Politechnika Rzeszowska
Wydział Elektrotechniki i Informatyki
Katedra Informatyki i Automatyki

Laboratorium

Sterowanie procesami dyskretnymi

Stanowisko 4

Modelowanie systemów produkcyjnych
z użyciem czasowych sieci Petriego
– oprogramowanie CPN Tools

Rzeszów 2012

1. Oprogramowanie CPN Tools

Oprogramowanie CPN Tools pozwala na wygodne wykorzystanie formalizmu hierarchicznych czasowych kolorowanych sieci Petrigo do modelowania, symulowania i analizy systemów dyskretnych. W bieżącym ćwiczeniu oprogramowanie CPN Tools zostanie użyte do modelowania i symulacji systemów produkcyjnych.

1.1. Formalizm HTCPN

Pakiet CPN Tools wspiera formalizm hierarchicznych kolorowanych czasowych sieci Petriego (*Hierarchical Timed Coloured Petri Nets*, HTCPN). Formalizm HTCPN stanowi jedną z odmian sieci Petriego wysokiego poziomu, które powstały jako modyfikacje i rozszerzenia podstawowej struktury sieci Petriego (tzw. sieci miejsc i przejść, P/T-net). Formalizm HTCPN posiada trzy rozszerzenia wysokiego poziomu:

- hierarchiczność,
- kolorowanie,
- reprezentacja czasu.

Hierarchiczność pozwala na budowanie wielopoziomowych modeli sieciowych, w których pojedynczy element (tranzycja) na danym poziomie struktury reprezentowany jest na poziomie podrzędnym przez zagnieżdżoną podstrukturę. Hierarchiczność nie będzie wykorzystywana w modelach tworzonych w ramach ćwiczenia.

Sieci kolorowane posiadają zdefiniowane typy (zwane *zbiorami kolorów*). Są to typy podobne do tych używanych w strukturalnych językach programowania (*int*, *string*, *bool*, listy, rekordy, krotki itp.). Każdemu miejscu przypisany jest jeden ze zbiorów kolorów. Znaczniki reprezentują wartości (*kolory*) zgodne z typami miejsc, w których się znajdują. Opisy łuków (elementów łączących miejsca i tranzycje) także są wyrażeniami typowanymi. Jeżeli wartość wyrażenia łuku wychodzącego z pewnego miejsca sieci reprezentuje dowolny podzbiór znaczników obecnych w tym miejscu, mówimy

że wystąpiło *wiązanie* (*binding*) wartości wyrażenia łuku. Gdy wszystkie łuki wejściowe wybranej tranzycji posiadają wiązanie wyrażeń z połączonymi z nimi miejscami, tranzycja taka jest *przygotowana do wykonania*. *Wykonanie* tranzycji powoduje usunięcie ze wszystkich jej miejsc wejściowych podzbiorów znaczników określonych przez wartości wyrażeń łuków wejściowych i dodanie do wszystkich miejsc wyjściowych zbiorów znaczników określonych przez wartości wyrażeń łuków wyjściowych. Kolorowanie będzie wykorzystywane w modelach tworzonych w ramach ćwiczenia w niewielkim zakresie.

Sieci czasowe zawierają rozszerzenia pozwalające na włączenie do modelu reprezentacji czasu, co umożliwia synchronizację i uporządkowanie zdarzeń (wykonań tranzycji) według osi czasu. Istnieje wiele wariantów reprezentacji czasu w sieciach Petriego. W przypadku formalizmu HTCPN reprezentacja ta sprowadza się do dwóch elementów:

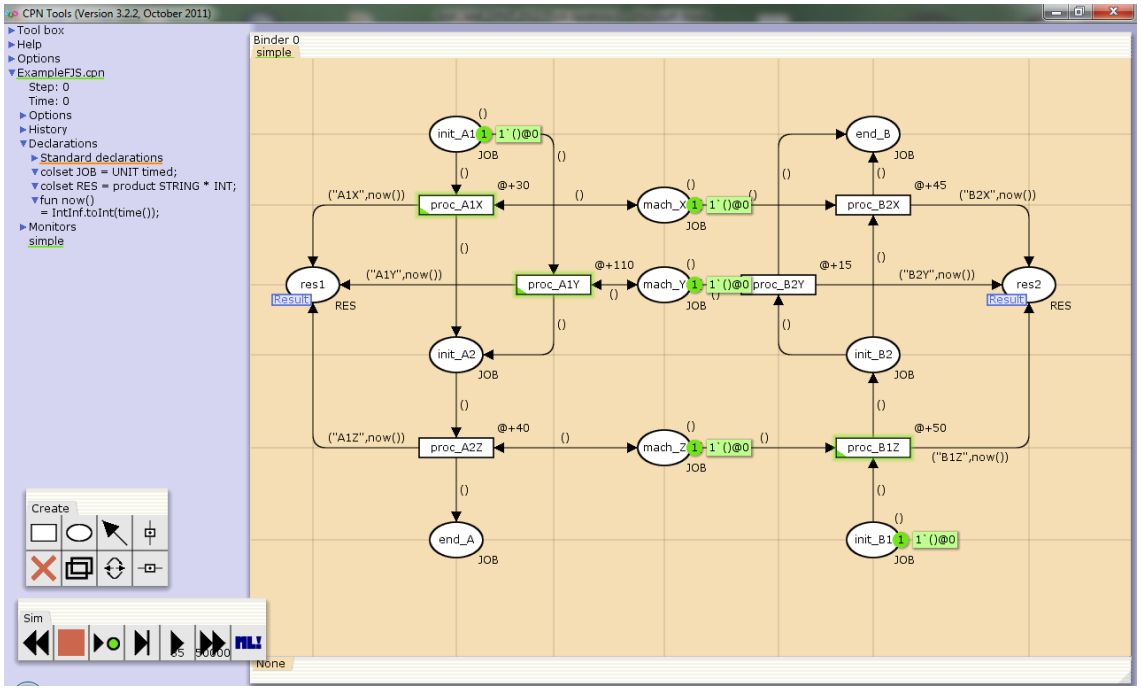
- *zegara globalnego* (*global clock*) – jest to zmienna wspólna dla całego modelu, przyjmująca wartości nieujemne, której wartość może jedynie wzrastać,
- *pieczątek czasowych* (*timestapms*) – są to liczby nieujemne przypisywane indywidualnie poszczególnym znacznikom, znacznik może zostać pobrany z danego miejsca tylko wtedy, gdy jego pieczątka czasowa ma wartość nie większą od wartości zegara globalnego.

Synchronizujący i porządkujący mechanizm reprezentacji czasu ma charakter ograniczenia globalnego, które polega na tym, że dla każdej wartości zegara globalnego wykonywane są wszystkie przygotowane tranzycie i dopiero po ich wykonaniu następuje zwiększenie wartości zegara globalnego o najmniejszą możliwą wartość, która gwarantuje przygotowanie co najmniej jednej kolejnej tranzycji. Wykonanie tranzycji powoduje wprowadzenie do jej miejsc wyjściowych nowych znaczników wraz z pieczątkami czasowymi. Wartości pieczątek czasowych mogą być wyznaczane dowolnie przez wyrażenia łuków wyjściowych, jednak w praktyce najczęściej stosuje się notację tzw. *opóźnienia*, stowarzyszonego z całą tranzycją lub indywidualnymi łukami, która polega na utworzeniu pieczątek czasowych o wartościach równych sumie

wartości zegara globalnego w momencie wykonania tranzycji i zadanej wartości opóźnienia. Reprezentacja czasu będzie wykorzystywana w modelach tworzonych w ramach ćwiczenia, aby ulokować na osi czasu zdarzenia symulowane w systemie produkcyjnym (momenty rozpoczęcia operacji).

