. Courvreur, D. Poitrenaud)
* 2 times
* configuration 0
local.1 quiet.1
* Process Name[.Instance], Actions, (empty)
P1
apply.1 act.1 enter.1 release.1 crosstalk.1 echo.1 return.1
* Action Name[.Instance], Pre, Post, (empty)
apply.1
local.1
pend.1

act.1
pend.1 quiet.1
sent.1 requested.1

enter.1
granted.1 sent.1
crit.1 quiet.1

release.1
crit.1
terminated.1 local.1

crosstalk.1
requested.2 sent.1
crit.1 served.1

echo.1
quiet.1 requested.2
granted.2 served.1

return.1
served.1 terminated.2
quiet.1
* end

2 Przykładowe przestrzenie stanów (*.ssp)

1. Mutex - bez redukcji

Graf konfiguracji. Sieć:
Mutex (5 & 6 - critical section) - see Mutex_2
Krotność procesów: 1

Ilość tranzycji: 44

Ilość węzłów: 20

1. Konfiguracja (3): 1 2 7
   Zdarzenia umożliwone (4): a d g h
   Ślad z konfiguracji 0 (0): [t0]
   Ostatni ślad pierwotny: [t0]

2. Konfiguracja (3): 2 3 7
   Zdarzenia umożliwone (3): d g h
   Ślad z konfiguracji 1 (1): [a]
   Ostatni ślad pierwotny: [a]

3. Konfiguracja (3): 3 4 7
   Zdarzenia umożliwone (2): g h
   Ślad z konfiguracji 2 (2): [a;d]
   Ostatni ślad pierwotny: [a]

4. Konfiguracja (3): 3 4 8
   Zdarzenia umożliwone (1): b
   Ślad z konfiguracji 3 (3): [a;d;g]
   Ostatni ślad pierwotny: [a]

5. Konfiguracja (3): 10 4 5
   Zdarzenia umożliwone (1): c
   Ślad z konfiguracji 4 (2): [b;d]
   Ostatni ślad pierwotny: [a]

6. Konfiguracja (3): 1 4 7

Zdarzenia umożliwone (3): a g h
Ślad z konfiguracji 5 (2): [c;d]
Ostatni ślad pierwotny: [a]

7. Konfiguracja (3): 1 4 8
Zdarzenia umożliwone (1): a
Ślad z konfiguracji 6 (2): [d;g]
Ostatni ślad pierwotny: [a]

8. Konfiguracja (3): 1 4 9
Zdarzenia umożliwone (2): a e
Ślad z konfiguracji 6 (2): [d;h]
Ostatni ślad pierwotny: [a]

9. Konfiguracja (3): 3 4 9
Zdarzenia umożliwone (1): e
Ślad z konfiguracji 8 (3): [a;d;h]
Ostatni ślad pierwotny: [a]

10. Konfiguracja (3): 11 3 6
Zdarzenia umożliwone (1): f
Ślad z konfiguracji 9 (2): [a;e]
Ostatni ślad pierwotny: [a]

11. Konfiguracja (3): 2 3 7
Zdarzenia umożliwone (3): d g h
Ślad z konfiguracji 10 (2): [a;f]
Ostatni ślad pierwotny: [a]

12. Konfiguracja (3): 2 3 8
Zdarzenia umożliwone (2): b d
Ślad z konfiguracji 11 (2): [a;g]
Ostatni ślad pierwotny: [a]

13. Konfiguracja (3): 10 2 5
Zdarzenia umożliwone (2): c d
Ślad z konfiguracji 12 (1): [b]
Ostatni ślad pierwotny: [b]

14. Konfiguracja (3): 1 2 7
Zdarzenia umożliwone (4): a d g h
Ślad z konfiguracji 13 (1): [c]
Ostatni ślad pierwotny: [c]

15. Konfiguracja (3): 1 2 8
Zdarzenia umożliwone (2): a d
Ślad z konfiguracji 14 (1): [g]

Ostatni ślad pierwotny: [g]

