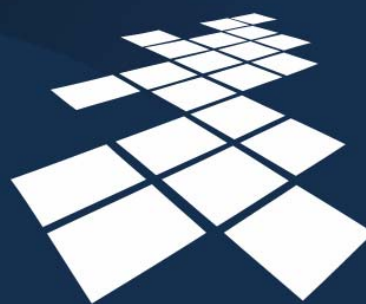


Przetwarzanie rozproszone

Wykład prowadzą:

Jerzy Brzeziński
Jacek Kobusiński



UCZELNIA
ONLINE

Plan wykładu

Przetwarzanie rozproszone

Plan wykładu

- Proces sekwencyjny
- Komunikaty, kanały komunikacyjne
- Stan kanału
- Operacje komunikacyjne
- Model formalny procesu sekwencyjnego
- Zdarzenia
- Warunek uaktywnienia
- Modele żądań

Przetwarzanie rozproszone (2)

Celem wykładu jest zapoznanie słuchacza z podstawowymi pojęciami i problemami związanymi z przetwarzaniem rozproszonym. Wykład obejmie omówienie m. in. definicji procesu sekwencyjnego, komunikatu, kanału komunikacyjnego czy też stanu kanału. W dalszej części wykładu omówione zostaną różnego rodzaju operacje komunikacyjne, a także scharakteryzowane zostaną rodzaje komunikacji. Student zapozna się także z formalnym modelem procesu sekwencyjnego. W następnej części wykładu zdefiniowane zostanie pojęcie zdarzenia i podane zostaną definicje różnych ich rodzajów. Kolejnym tematem który zostanie poruszony będzie pojęcie warunku uaktywnienia i definicje z tym związane. Na zakończenie przedstawione zostaną klasyczne modele żądań. Omówione zostaną ich definicje formalne i własności, a także zaprezentowana zostanie w sposób graficzny ich główna idea.

Przetwarzanie rozproszone



Przetwarzanie rozproszone

Procesem rozproszonym (przetwarzaniem rozproszonym) nazywamy **współbieżne i skoordynowane** (ang. *concurrent and coordinated*) wykonanie w środowisku rozproszonym zbioru $\mathcal{P}$ procesów sekwencyjnych $P_1, P_2, P_3, \dots, P_n$ współdziałających w realizacji wspólnego celu przetwarzania.

Przetwarzanie rozproszone (3)

Procesem rozproszonym (przetwarzaniem rozproszonym) nazywamy **współbieżne i skoordynowane** (ang. *concurrent and coordinated*) wykonanie w środowisku rozproszonym zbioru $\mathcal{P}$ procesów sekwencyjnych $P_1, P_2, P_3, \dots, P_n$ współdziałających w realizacji wspólnego celu przetwarzania.

Proces sekwencyjny

Przetwarzanie rozproszone



Proces sekwencyjny

Nieformalnie, każdy proces sekwencyjny jest działaniem wynikającym z wykonywania w pewnym środowisku (kontekście) programu sekwencyjnego (algorytmu sekwencyjnego), który składa się z ciągu operacji (instrukcji, wyrażeń) atomowych (nieprzerywalnych).

Przetwarzanie rozproszone (4)

Nieformalnie, każdy **proces sekwencyjny (zadanie)** jest działaniem wynikającym z wykonywania w pewnym środowisku (kontekście) **programu sekwencyjnego (algorytmu sekwencyjnego)**, który składa się z ciągu **operacji (instrukcji, wyrażeń) atomowych** (nieprzerywalnych).

Klasy operacji

Przetwarzanie rozproszone



Klasy operacji

Wyróżnia się dwie podstawowe klasy operacji:

- **wewnętrzne** (ang. *internal*)
odnoszą się tylko do zmiennych lokalnych programu
- **komunikacyjne** (ang. *communication*)
odnoszą się do środowiska i dotyczą komunikatów (ang. *messages*) oraz kanałów (ang. *channels*)

Przetwarzanie rozproszone (5)

Wyróżnia się dwie podstawowe klasy operacji:

- wewnętrzne (ang. *internal, local*),
- komunikacyjne (ang. *communication*).