1.2. Obsługa oprogramowania

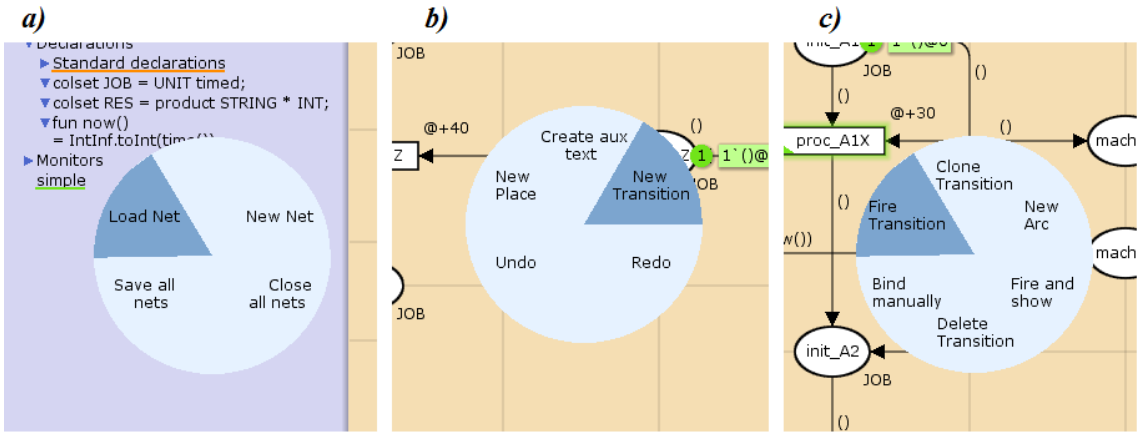
Modele sieciowe utworzone w programie CPN Tools mogą być zapisywane w postaci projektów (pliki z rozszerzeniem **cpn**). Możliwe jest otwarcie samego programu za pomocą jego ikony lub programu wraz z załadowanym wybranym projektem za pomocą ikony projektu (rys. 1.1).



Rys. 1.1. Okno programu CPN Tool z otwartym przykładowym projektem

1.2.1. Wykonywanie komend

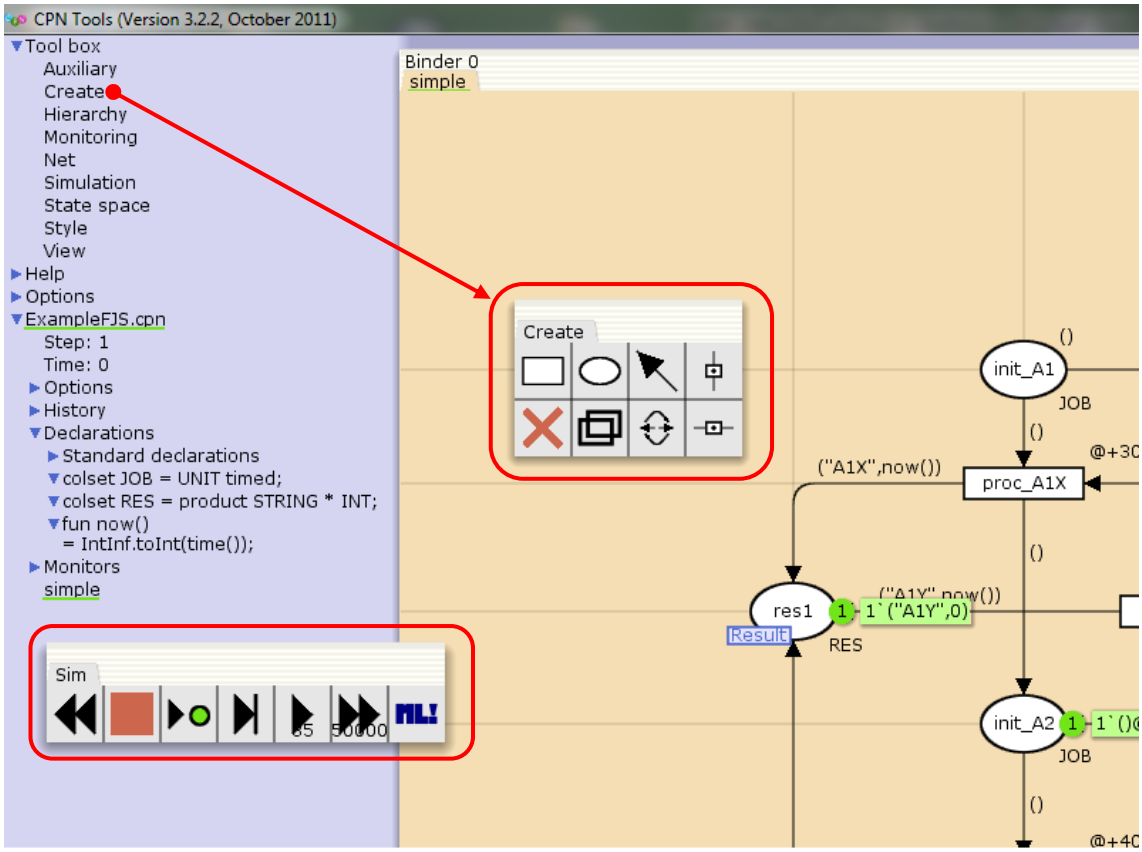
W programie CPN Tool do wykonywania poleceń udostępniono dwa główne mechanizmy. Pierwszym z nich jest mechanizm menu kontekstowego. Menu kontekstowe pojawia się po wskazaniu kursorem odpowiedniego obszaru lub elementu graficznego i naciśnięciu (oraz przytrzymaniu) prawego przycisku myszy (rys. 1.2).



Rys. 1.2. Przykłady menu kontekstowego: a) dla panelu głównego, b) po wskazaniu pustego miejsca na karcie edycji modelu, c) po wskazaniu przygotowanej tranzycji

Menu kontekstowe ma formę koła z wpisanymi opcjami, które mogą zostać wybrane. Opcje zależą od wskazanego elementu, np. dla panelu głównego dostępne są opcje związane z tworzeniem, otwieraniem, zapisywaniem i zamykaniem modeli (rys. 1.2a), dla pustego miejsca na karcie edycji modelu pojawia się menu z opcjami m.in. wstawienia nowych elementów (rys. 1.2b), po wskazaniu przygotowanej tranzycji dostępne są m.in. opcje wiązania manualnego i wykonania tranzycji. **Prawy przycisk myszy należy trzymać cały czas wciśnięty, aż do wskazania wybranej opcji. Zwolnienie przycisku zamyka menu i powoduje wybranie wskazanej opcji.** Można zrezygnować z wyboru, ustawiając kursor przed zwolnieniem przycisku tak, aby żadna opcja nie była podświetlona (w środku koła lub na zewnątrz). Należy przećwiczyć obsługę menu kontekstowego, gdyż jest ona inna niż standardowo w systemie Windows.