16. Konfiguracja (3): 1 2 9
Zdarzenia umożliwone (2): a d
Ślad z konfiguracji 14 (1): [h]
Ostatni ślad pierwotny: [h]

17. Konfiguracja (3): 2 3 9
Zdarzenia umożliwone (1): d
Ślad z konfiguracji 16 (2): [a;h]
Ostatni ślad pierwotny: [h]

18. Konfiguracja (3): 1 11 6
Zdarzenia umożliwone (2): a f
Ślad z konfiguracji 8 (1): [e]
Ostatni ślad pierwotny: [e]

19. Konfiguracja (3): 1 2 7
Zdarzenia umożliwone (4): a d g h
Ślad z konfiguracji 18 (1): [f]
Ostatni ślad pierwotny: [f]

20. Konfiguracja (3): 1 4 7
Zdarzenia umożliwone (3): a g h
Ślad z konfiguracji 19 (1): [d]
Ostatni ślad pierwotny: [d]

2. Mutex - redukcja efektywna

Graf konfiguracji. Sieć:
Mutex (5 & 6 - critical section) - see Mutex_2
Krotność procesów: 1

Ilość tranzycji: 33

Ilość węzłów: 20

1. Konfiguracja (3): 1 2 7
Zdarzenia umożliwone (4): a d g h
Ślad z konfiguracji 0 (0): [t0]
Ostatni ślad pierwotny: [t0]

2. Konfiguracja (3): 2 3 7
Zdarzenia umożliwone (3): d g h
Ślad z konfiguracji 1 (1): [a]
Ostatni ślad pierwotny: [a]

3. Konfiguracja (3): 3 4 7

Zdarzenia umożliwone (2): g h
Ślad z konfiguracji 2 (2): [a;d]
Ostatni ślad pierwotny: [a]

4. Konfiguracja (3): 3 4 8
Zdarzenia umożliwone (1): b
Ślad z konfiguracji 3 (3): [a;d;g]
Ostatni ślad pierwotny: [a]

5. Konfiguracja (3): 10 4 5
Zdarzenia umożliwone (1): c
Ślad z konfiguracji 4 (2): [b;d]
Ostatni ślad pierwotny: [a]

6. Konfiguracja (3): 1 4 7
Zdarzenia umożliwone (1): a
Ślad z konfiguracji 5 (2): [c;d]
Ostatni ślad pierwotny: [a]

7. Konfiguracja (3): 3 4 9
Zdarzenia umożliwone (1): e
Ślad z konfiguracji 3 (3): [a;d;h]
Ostatni ślad pierwotny: [a]

8. Konfiguracja (3): 11 3 6
Zdarzenia umożliwone (1): f
Ślad z konfiguracji 7 (2): [a;e]
Ostatni ślad pierwotny: [a]

9. Konfiguracja (3): 2 3 7
Zdarzenia umożliwone (3): d g h
Ślad z konfiguracji 8 (2): [a;f]
Ostatni ślad pierwotny: [a]

10. Konfiguracja (3): 2 3 8
Zdarzenia umożliwone (1): b
Ślad z konfiguracji 9 (2): [a;g]
Ostatni ślad pierwotny: [a]

11. Konfiguracja (3): 10 2 5
Zdarzenia umożliwone (1): c
Ślad z konfiguracji 10 (1): [b]
Ostatni ślad pierwotny: [b]

12. Konfiguracja (3): 1 2 7
Zdarzenia umożliwone (3): a g h
Ślad z konfiguracji 11 (1): [c]

Ostatni ślad pierwotny: [c]