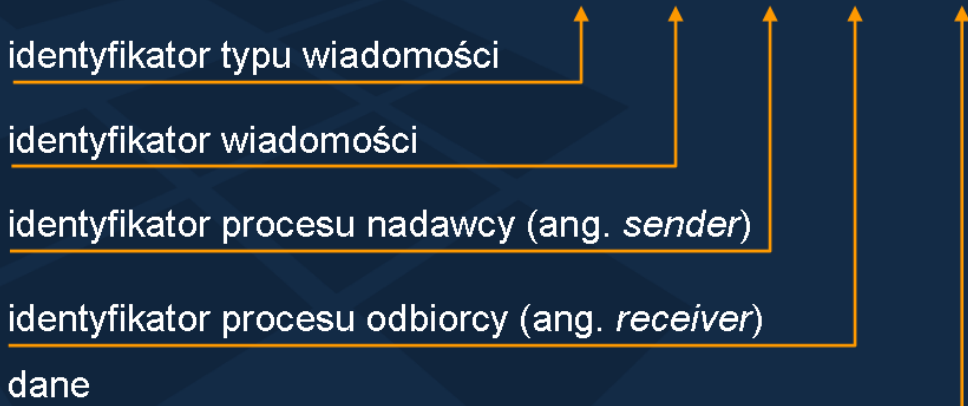
Operacje wewnętrzne odnoszą się tylko do zmiennych lokalnych programu. **Operacje komunikacyjne** odnoszą się natomiast do środowiska i dotyczą komunikatów (ang. *messages*) oraz kanałów (ang. *channels*).

Przetwarzanie rozproszone



Komunikat - definicja

Komunikat (wiadomość) jest dynamiczną strukturą danych:

$$M = \langle tag, mId, sId, rId, data \rangle$$


identyfikator typu wiadomości

identyfikator wiadomości

identyfikator procesu nadawcy (ang. *sender*)

identyfikator procesu odbiorcy (ang. *receiver*)

dane

Przetwarzanie rozproszone (6)

Formalnie, **komunikat (wiadomość)** $M = \langle tag, mId, sId, rId, data \rangle$ jest dynamiczną strukturą danych, która obejmuje na ogół następujące atrybuty:

- identyfikator typu wiadomości - *tag*,
- identyfikator wiadomości - *mId*,
- identyfikator procesu *nadawcy* (ang. *sender*) - *sId*,
- identyfikator procesu *odbiorcy* (ang. *receiver*) - *rId*,
- pole danych - *data*.

Kanał – definicja

Przetwarzanie rozproszone



Kanał - definicja

Kanał jest obiektem (zmienną) skojarzonym z uporządkowaną parą procesów $\langle P_i, P_j \rangle$, modelującym jednokierunkowe łącze transmisyjne.

Typem tego obiektu jest zbiór wiadomości, którego rozmiar nazywany jest **pojemnością kanału**.

Przetwarzanie rozproszone (7)

Kanał jest obiektem (zmienną) skojarzonym z uporządkowaną parą procesów $\langle P_i, P_j \rangle$, modelującym jednokierunkowe łącze transmisyjne. Typem tego obiektu jest zbiór wiadomości, którego rozmiar nazywany jest **pojemnością kanału**.



Kanały incydentne, wejściowe i wyjściowe

Kanał skojarzony z parą procesów $\langle P_i, P_j \rangle$, oznaczamy przez C_{ij} oraz nazywamy kanałem incydentnym z procesem P_i i z procesem P_j . Ponadto, kanał C_{ij} nazywamy kanałem wyjściowym procesu P_i oraz kanałem wejściowym procesu P_j .



Kanał skojarzony z parą procesów $\langle P_i, P_j \rangle$, oznaczamy przez C_{ij} oraz nazywamy kanałem *incydentnym* z procesem P_i i z procesem P_j . Ponadto, kanał C_{ij} nazywamy **kanałem wyjściowym** procesu P_i oraz **kanałem wejściowym** procesu P_j .

Zbiory kanałów

Przetwarzanie rozproszone



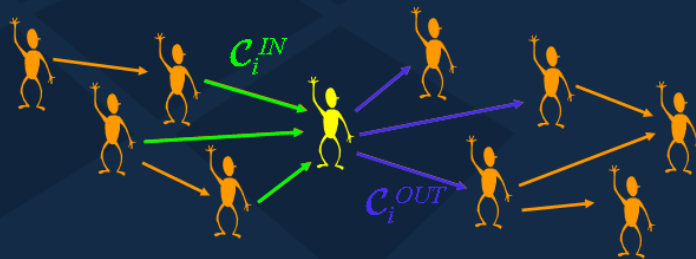
Zbiory kanałów

Zbiór kanałów wejściowych i wyjściowych tego procesu odpowiednio przez $\mathcal{C}_i^{IN}$ i $\mathcal{C}_i^{OUT}$.