Drugi mechanizm wykonywania komend oparty jest na paskach ikon narzędziowych (rys. 1.3). Dostępne paski ikon odpowiadają wpisom widocznym po rozwinięciu w głównym panelu gałęzi **Tool box**, są to paski ikon: **Auxiliary**, **Create**, **Hierarchy** itd. Otwarcie wybranego paska ikon odbywa się poprzez wskazanie jego nazwy i przeciągnięcie jej (trzymając wciśnięty lewy klawisz myszy) na dowolną kartę edycji (np. kartę edycji modelu). Zaleca się, aby w trakcie wykonywania ćwiczenia otwarte były paski **Create** oraz **Simulation**, gdyż zawierają one podstawowe potrzebne narzędzia.



Rys. 1.3. Paski ikon narzędziowych

Wybór określonego narzędzia z paska odbywa się poprzez kliknięcie na nim lewym przyciskiem myszy. Gdy narzędzie jest aktywne, jego ikona przemieszcza się obok kursora myszy. Wykonanie polecenia związanego z ikoną odbywa się przez kliknięcie na odpowiednim elemencie, dla którego wybrana opcja ma sens. Raz wybrane narzędzie może być używane wielokrotnie, aż do jego odłożenia. Odłożenie następuje poprzez: naciśnięcie klawisza **ESC**, ponowne kliknięcie ikony narzędzia na pasku narzędzi lub wybór innego narzędzia.

1.2.2. Edycja modelu sieciowego

W edycji modelu sieci Petriego przy użyciu oprogramowania CPN Tools można wyróżnić trzy rodzaje czynności, które zazwyczaj wzajemnie się przeplatają:

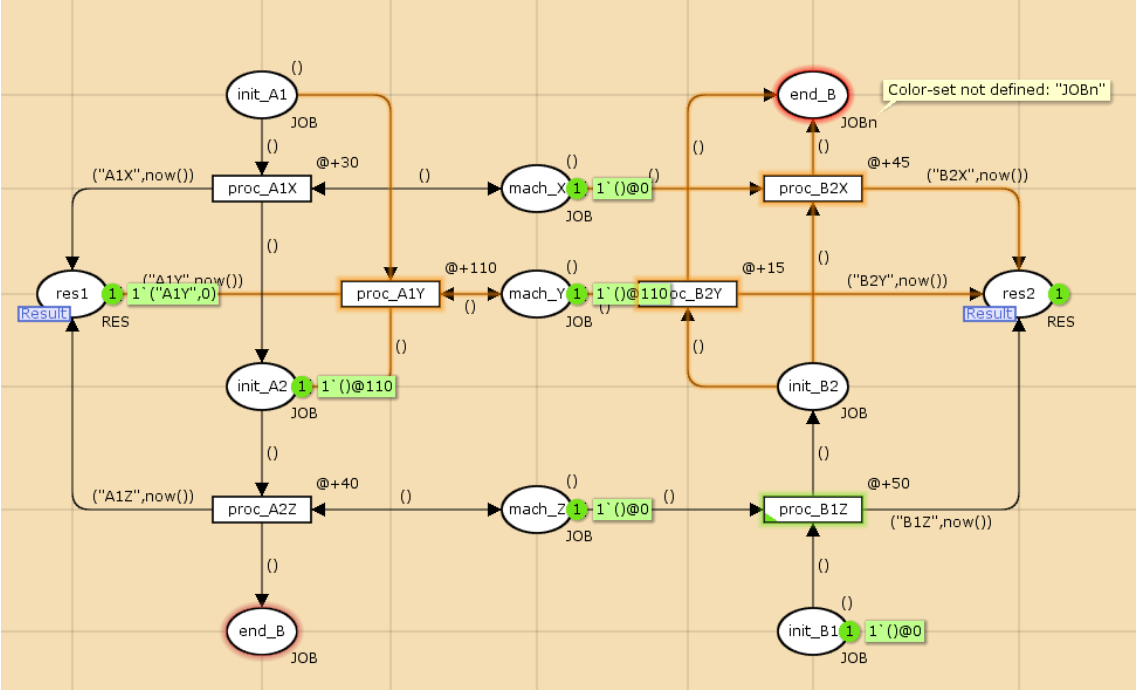
a) Tworzenie deklaracji dla modelu. Deklaracje w formalizmie HTCPN są zapisami w języku SML (*Standard Meta Language*), obejmują one m.in.:

zbiory kolorów, stałe, zmienne, funkcje. W programie CPN Tools są one umieszczane w głównym panelu w sekcji **Declarations**. Tworzenie deklaracji wykracza poza ramy niniejszego ćwiczenia. W edytowanych modelach wykorzystywane będę tylko gotowe deklaracje, pochodzące z projektu przykładowego.

b) Edycja struktury modelu, tzn. tworzenie miejsc i tranzycji oraz łączenie ich odpowiednio skierowanymi łukami. W pakiecie CPN Tools czynności te mają charakter intuicyjny. Aby utworzyć nowe miejsce lub tranzycję, należy albo wybrać odpowiednie narzędzie z paska **Create**, albo opcję z menu kontekstowego (rys. 1.2b). Podobnie w przypadku tworzenia łuków, przy czym opcja z menu kontekstowego jest w tym przypadku wygodniejsza: należy wskazać element, z którego łuk ma wychodzić, wybrać opcję **New Arc**, wskazać element docelowy i kliknąć na niego. Zwrot raz utworzonego łuku można modyfikować, w tym także uczynić go dwukierunkowym, wybierając opcję menu kontekstowego **Change Direction**.

c) Edycja inskrypcji, tzn. wyrażeń przypisanych elementom struktury sieciowej. Są one reprezentowane w języku SML. Pierwsze kliknięcie lewym klawiszem myszy na wybrany element (miejsce, tranzycję, łuk) powoduje wejście w tryb edycji inskrypcji tego elementu. Za pomocą drugiego kliknięcia można dokładnie ustawić kursor w tekście inskrypcji. Miejsca i tranzycje mają po kilka inskrypcji (np. miejsca mają inskrypcje definiujące nazwę, zbiór kolorów oraz znakowanie początkowe), można się między nimi przemieszczać cyklicznie, wciskając klawisz **TAB**.

W czasie edycji modelu sieciowego w oprogramowaniu CPN Tool dokonywana jest na bieżąco weryfikacja poprawności struktury po każdej wprowadzonej zmianie. Niekompletność modelu lub błędy wyświetlane są za pomocą odpowiednich oznaczeń i komunikatów (rys. 1.4).



Rys. 1.4. CPN Tools – ostrzeżenia i błędy

Jeżeli nazwa jakiegoś elementu struktury powtarza się (np. miejsce dolne end_B, rys. 1.4.), następuje podświetlenie tego elementu kolorem różowym. Jest to rodzaj ostrzeżenia – struktura będzie działać poprawnie, ale dublowanie nazw nie jest wskazane.

Jeśli nie dla wszystkich łuków powiązanych z daną tranzycją zostały wprowadzone inskrypcje (np. łuk z miejsca init_A1 do tranzycji proc_A1Y nie ma inskrypcji, rys. 1.4), to tranzycja taka nie może zostać przeanalizowana pod względem przygotowania do wykonania. Jest ona wtedy oznaczana kolorem brązowym, wraz ze wszystkimi połączonymi z nią łukami, z którymi stanowi nieaktywny (nieuwzględniany w symulacji) fragment struktury.

Jeżeli występują różnego rodzaju błędy składniowe, w tym błędy w inskrypcjach (np. do górnego miejsca end_B przypisano niedefiniowany zbiór kolorów JOBn, rys. 1.4), wtedy element, którego dotyczy błąd podświetlany jest kolorem czerwony i dodawany jest odpowiedni komunikat. Podstruktura sieciowa, która z powodu błędu stała się nieaktywna, oznaczana jest kolorem brązowym.