13. Konfiguracja (3): 1 2 8
Zdarzenia umożliwone (1): a
Ślad z konfiguracji 12 (1): [g]
Ostatni ślad pierwotny: [g]

14. Konfiguracja (3): 1 2 9
Zdarzenia umożliwone (1): d
Ślad z konfiguracji 12 (1): [h]
Ostatni ślad pierwotny: [h]

15. Konfiguracja (3): 1 4 9
Zdarzenia umożliwone (1): e
Ślad z konfiguracji 14 (2): [d;h]
Ostatni ślad pierwotny: [h]

16. Konfiguracja (3): 1 11 6
Zdarzenia umożliwone (1): f
Ślad z konfiguracji 15 (1): [e]
Ostatni ślad pierwotny: [e]

17. Konfiguracja (3): 1 2 7
Zdarzenia umożliwone (3): d g h
Ślad z konfiguracji 16 (1): [f]
Ostatni ślad pierwotny: [f]

18. Konfiguracja (3): 1 4 7
Zdarzenia umożliwone (2): g h
Ślad z konfiguracji 17 (1): [d]
Ostatni ślad pierwotny: [d]

19. Konfiguracja (3): 1 4 8
Zdarzenia umożliwone (1): a
Ślad z konfiguracji 18 (2): [d;g]
Ostatni ślad pierwotny: [d]

20. Konfiguracja (3): 2 3 9
Zdarzenia umożliwone (1): d
Ślad z konfiguracji 9 (2): [a;h]
Ostatni ślad pierwotny: [a]

3. Mutex - redukcja idealna

Graf konfiguracji. Sieć:
Mutex (5 & 6 - critical section) - see Mutex_2
Krotność procesów: 1

Ilość tranzycji: 33

Ilość węzłów: 9

1. Konfiguracja (3): 1 2 7
   Zdarzenia umożliwone (4): a d g h
   Ślad z konfiguracji 0 (0): [t0]
   Ostatni ślad pierwotny: [t0]

2. Konfiguracja (3): 2 3 7
   Zdarzenia umożliwone (3): d g h
   Ślad z konfiguracji 1 (1): [a]
   Ostatni ślad pierwotny: [a]

3. Konfiguracja (3): 10 2 5
   Zdarzenia umożliwone (1): c
   Ślad z konfiguracji 10 (1): [b]
   Ostatni ślad pierwotny: [b]

4. Konfiguracja (3): 1 2 7
   Zdarzenia umożliwone (3): a g h
   Ślad z konfiguracji 11 (1): [c]
   Ostatni ślad pierwotny: [c]

5. Konfiguracja (3): 1 2 8
   Zdarzenia umożliwone (1): a
   Ślad z konfiguracji 12 (1): [g]
   Ostatni ślad pierwotny: [g]

6. Konfiguracja (3): 1 2 9
   Zdarzenia umożliwone (1): d
   Ślad z konfiguracji 12 (1): [h]
   Ostatni ślad pierwotny: [h]

7. Konfiguracja (3): 1 11 6
   Zdarzenia umożliwone (1): f
   Ślad z konfiguracji 15 (1): [e]
   Ostatni ślad pierwotny: [e]

8. Konfiguracja (3): 1 2 7
   Zdarzenia umożliwone (3): d g h
   Ślad z konfiguracji 16 (1): [f]
   Ostatni ślad pierwotny: [f]

9. Konfiguracja (3): 1 4 7
   Zdarzenia umożliwone (2): g h
   Ślad z konfiguracji 17 (1): [d]

Ostatni ślad pierwotny: [d]

4. Toy 2 (3 procesy) - bez redukcji

Graf konfiguracji. Sieć:
Toy 2 - global state space = 2^k states, after reduction k+1 states
Krotność procesów: 3

Ilość tranzycji: 12

Ilość węzłów: 8

1. Konfiguracja (3): 1.1 1.2 1.3
   Zdarzenia umożliwone (3): a.1 a.2 a.3
   Ślad z konfiguracji 0 (0): [t0]
   Ostatni ślad pierwotny: [t0]

2. Konfiguracja (3): k+1.1 1.2 1.3
   Zdarzenia umożliwone (2): a.2 a.3
   Ślad z konfiguracji 1 (1): [a.1]
   Ostatni ślad pierwotny: [a.1]

3. Konfiguracja (3): k+1.1 k+1.2 1.3
   Zdarzenia umożliwone (1): a.3
   Ślad z konfiguracji 2 (2): [a.1;a.2]
   Ostatni ślad pierwotny: [a.1]

4. Konfiguracja (3): k+1.1 k+1.2 k+1.3
   Zdarzenia umożliwone (0):
   Ślad z konfiguracji 3 (3): [a.1;a.2;a.3]
   Ostatni ślad pierwotny: [a.1]

5. Konfiguracja (3): k+1.1 1.2 k+1.3
   Zdarzenia umożliwone (1): a.2
   Ślad z konfiguracji 2 (2): [a.1;a.3]
   Ostatni ślad pierwotny: [a.1]