Zbiór wszystkich kanałów incydentnych procesu P_i oznaczmy przez $\mathcal{C}_i$.

Tak więc:

$$\mathcal{C}_i = \mathcal{C}_i^{IN} \cup \mathcal{C}_i^{OUT} \quad (2.1)$$



Przetwarzanie rozproszone (9)

Zbiór wszystkich kanałów incydentnych procesu P_i oznaczmy przez $\mathcal{C}_i$, a zbiór kanałów wejściowych i wyjściowych tego procesu odpowiednio przez $\mathcal{C}_i^{IN}$ i $\mathcal{C}_i^{OUT}$. Tak więc, $\mathcal{C}_i = \mathcal{C}_i^{IN} \cup \mathcal{C}_i^{OUT}$.



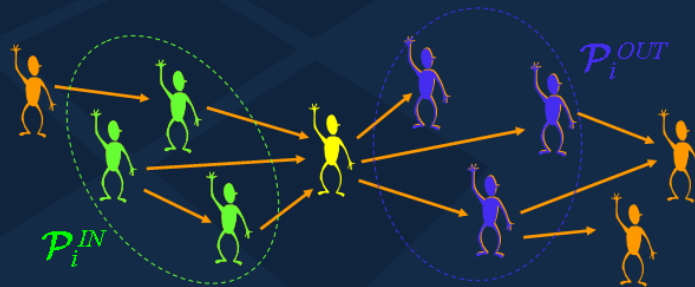
Zbiory procesów sąsiednich

Zbiór sąsiednich procesów wejściowych procesu P_i

$$\mathcal{P}_i^{IN} = \{P_j : \langle P_j, P_i \rangle \in \mathcal{C}_i^{IN}\} \quad (2.2)$$

Zbiór sąsiednich procesów wyjściowych procesu P_i

$$\mathcal{P}_i^{OUT} = \{P_j : \langle P_i, P_j \rangle \in \mathcal{C}_i^{OUT}\} \quad (2.3)$$



Przyjmujemy ponadto, że:

$$\mathcal{P}_i^{IN} = \{P_j : \langle P_j, P_i \rangle \in \mathcal{C}_i^{IN}\}$$

$$\mathcal{P}_i^{OUT} = \{P_j : \langle P_i, P_j \rangle \in \mathcal{C}_i^{OUT}\}$$

Zbiory $\mathcal{P}_i^{IN}$ i $\mathcal{P}_i^{OUT}$ są zbiorami **procesów sąsiednich** (incydentnych), odpowiednio **wejściowych** i **wyjściowych**, procesu P_i .

Stan kanału

Przetwarzanie rozproszone



Stan kanału

Przez **stan** L_{ij} **kanału** C_{ij} rozumiemy zbiór, lub uporządkowany zbiór, wiadomości wysłanych przez proces P_i lecz jeszcze nie odebranych przez proces P_j .

Przetwarzanie rozproszone (11)

Przez **stan** L_{ij} **kanału** C_{ij} rozumiemy zbiór, lub uporządkowany zbiór, wiadomości wysłanych przez P_i lecz jeszcze nie odebranych przez proces P_j .



Modelowanie opóźnień w kanale

W celu modelowania w kanale opóźnień komunikacyjnych, w zbiorze wiadomości L_{ij} wyróżnia się dwa rozłączne podzbiory:

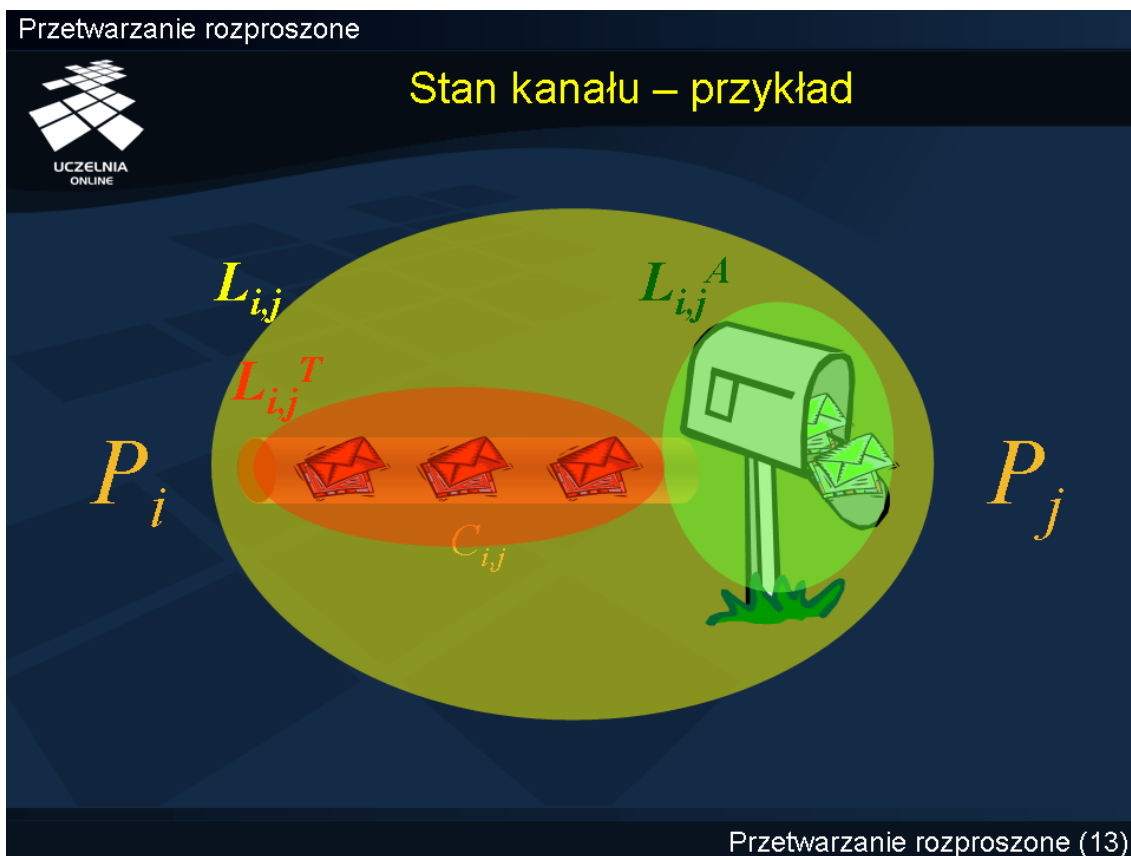
- zbiór wiadomości transmitowanych L_{ij}^T
(ang. *in-transit*)
- zbiór wiadomości dostępnych L_{ij}^A
(ang. *available, arrived, ready*)

Oczywiście, w każdej chwili

$$L_{ij} = L_{ij}^T \cup L_{ij}^A \quad (2.4)$$

W celu modelowania w kanale opóźnień komunikacyjnych istotnych w perspektywie dalszych rozważań, w zbiorze wiadomości L_{ij} wyróżnia się dwa rozłączne podzbiory: zbiór wiadomości transmitowanych (ang. *in-transit*) oraz zbiór wiadomości bezpośrednio dostępnych (ang. *available, arrived, ready*). **Zbiór wiadomości transmitowanych**, oznaczony przez L_{ij}^T , zawiera wiadomości, które zostały wysłane przez proces P_i lecz nie dotarły jeszcze do węzła realizującego proces P_j i tym samym nie są dostępne (widoczne) dla P_j . **Zbiór wiadomości dostępnych**, oznaczony przez L_{ij}^A , zawiera natomiast wiadomości wysłane przez P_i , które dotarły już (ang. *arrived*) do węzła procesu P_j i czekają w buforze kanału na ewentualne pobranie przez proces P_j . Oczywiście, w każdej chwili $L_{ij} = L_{ij}^T \cup L_{ij}^A$.

Stan kanału – przykład



W celu zobrazowania przedstawionych pojęć wyobraźmy sobie dwa procesy, P_i oraz P_j , połączone kanałem komunikacyjnym $L_{i,j}$. W pewnej chwili proces P_i rozpoczyna wysyłanie wiadomości do procesu P_j . Po pewnym czasie część z wiadomości wysłanych przez P_i staje się dostępna dla procesu P_j . Zwróćmy uwagę, że w kanale komunikacyjnym nadal znajdować się mogą wiadomości które zostały wysłane przez proces P_i , ale nie dotarły jeszcze do procesu P_i . Tworzą one zbiór wiadomości transmitowanych $L_{i,j}^T$. Wiadomości, które dotarły już do procesu P_j i są dla niego dostępne, tworzą zbiór wiadomości dostępnych $L_{i,j}^A$. Suma obu tych zbiorów tworzy zbiór wiadomości $L_{i,j}$, wysłanych przez P_i do procesu P_j , które nie zostały jeszcze przez niego odebrane. Powyższy slajd ilustruje przedstawione pojęcia i relacje między nimi.