Końcowa postać tworzonego modelu nie powinna zawierać żadnych błędów ani ostrzeżeń.

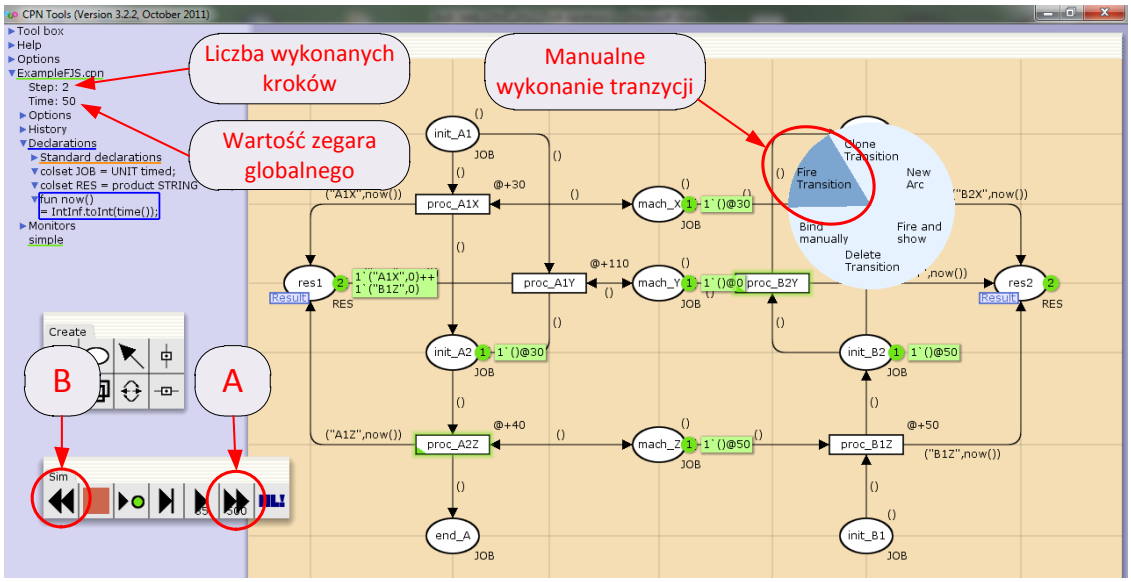
1.2.3. Symulacja modelu sieciowego

Jedną z funkcjonalności oprogramowania CPN Tools jest symulacja modelu sieciowego z bezpośrednim wyświetlaniem rezultatów w postaci graficznej, naniesionych na strukturę modelu. Symulacja sieci Petriego polega na wykonywaniu kolejnych przygotowanych tranzycji. W przypadku konfliktów pomiędzy tranzycjami algorytm symulacji powinien określać sposób ich rozstrzygania. Oprogramowanie CPN Tools udostępnia dwa tryby symulacji:

a) **Tryb manualny** – jest to krokowy tryb symulacji, w którym użytkownik oprogramowania decyduje w każdym kroku o wyborze i wykonaniu tranzycji,

b) **Tryb automatyczny** – polega na automatycznym wykonaniu zadanej liczby kroków sieci bez ingerencji użytkownika, konflikty tranzycji rozstrzygane są poprzez losowy wybór jednej ze zbioru równocześnie przygotowanych tranzycji.

Możliwe jest naprzemienne używanie obu trybów. Wykonanie pojedynczej tranzycji w trybie manualnym najwygodniej jest zrealizować poprzez opcję menu kontekstowego **Fire Transition**, wywołaną dla wybranej, przygotowanej tranzycji (rys. 1.5). Tranzycje przygotowane oznaczone są zieloną obwódką.



Rys. 1.5. CPN Tools – symulacja modelu sieciowego

Symulację w trybie automatycznym najłatwiej jest uruchomić za pomocą narzędzia oznaczonego na rysunku 1.5 symbolem **A**. Jest to narzędzie konfigurowalne. Liczbowy parametr konfiguracyjny określa jaka maksymalna liczba kroków sieci ma zostać wykonana w trybie automatycznym, o ile sieć nie osiągnie wcześniej znakowania martwego. Narzędzie uruchamia się poprzez kliknięcie w dowolnym miejscu okna z modelem sieciowym.

Po każdym etapie symulacji aktualizowana jest graficzna reprezentacja modelu sieciowego tak, że pokazywane są bieżące znakowania miejsc oraz stany przygotowania tranzycji. W panelu głównym wyświetlana jest także aktualna liczba wykonanych kroków i wartość zegara globalnego.

Po każdym etapie symulacji można przywrócić znakowanie początkowe sieci, używając narzędzia oznaczonego na rysunku 1.5 symbolem **B** (kliknięcie w dowolnym miejscu okna z modelem sieciowym).

2. Modelowanie systemów produkcyjnych

Na przykładach zademonstrowane zostaną możliwości modelowania prostych struktur systemów produkcyjnych z użyciem formalizmu HTCPN i narzędzia CPN Tool.

2.1. Struktura środowiska maszynowego typu FJS

Środowisko maszynowe typu gniazdowego z maszynami alternatywnymi (*flexible job shop*, FJS) scharakteryzowane jest następująco:

a) Dany jest zbiór *J* zadań (*jobs*) i *M* maszyn.
b) Każde zadanie składa się ze ściśle zdefiniowanego ciągu operacji. Liczba i kolejność operacji może być odmienna dla każdego z zadań.
c) Dla każdej operacji zdefiniowany jest niepusty podzbiór maszyn, które mogą wykonać daną operację.
d) Dla każdej pary operacja-maszyna zdefiniowany jest czas przetwarzania.

Przykład 2.1

Dany jest problem produkcyjny typu FJS z dwoma zadaniami (*J_A*, *J_B*) i trzema maszynami (*M_X*, *M_Y*, *M_Z*), scharakteryzowany parametrami przedstawionymi w tabelach 2.1 i 2.2. Zadanie *J_A* jest wykonywane najpierw przez maszynę *M_X* lub *M_Y* (pierwsza operacja), a następnie przez maszynę *M_Z* (druga operacja). Zadanie *J_B* wykonywane jest najpierw przez maszynę *M_Z*, a następnie przez maszynę *M_X* lub *M_Y*. Czas wykonywania operacji 1 z zadania *J_A* przez maszynę *M_X* wynosi 30, czas wykonywania operacji 1 z zadania *J_A* przez maszynę *M_Y* wynosi 110 itd.