6. Konfiguracja (3): 1.1 k+1.2 1.3
   Zdarzenia umożliwone (2): a.1 a.3
   Ślad z konfiguracji 1 (1): [a.2]
   Ostatni ślad pierwotny: [a.2]

7. Konfiguracja (3): 1.1 k+1.2 k+1.3
   Zdarzenia umożliwone (1): a.1
   Ślad z konfiguracji 6 (2): [a.2;a.3]
   Ostatni ślad pierwotny: [a.2]

8. Konfiguracja (3): 1.1 1.2 k+1.3

Zdarzenia umożliwone (2): a.1 a.2
Ślad z konfiguracji 1 (1): [a.3]
Ostatni ślad pierwotny: [a.3]

5. Toy 2 (3 procesy) - redukcja efektywna (jak idealna)

Graf konfiguracji. Sieć:
Toy 2 - global state space = 2^k states, after reduction k+1 states
Krotność procesów: 3

Ilość tranzycji: 3

Ilość węzłów: 4

1. Konfiguracja (3): 1.1 1.2 1.3
   Zdarzenia umożliwone (3): a.1 a.2 a.3
   Ślad z konfiguracji 0 (0): [t0]
   Ostatni ślad pierwotny: [t0]

2. Konfiguracja (3): k+1.1 1.2 1.3
   Zdarzenia umożliwone (0):
   Ślad z konfiguracji 1 (1): [a.1]
   Ostatni ślad pierwotny: [a.1]

3. Konfiguracja (3): 1.1 k+1.2 1.3
   Zdarzenia umożliwone (0):
   Ślad z konfiguracji 1 (1): [a.2]
   Ostatni ślad pierwotny: [a.2]

4. Konfiguracja (3): 1.1 1.2 k+1.3
   Zdarzenia umożliwone (0):
   Ślad z konfiguracji 1 (1): [a.3]
   Ostatni ślad pierwotny: [a.3]

6. Toy 3 (3 procesy) - bez redukcji

Graf konfiguracji. Sieć:
Toy 3 - global state space = 3^k states, after reduction 2k^2 states
Krotność procesów: 3

Ilość tranzycji: 58

Ilość węzłów: 28

1. Konfiguracja (3): 1.1 1.2 1.3
   Zdarzenia umożliwone (3): a.1 a.2 a.3
   Ślad z konfiguracji 0 (0): [t0]
   Ostatni ślad pierwotny: [t0]

2. Konfiguracja (3): k+1.1 1.2 1.3
   Zdarzenia umożliwione (3): ak+1.1 a.2 a.3
   Ślad z konfiguracji 1 (1): [a.1]
   Ostatni ślad pierwotny: [a.1]

3. Konfiguracja (3): 2k+1.1 1.2 1.3
   Zdarzenia umożliwione (2): a.2 a.3
   Ślad z konfiguracji 2 (1): [ak+1.1]
   Ostatni ślad pierwotny: [ak+1.1]

4. Konfiguracja (3): 2k+1.1 k+1.2 1.3
   Zdarzenia umożliwione (2): ak+1.2 a.3
   Ślad z konfiguracji 3 (2): [ak+1.1;a.2]
   Ostatni ślad pierwotny: [ak+1.1]

5. Konfiguracja (3): 2k+1.1 2k+1.2 1.3
   Zdarzenia umożliwione (1): a.3
   Ślad z konfiguracji 4 (2): [ak+1.1;ak+1.2]
   Ostatni ślad pierwotny: [ak+1.1]

6. Konfiguracja (3): 2k+1.1 2k+1.2 k+1.3
   Zdarzenia umożliwione (1): ak+1.3
   Ślad z konfiguracji 5 (3): [ak+1.1;ak+1.2;a.3]
   Ostatni ślad pierwotny: [ak+1.1]

7. Konfiguracja (3): 2k+1.1 2k+1.2 2k+1.3
   Zdarzenia umożliwione (1): a2k+1
   Ślad z konfiguracji 6 (3): [ak+1.1;ak+1.2;ak+1.3]
   Ostatni ślad pierwotny: [ak+1.1]