	M_X	**M_Y**	**M_Z**
J_A	1	1	2
J_B	2	2	1

Tab. 2.1. Kolejność operacji w systemie FJS

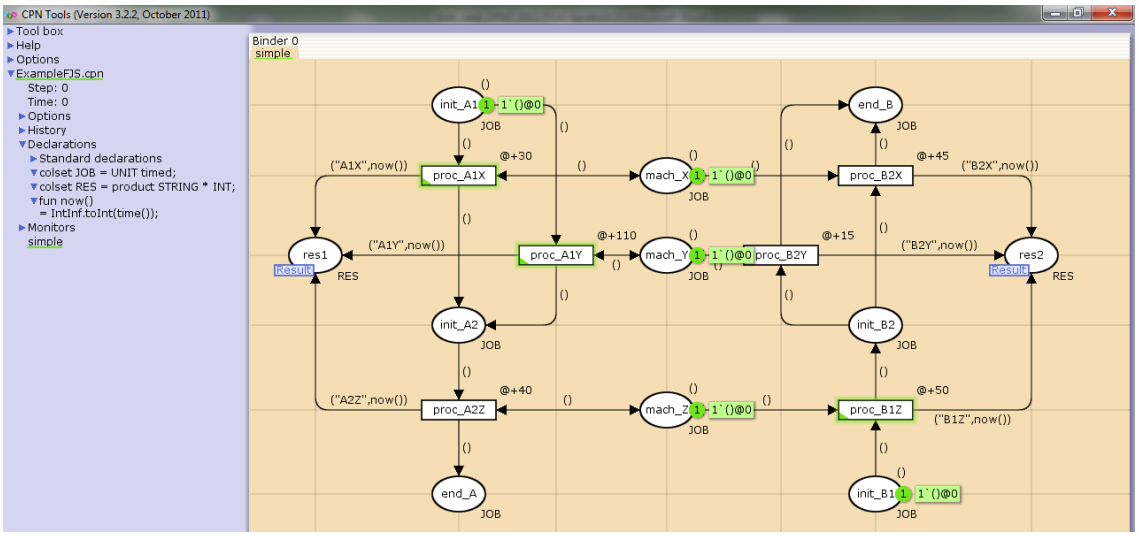
	M_X	**M_Y**	**M_Z**
J_A	30	110	40
J_B	45	15	50

Tab. 2.2. Czasy operacji w systemie FJS

Opracować model symulacyjny systemu z wykorzystaniem oprogramowania CPN Tools. Przeprowadzić symulację modelu, która pozwoli zbudować jeden (dowolny) harmonogram produkcyjny dla rozważanego systemu. Przedstawić harmonogram na wykresie Gantta.

Rozwiązanie

Model sieciowy zadanego problemu produkcyjnego, zbudowany w programie CPN Tools, został przedstawiony na rysunku 2.1.



Rys. 2.1. Model sieciowy przykładowego systemu produkcyjnego typu FJS

Dla modelu zdefiniowano dwa zbiory kolorów:

```
colset JOB = UNIT timed;
colset RES = product STRING * INT;
```

Zbiór kolorów **JOB** jest identyczny z typem podstawowym *UNIT*, z tym wyjątkiem, że dodatkowo został zdefiniowany jako typ czasowy, tzn. znaczniki tego typu mają przypisane pieczątki czasowe i dotyczą ich reguły symulacji z modelowaniem czasu, które przedstawiono punkcie 1.1. Typ *UNIT* jest najprostszym typem wbudowanym, zdefiniowanym w pakiecie CPN Tools. Typ ten zawiera tylko jedną wartość, oznaczaną pustą parą nawiasów (). Zatem znaczniki takiego nie są względem siebie rozróżnialne i mogą przenosić żadnych dodatkowych informacji, w ten sposób emulują działanie tradycyjnych „czarnych" znaczników, właściwych dla sieci Petriego niskiego poziomu.

Zbiór kolorów **RES** (*result*) został zdefiniowany jak krotka dwuelementowa, czyli uporządkowana para wartości. Pierwsza wartość jest typu łańcuch znakowego (*String*), druga posiada typ całkowitoliczbowy (*INT*). Typ ten jest przeznaczony do przechowywania informacji opisujących parametry realizacji operacji produkcyjnej (nazwa przypisanej maszyny oraz czas rozpoczęcia operacji).

Zdefiniowana została także funkcja **now()**, która zwraca wartość zegara globalnego, właściwą dla momentu jej wywołania.

Elementy struktury sieciowej modelu można podzielić na trzy grupy:

a) Miejsca z przypisanym zbiorem kolorów *JOB* – znakowanie tych miejsc reprezentuje pozycje operacji na szlaku przetwarzania. Np. znacznik w miejscu *init_A1* oznacza, że zadanie *J_A* oczekuje na operację 1, w miejscu *init_A2* oznacza, że zadanie *J_A* oczekuje na operację 2, w miejscu *end_A* oznacza, że zadanie *J_A* zostało ukończone.
b) Tranzycje – wykonania tranzycji modelują zdarzenia rozpoczęcia określonych operacji produkcyjnych.
c) Miejsca z przypisanym zbiorem kolorów *RES* – w praktyce jest jedno takie miejsce w strukturze sieciowej, ponieważ dwa miejsca **res1** i **res2**

połączone są wspólną etykietą fuzji **Result**. Fuzja o jednej nazwie grupuje miejsca, rozdzielone jedynie graficznie dla przejrzystości, które funkcjonalnie działają jak pojedyncze miejsce. Fuzja *Result* przeznaczona jest do gromadzenia informacji o operacjach. Wykonanie każdej tranzycji przekazuje do fuzji *Result* znacznik w postaci pary, której pierwszy element jest krótkim napisem zestawiającym symbol zadania numer operacji i symbol maszyny, natomiast drugi element – jako wartość zwrócona przez funkcję *now*() – reprezentuje czas rozpoczęcia operacji.

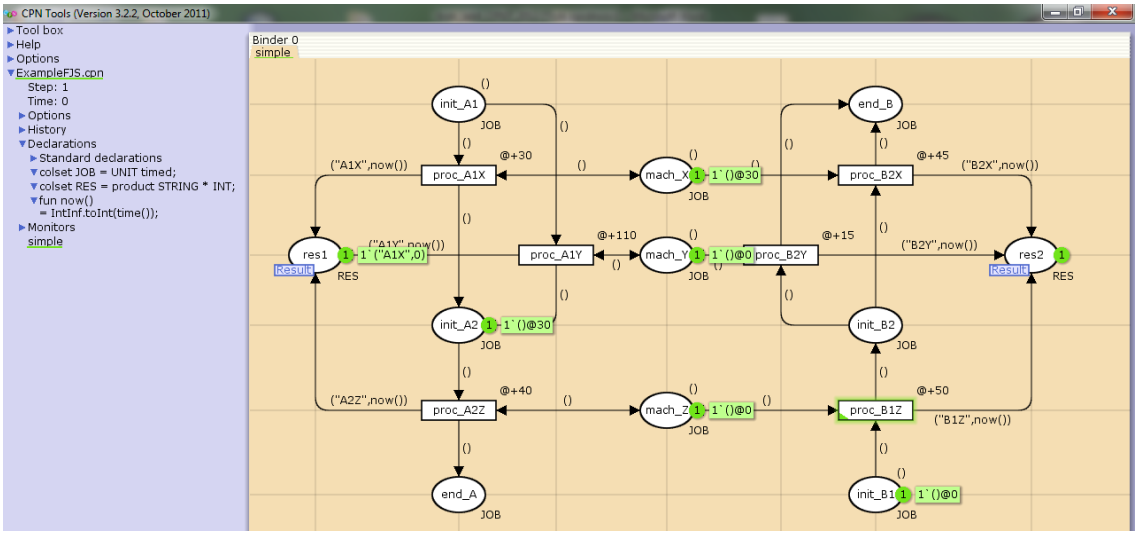
Z uwagi na mały rozmiar modelu, jego pełna symulacja składa się tylko z czterech kroków. Znakowanie początkowe widoczne jest na rysunku 2.1. Miejsca *init_A1* oraz *init_B1* są znakowane pojedynczymi znacznikami (), które reprezentują zadania oczekujące na pierwsze operacje, pieczątki czasowe tych znaczników mają wartość 0, czyli znaczniki mogą być pobrane z tych miejsc w chwili początkowej (nie ma ograniczeń czasowych na pierwsze operacje obu zadań). Także w miejscach *mach_X*, *mach_Y* i *mach_Z* znajdują się znaczniki () z pieczątkami czasowymi 0, co oznacza, że wszystkie maszyny są wolne w chwili początkowej. Każda z operacji może się rozpocząć, gdy wykonana została operacja poprzedzająca (ograniczenie kolejnościowe) oraz gdy jest dostępna przynajmniej jedna z odpowiednich maszyn (ograniczenie zasobowe). W modelu sieciowym odpowiada to przygotowaniu właściwej tranzycji **proc**, uwarunkowanemu obecnością znaczników z odpowiednimi pieczątkami czasowymi w jej miejscach wejściowych.

W stanie znakowania początkowego przygotowane są tranzycje *proc_A1X*, *proc_A1Y* oraz *proc_B1Z*. Każdą z nich można wykonać wywołując opcję *Fire Transition*. Niech wybrana zostanie tranzycja *proc_A1X*. Po jej wykonaniu model przechodzi do stanu pokazanego na rysunku 2.2. Znaczniki () zostały pobrane z miejsc wejściowych wykonanej tranzycji (*init_A1* oraz *mach_X*) i wprowadzone do jej miejsc wyjściowych:

- do miejsca *init_A2* został wprowadzony znacznik () z pieczątką czasową równą 30, co wynika z wartości opóźnienia tranzycji, zdefiniowanej wyrażeniem @+30,

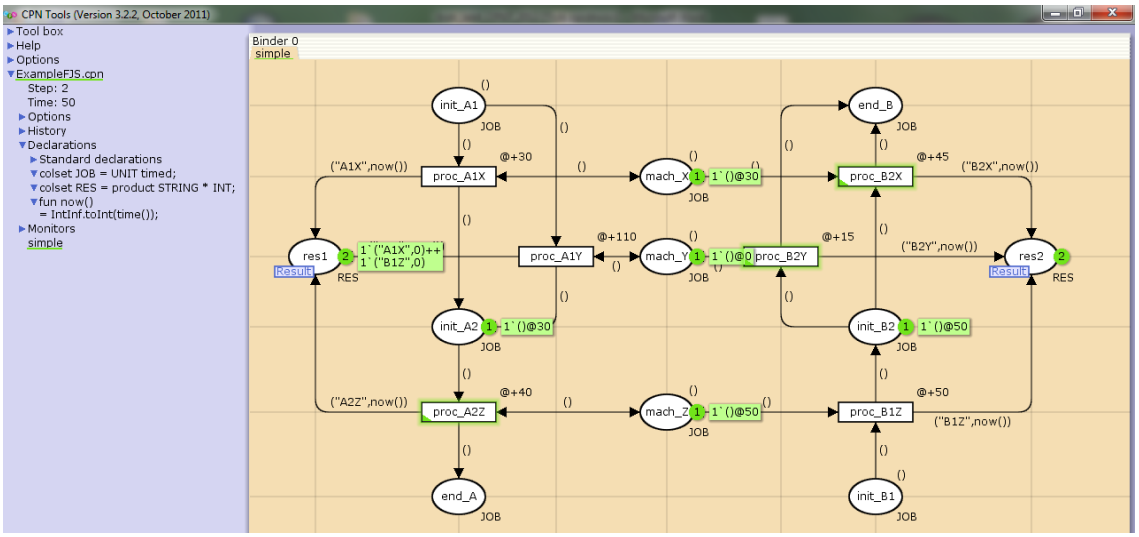
- do miejsca *mach_X* (jest to jednocześnie miejsce wejściowe i wyjściowe tranzycji *proc_A1X*) został wprowadzony taki sam znacznik ()@30,
- do fuzji miejsc *Result* został wprowadzony znacznik (*"A1X"*, 0), którego celem jest zachowanie informacji o zdarzeniu, że operacja 1 zadania *A* rozpoczęła się na maszynie *M_X* w chwili 0.

Dzięki opóźnieniu pieczątek czasowych, dodanemu w ramach wykonania tranzycji *proc_A1X*, znaczniki z miejsc *init_A2* oraz *mach_X* nie będą mogły zostać pobrane, dopóki zegar globalny nie osiągnie wartości równej lub większej 30. W ten sposób modelowana jest zajętość przetwarzanego materiału oraz maszyny w czasie trwania operacji.



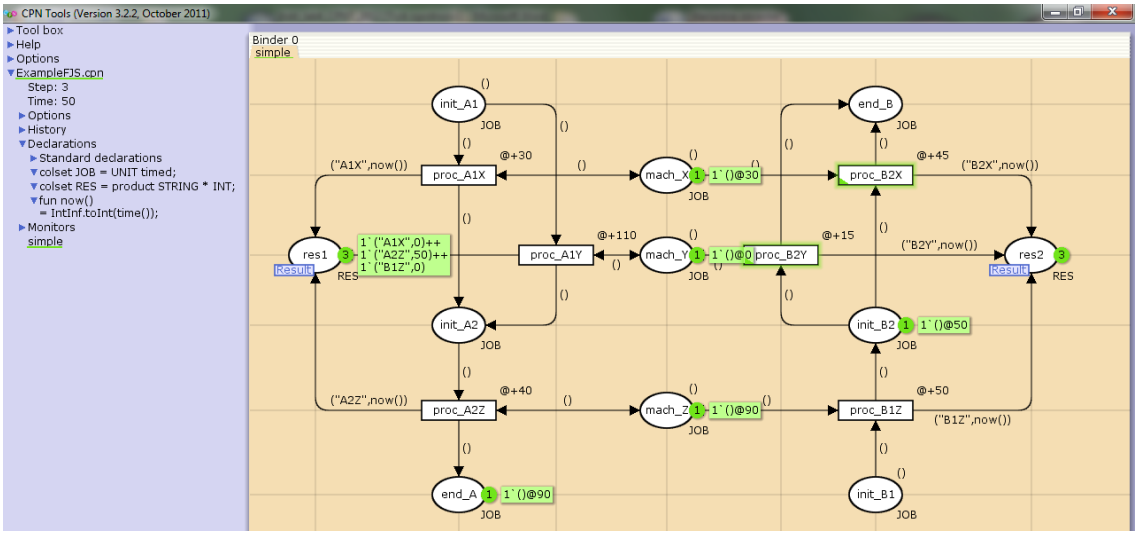
Rys. 2.2. Model sieciowy – stan po wykonaniu tranzycji *proc_A1X* (po kroku 1)

Wykonanie tranzycji *proc_A1X* spowodowało zablokowanie przygotowania tranzycji *proc_A1Y* (tranzycje te były w konflikcie). Jedyną tranzycją przygotowaną do wykonania w stanie pokazanym na rysunku 2.2 pozostaje *proc_B1Z*. Wykonanie tej tranzycji przeprowadza sieć do stanu przedstawionego na rysunku 2.3. Warto zauważyć, że po kroku 2 nastąpiło zwiększenie wartości zegara globalnego (do 50). Stało się tak, ponieważ 50 jest minimalną wartością zegara globalnego, zapewniającą gotowość co najmniej jednej tranzycji w strukturze sieciowej. W nowym stanie przygotowane są trzy tranzycje (*proc_A2Z*, *proc_B2X*, *proc_B2Y*), przy czym dwie ostatnie z wymienionych pozostają w konflikcie.