8. Konfiguracja (3): 1.1 1.2 1.3
   Zdarzenia umożliwione (3): a.1 a.2 a.3
   Ślad z konfiguracji 7 (1): [a2k+1]
   Ostatni ślad pierwotny: [a2k+1]

9. Konfiguracja (3): 1.1 k+1.2 1.3
   Zdarzenia umożliwione (3): a.1 ak+1.2 a.3
   Ślad z konfiguracji 8 (1): [a.2]
   Ostatni ślad pierwotny: [a.2]

10. Konfiguracja (3): k+1.1 k+1.2 1.3
    Zdarzenia umożliwione (3): ak+1.1 ak+1.2 a.3
    Ślad z konfiguracji 9 (2): [a.1;a.2]
    Ostatni ślad pierwotny: [a.2]

11. Konfiguracja (3): k+1.1 2k+1.2 1.3
    Zdarzenia umożliwione (2): ak+1.1 a.3

Ślad z konfiguracji 10 (2): [a.1;ak+1.2]
Ostatni ślad pierwotny: [a.2]

12. Konfiguracja (3): k+1.1 2k+1.2 k+1.3
Zdarzenia umożliwone (2): ak+1.1 ak+1.3
Ślad z konfiguracji 11 (3): [a.1;ak+1.2;a.3]
Ostatni ślad pierwotny: [a.2]

13. Konfiguracja (3): k+1.1 2k+1.2 2k+1.3
Zdarzenia umożliwone (1): ak+1.1
Ślad z konfiguracji 12 (3): [a.1;ak+1.2;ak+1.3]
Ostatni ślad pierwotny: [a.2]

14. Konfiguracja (3): k+1.1 k+1.2 k+1.3
Zdarzenia umożliwone (3): ak+1.1 ak+1.2 ak+1.3
Ślad z konfiguracji 10 (3): [a.1;a.2;a.3]
Ostatni ślad pierwotny: [a.2]

15. Konfiguracja (3): 2k+1.1 k+1.2 k+1.3
Zdarzenia umożliwone (2): ak+1.2 ak+1.3
Ślad z konfiguracji 14 (3): [ak+1.1;a.2;a.3]
Ostatni ślad pierwotny: [a.2]

16. Konfiguracja (3): 2k+1.1 k+1.2 2k+1.3
Zdarzenia umożliwone (1): ak+1.2
Ślad z konfiguracji 15 (3): [ak+1.1;a.2;ak+1.3]
Ostatni ślad pierwotny: [a.2]

17. Konfiguracja (3): k+1.1 k+1.2 2k+1.3
Zdarzenia umożliwone (2): ak+1.1 ak+1.2
Ślad z konfiguracji 14 (3): [a.1;a.2;ak+1.3]
Ostatni ślad pierwotny: [a.2]

18. Konfiguracja (3): 1.1 2k+1.2 1.3
Zdarzenia umożliwone (2): a.1 a.3
Ślad z konfiguracji 9 (1): [ak+1.2]
Ostatni ślad pierwotny: [ak+1.2]

19. Konfiguracja (3): 1.1 2k+1.2 k+1.3
Zdarzenia umożliwone (2): a.1 ak+1.3
Ślad z konfiguracji 18 (2): [ak+1.2;a.3]
Ostatni ślad pierwotny: [ak+1.2]

20. Konfiguracja (3): 1.1 2k+1.2 2k+1.3
Zdarzenia umożliwone (1): a.1
Ślad z konfiguracji 19 (2): [ak+1.2;ak+1.3]
Ostatni ślad pierwotny: [ak+1.2]

21. Konfiguracja (3): 1.1 k+1.2 k+1.3
    Zdarzenia umożliwone (3): a.1 ak+1.2 ak+1.3
    Ślad z konfiguracji 9 (2): [a.2;a.3]
    Ostatni ślad pierwotny: [ak+1.2]

22. Konfiguracja (3): 1.1 k+1.2 2k+1.3
    Zdarzenia umożliwone (2): a.1 ak+1.2
    Ślad z konfiguracji 21 (2): [a.2;ak+1.3]
    Ostatni ślad pierwotny: [ak+1.2]