Rys. 2.3. Model sieciowy – stan po wykonaniu tranzycji *proc_B1Z* (po kroku 2)

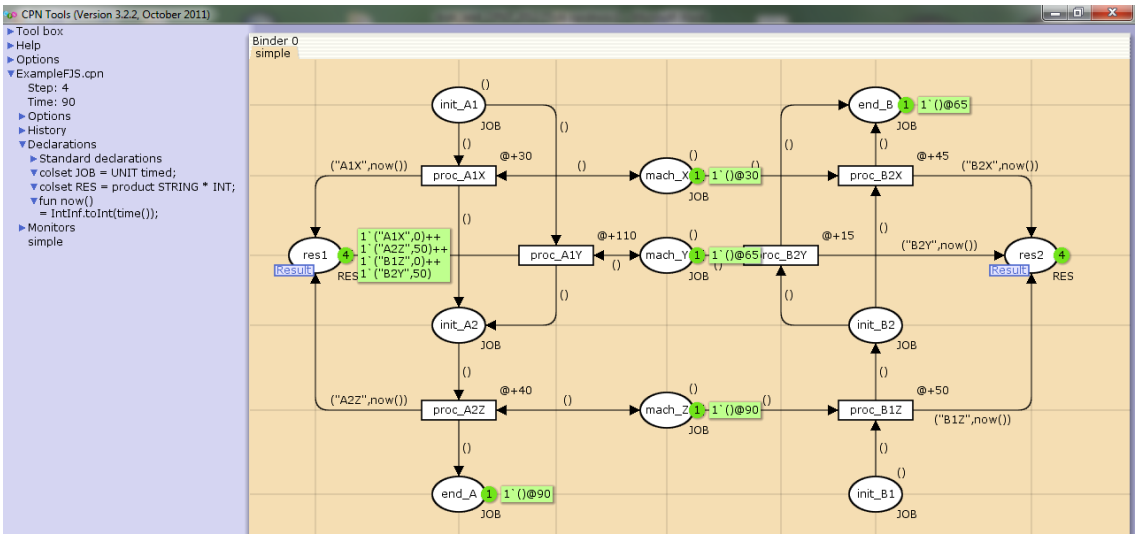
Niech jako kolejna do wykonania wybrana zostanie tranzycja *proc_A2Z*. Po jej wykonaniu sieć osiąga stan przedstawiony na rysunku 2.4. Zegar globalny nadal zachowuje wartość 50, ponieważ nadal istnieją tranzycje przygotowane dla takiej wartości zegara globalnego.



Rys. 2.4. Model sieciowy – stan po wykonaniu tranzycji *proc_A2Z* (po kroku 3)

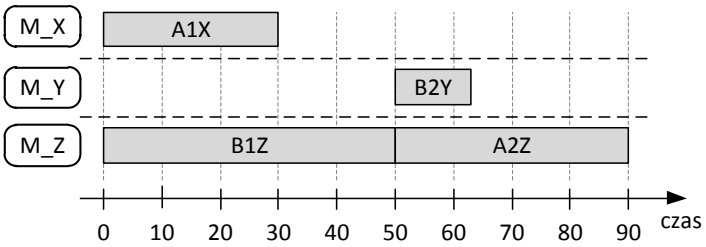
Z pozostałych dwóch przygotowanych tranzycji, będących w konflikcie (*proc_B2X*, *proc_B2Y*), niech zostanie wybrana do wykonania tranzycja *proc_B2Y*. Powoduje to przejście sieci do znakowania widocznego na rysunku 2.5, które jest znakowaniem martwym (końcowym), gdyż żadna tranzycja nie

jest już gotowa do wykonania. Zbiór znaczników zgromadzonych w fuzji miejsc *Result* reprezentuje informacje o przydziale maszyn oraz o czasach rozpoczęcia wszystkich czterech operacji realizowanych przez system produkcyjny.



Rys. 2.5. Model sieciowy – stan po wykonaniu tranzycji *proc_B2Y* (po kroku 4), znakowanie końcowe

W oparciu o znakowanie miejsc fuzji *Result* oraz dane dotyczące długości trwania operacji, zgromadzone w tabeli 2.2, można sporządzić reprezentację harmonogramu produkcji w postaci wykresu Gantta (rys. 2.6).



Rys. 2.6. Przykładowy harmonogram produkcji dla systemu FJS

Operacje *A1X* oraz *B1Z* rozpoczynają się niezwłocznie, ponieważ nie występują dla nich żadne ograniczenia. W przypadku operacji 2 zadania *J_B* wykonywanej na maszynie *M_Y* (*B2Y*) aktywne jest ograniczenie kolejnościowe – operacja zaczyna się natychmiast po zakończeniu operacji poprzedzającej z tego samego zadania. W przypadku operacji 2 zadania *J_A* wykonywanej na maszynie *M_Z* (*A2Z*) aktywne jest ograniczenie zasobowe – operacja zaczyna się natychmiast po zwolnieniu maszyny *M_Z* przez operację *B1Z*.

2.2. Przetwarzanie wsadowe

Dokonując niewielkiej modyfikacji wcześniej rozważanej struktury sieciowej, można przystosować ją do reprezentacji produkcji w trybie tzw. przetwarzania wsadowego. W przetwarzaniu wsadowym każda operacja stanowi obróbkę pewnej części całkowitego zadania produkcyjnego, której rozmiar wyznaczony jest przez pojemność procesu. Ten charakter przetwarzania może dotyczyć takich procesów jak mycie, suszenie, nagrzewanie itp.

Przykład 2.2

Dany jest problem produkcyjny typu FJS z przetwarzaniem wsadowym. Parametry problemu pozostają takie same, jak zdefiniowane dla przykładu 2.1 (tabele 2.1, 2.2), przy czym tabela 2.3 określa dodatkowo rozmiary zadań i pojemności procesowe przetwarzania wsadowego.

	Rozmiar	**M_X**	**M_Y**	**M_Z**
J_A	50	10	20	25
J_B	60	30	10	20

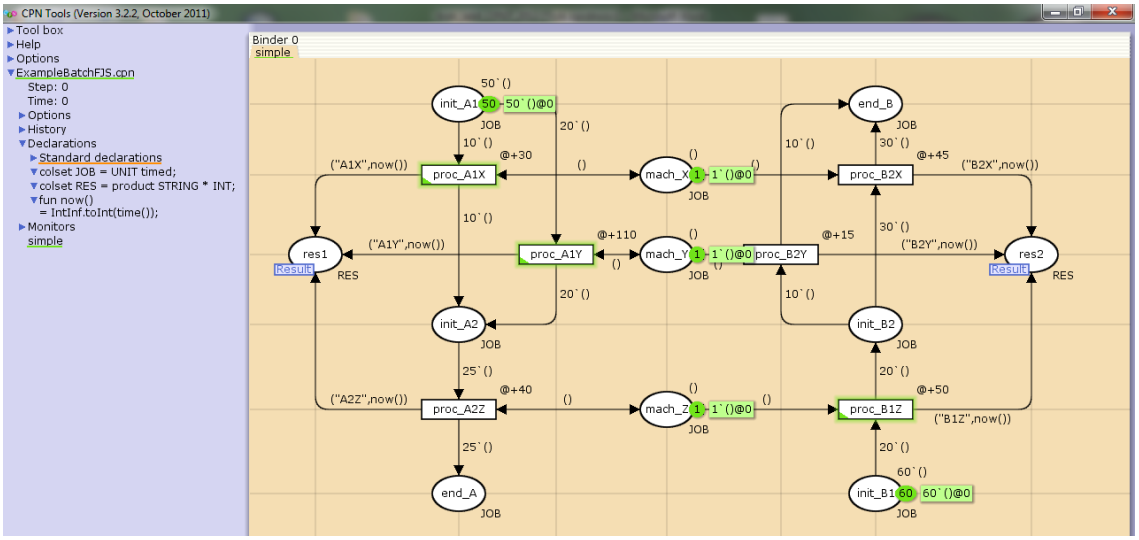
Tab. 2.3. Rozmiary zadań i pojemności procesowe przetwarzania wsadowego

Opracować model symulacyjny systemu z wykorzystaniem oprogramowania CPN Tools. Przeprowadzić symulację modelu, która pozwoli zbudować jeden (dowolny) harmonogram produkcyjny dla rozważanego systemu. Przedstawić harmonogram na wykresie Gantta.