23. Konfiguracja (3): 1.1 1.2 k+1.3
    Zdarzenia umożliwone (3): a.1 a.2 ak+1.3
    Ślad z konfiguracji 8 (1): [a.3]
    Ostatni ślad pierwotny: [a.3]

24. Konfiguracja (3): k+1.1 1.2 k+1.3
    Zdarzenia umożliwone (3): ak+1.1 a.2 ak+1.3
    Ślad z konfiguracji 23 (2): [a.1;a.3]
    Ostatni ślad pierwotny: [a.3]

25. Konfiguracja (3): 2k+1.1 1.2 k+1.3
    Zdarzenia umożliwone (2): a.2 ak+1.3
    Ślad z konfiguracji 24 (2): [ak+1.1;a.3]
    Ostatni ślad pierwotny: [a.3]

26. Konfiguracja (3): 2k+1.1 1.2 2k+1.3
    Zdarzenia umożliwone (1): a.2
    Ślad z konfiguracji 25 (2): [ak+1.1;ak+1.3]
    Ostatni ślad pierwotny: [a.3]

27. Konfiguracja (3): k+1.1 1.2 2k+1.3
    Zdarzenia umożliwone (2): ak+1.1 a.2
    Ślad z konfiguracji 24 (2): [a.1;ak+1.3]
    Ostatni ślad pierwotny: [a.3]

28. Konfiguracja (3): 1.1 1.2 2k+1.3
    Zdarzenia umożliwone (2): a.1 a.2
    Ślad z konfiguracji 23 (1): [ak+1.3]
    Ostatni ślad pierwotny: [ak+1.3]

7. Toy 3 (3 procesy) - redukcja efektywna

Graf konfiguracji. Sieć:
Toy 3 - global state space = 3^k states, after reduction 2k^2 states
Krotność procesów: 3

Ilość tranzycji: 20

Ilość węzłów: 16

1. Konfiguracja (3): 1.1 1.2 1.3
   Zdarzenia umożliwone (3): a.1 a.2 a.3
   Ślad z konfiguracji 0 (0): [t0]
   Ostatni ślad pierwotny: [t0]

2. Konfiguracja (3): k+1.1 1.2 1.3
   Zdarzenia umożliwone (1): ak+1.1
   Ślad z konfiguracji 1 (1): [a.1]
   Ostatni ślad pierwotny: [a.1]

3. Konfiguracja (3): 2k+1.1 1.2 1.3
   Zdarzenia umożliwone (1): a.2
   Ślad z konfiguracji 2 (1): [ak+1.1]
   Ostatni ślad pierwotny: [ak+1.1]

4. Konfiguracja (3): 2k+1.1 k+1.2 1.3
   Zdarzenia umożliwone (1): ak+1.2
   Ślad z konfiguracji 3 (2): [ak+1.1;a.2]
   Ostatni ślad pierwotny: [ak+1.1]

5. Konfiguracja (3): 2k+1.1 2k+1.2 1.3
   Zdarzenia umożliwone (1): a.3
   Ślad z konfiguracji 4 (2): [ak+1.1;ak+1.2]
   Ostatni ślad pierwotny: [ak+1.1]

6. Konfiguracja (3): 2k+1.1 2k+1.2 k+1.3
   Zdarzenia umożliwone (1): ak+1.3
   Ślad z konfiguracji 5 (3): [ak+1.1;ak+1.2;a.3]
   Ostatni ślad pierwotny: [ak+1.1]

7. Konfiguracja (3): 2k+1.1 2k+1.2 2k+1.3
   Zdarzenia umożliwone (1): a2k+1
   Ślad z konfiguracji 6 (3): [ak+1.1;ak+1.2;ak+1.3]
   Ostatni ślad pierwotny: [ak+1.1]

8. Konfiguracja (3): 1.1 1.2 1.3
   Zdarzenia umożliwone (3): a.1 a.2 a.3
   Ślad z konfiguracji 7 (1): [a2k+1]
   Ostatni ślad pierwotny: [a2k+1]

9. Konfiguracja (3): 1.1 k+1.2 1.3
   Zdarzenia umożliwone (1): ak+1.2
   Ślad z konfiguracji 8 (1): [a.2]
   Ostatni ślad pierwotny: [a.2]

10. Konfiguracja (3): 1.1 2k+1.2 1.3
    Zdarzenia umożliwone (1): a.1
    Ślad z konfiguracji 9 (1): [ak+1.2]
    Ostatni ślad pierwotny: [ak+1.2]

11. Konfiguracja (3): k+1.1 2k+1.2 1.3
    Zdarzenia umożliwone (1): ak+1.1
    Ślad z konfiguracji 10 (2): [a.1;ak+1.2]
    Ostatni ślad pierwotny: [ak+1.2]

12. Konfiguracja (3): 1.1 1.2 k+1.3
    Zdarzenia umożliwone (1): ak+1.3
    Ślad z konfiguracji 8 (1): [a.3]
    Ostatni ślad pierwotny: [a.3]

13. Konfiguracja (3): 1.1 1.2 2k+1.3
    Zdarzenia umożliwone (1): a.1
    Ślad z konfiguracji 12 (1): [ak+1.3]
    Ostatni ślad pierwotny: [ak+1.3]

14. Konfiguracja (3): k+1.1 1.2 2k+1.3
    Zdarzenia umożliwone (1): ak+1.1
    Ślad z konfiguracji 13 (2): [a.1;ak+1.3]
    Ostatni ślad pierwotny: [ak+1.3]

15. Konfiguracja (3): 2k+1.1 1.2 2k+1.3
    Zdarzenia umożliwone (1): a.2
    Ślad z konfiguracji 14 (2): [ak+1.1;ak+1.3]
    Ostatni ślad pierwotny: [ak+1.3]

16. Konfiguracja (3): 2k+1.1 k+1.2 2k+1.3
    Zdarzenia umożliwone (1): ak+1.2
    Ślad z konfiguracji 15 (3): [ak+1.1;a.2;ak+1.3]
    Ostatni ślad pierwotny: [ak+1.3]

8. Toy 3 (3 procesy) - redukcja idealna

Graf konfiguracji. Sieć:
Toy 3 - global state space = 3^k states, after reduction 2k^2 states
Krotność procesów: 3

Ilość tranzycji: 20

Ilość węzłów: 8

1. Konfiguracja (3): 1.1 1.2 1.3
   Zdarzenia umożliwone (3): a.1 a.2 a.3

Ślad z konfiguracji 0 (0): [t0]
Ostatni ślad pierwotny: [t0]

2. Konfiguracja (3): k+1.1 1.2 1.3
Zdarzenia umożliwone (1): ak+1.1
Ślad z konfiguracji 1 (1): [a.1]
Ostatni ślad pierwotny: [a.1]

3. Konfiguracja (3): 2k+1.1 1.2 1.3
Zdarzenia umożliwone (1): a.2
Ślad z konfiguracji 2 (1): [ak+1.1]
Ostatni ślad pierwotny: [ak+1.1]

4. Konfiguracja (3): 1.1 1.2 1.3
Zdarzenia umożliwone (3): a.1 a.2 a.3
Ślad z konfiguracji 7 (1): [a2k+1]
Ostatni ślad pierwotny: [a2k+1]

5. Konfiguracja (3): 1.1 k+1.2 1.3
Zdarzenia umożliwone (1): ak+1.2
Ślad z konfiguracji 8 (1): [a.2]
Ostatni ślad pierwotny: [a.2]

6. Konfiguracja (3): 1.1 2k+1.2 1.3
Zdarzenia umożliwone (1): a.1
Ślad z konfiguracji 9 (1): [ak+1.2]
Ostatni ślad pierwotny: [ak+1.2]

7. Konfiguracja (3): 1.1 1.2 k+1.3
Zdarzenia umożliwone (1): ak+1.3
Ślad z konfiguracji 8 (1): [a.3]
Ostatni ślad pierwotny: [a.3]

8. Konfiguracja (3): 1.1 1.2 2k+1.3
Zdarzenia umożliwone (1): a.1
Ślad z konfiguracji 12 (1): [ak+1.3]
Ostatni ślad pierwotny: [ak+1.3]

3 Listing programu

Zawartość pliku źródłowego programu została dołączona na końcu pracy.