Rozwiązanie

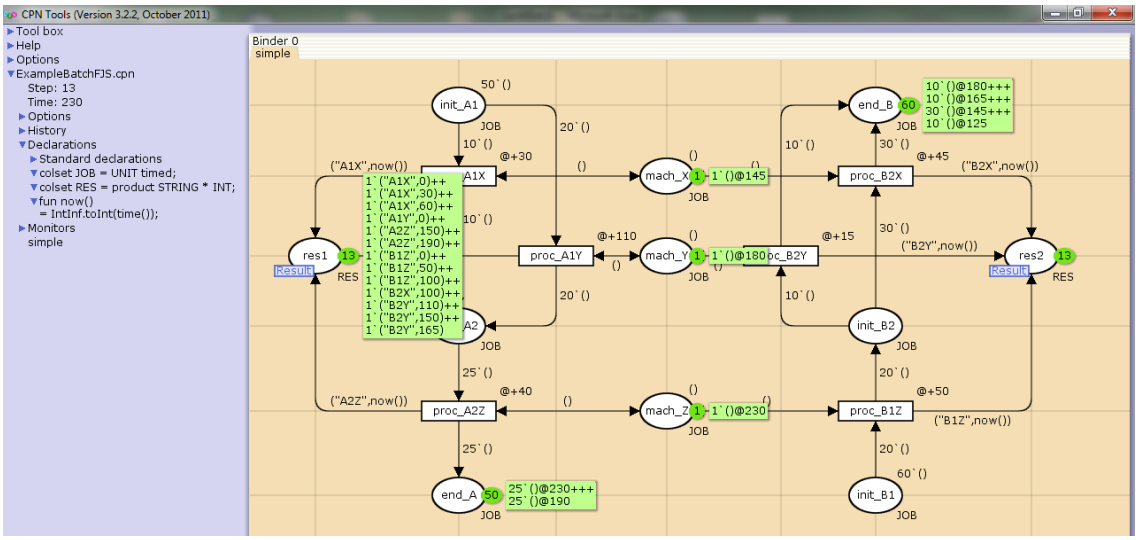
Za względu na zgodność struktury i większości parametrów systemów FJS z przykładów 2.1 i 2.2, poprzednio przeanalizowany model sieciowy wymaga jedynie niewielkiej adaptacji. Polega ona na zwiększeniu liczby znaczników znakowania początkowego miejsc *init_A1* i *init_B1* do wartości odpowiadających rozmiarom zadań oraz na zwiększeniu liczby znaczników przekazywanych przez łuki pomiędzy miejscami *init*/*end* do wartości odpowiadających pojemnościom procesów (rys. 2.7). Każde wykonanie tranzycji nadal odpowiada rozpoczęciu pojedynczej operacji, ale w

zmodyfikowanej strukturze operacja nie przetwarza jednorazowo całości zadania, lecz ściśle określoną jego część.



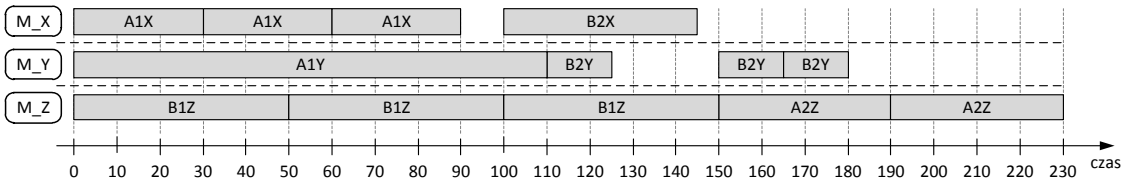
Rys. 2.7. Model sieciowy (FJS z przetwarzaniem wsadowym) – znakowanie początkowe

Przykładowy wynik symulacji (znakowanie końcowe) został przedstawiony na rysunku 2.8. Przetworzenie wszystkich zadań wymagało 13 operacji przetwarzania wsadowego.



Rys. 2.8. Model sieciowy (FJS z przetwarzaniem wsadowym) – przykładowe znakowanie końcowe

Znakowanie miejsc fuzji *Result*, tak jak w poprzednim przykładzie, zawiera informacje wystarczające (w połączeniu z danymi z tabeli 2.2) do przedstawienia harmonogramu produkcji w postaci wykresu Gantta (rys. 2.9).



Rys. 2.9. Przykładowy harmonogram produkcji dla systemu FJS z przetwarzaniem wsadowym

3. Zadania do samodzielnej realizacji

1. Otworzyć plik `ExampleFJS.cpn` zawierający model omówiony w przykładzie 2.1. Przeanalizować strukturę modelu sieciowego. Przeprowadzić symulację w trybie krokowym, według opisu podanego w punkcie 2.1. Czy wyniki są zgodne z podanymi w opisie?

2. Zmodyfikować projektu z pliku `ExampleFJS.cpn` (należy pracować na kopii tego pliku) tak, aby przekształcić go w model przetwarzania wsadowego, omówiony w przykładzie 2.2. Dokonać symulacji modelu. Czy jest możliwe uzyskanie wyników symulacji identycznych z tymi przedstawionymi na wykresie z rysunku 2.9?

3. Analizując harmonogram z rysunku 2.9 (jako pomoc można wykorzystać symulację w CPN Tools) odpowiedzieć na pytanie, jakie są przyczyny zatrzymania maszyn *M_X* i *M_Y* w przedziałach czasu odpowiednio 90-100 i 125-150. Czy są to ograniczenia zasobowe lub kolejnościowe?

4. Przygotować własny opis systemu produkcyjnego typu FJS za pomocą tabel podobnych do tabel 2.1, 2.2, przy czym rozmiar problemu powinien być większy: co najmniej **3 zadania**, co najmniej **po 3 operacje** w każdym zadaniu, co najmniej **5 maszyn**, co najmniej **2 maszyny alternatywne** dla każdego zadania. Rozszerzyć projektu z pliku `ExampleFJS.cpn` (należy pracować na kopii tego pliku) tak, aby zamodelować opisany system. Dokonać symulacji modelu. Wyniki przedstawić w postaci wykresu Gantta. Zidentyfikować aktywne ograniczenia zasobowe i kolejnościowe.

   Uwaga: Każdy zespół laboratoryjny powinien przedstawić system odmienny co do struktury i parametrów.

5. Rozszerzyć model utworzony w punkcie poprzednim tak, aby reprezentował system z przetwarzaniem wsadowym. Dokonać symulacji modelu. Przykładowy rezultat symulacji przedstawić w postaci wykresu Gantta.

W sprawozdaniu należy udokumentować wszystkie etapy realizacji ćwiczenia, przedstawić wyniki, sformułować wnioski.