Przetwarzanie rozproszone



Predykaty opisujące stan kanału

- $empty(C_{ij})$
- $in-transit(C_{ij})$
- $available(C_{ij})$

Przetwarzanie rozproszone (14)

W tym kontekście, z kanałem można skojarzyć predykaty opisujące poszczególne jego stany:

- $empty(C_{ij}) \equiv L_{ij} = \emptyset$
- $in-transit(C_{ij}) \equiv L_{ij}^T \neq \emptyset$
- $available(C_{ij}) \equiv L_{ij}^A \neq \emptyset$

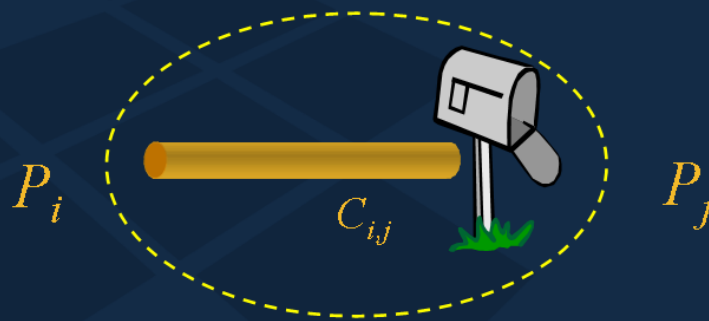
Predykat *empty*

Przetwarzanie rozproszone



Predykat *empty*

$$\text{empty}(C_{ij}) \equiv L_{ij} = \emptyset \quad (2.5)$$



Przetwarzanie rozproszone (15)

Predykat *empty*(C_{ij}) zachodzi wtedy i tylko wtedy, gdy $L_{ij} = \emptyset$ (kanał komunikacyjny jest pusty).

Predykat *in-transit*

Przetwarzanie rozproszone



Predykat *in-transit*

$$\textit{in-transit}(C_{ij}) \equiv L_{ij}^T \neq \emptyset \quad (2.6)$$



Przetwarzanie rozproszone (16)

Predykat *in-transit* (C_{ij}) zachodzi wtedy i tylko wtedy, gdy $L_{ij}^T \neq \emptyset$ (zbiór wiadomości transmitowanych nie jest pusty). Innymi słowy, w kanale komunikacyjnym znajdują się wiadomości wysłane przez proces P_i , które nie dotarły jeszcze do procesu P_j .

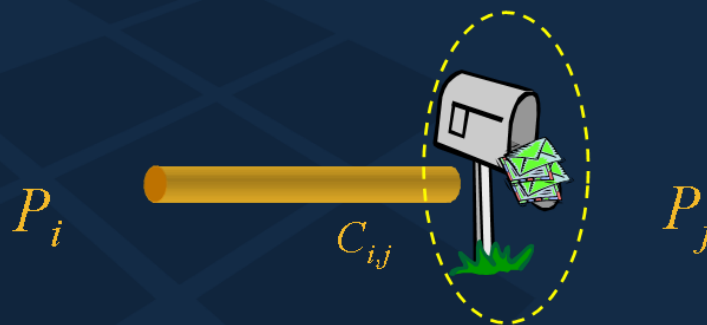
Predykat *available*

Przetwarzanie rozproszone



Predykat *available*

$$available(C_{ij}) \equiv L_{ij}^A \neq \emptyset \quad (2.7)$$



Przetwarzanie rozproszone (17)

Predykat *available* (C_{ij}) zachodzi wtedy i tylko wtedy, gdy $L_{ij}^A \neq \emptyset$. Innymi słowy, istnieją wiadomości, które zostały wysłane przez proces P_i i dotarły już do procesu P_j .

Indywidualne operacje komunikacyjne (1)

Przetwarzanie rozproszone



Indywidualne operacje komunikacyjne (1)

$send(P_i, P_j, M)$

Efektom wykonania tej operacji jest umieszczenie wiadomości M w kanale $C_{i,j}$, a więc wykonanie podstawienia

$$L_{i,j} := L_{i,j} \cup \{M\} \quad (2.8)$$

Przetwarzanie rozproszone (18)

Zbiór podstawowych operacji komunikacyjnych obejmuje indywidualne i grupowe operacje **nadawania** (ang. *send*) i **odbioru** (ang. *receive*).

$send(P_i, P_j, M)$ – w procesie nadawcy P_i , operacja ta jest sparametryzowana dodatkowo przez pojedynczą wiadomość M i identyfikator procesu odbiorcy P_j (lub odpowiednio kanału $C_{i,j}$). Efektom wykonania tej operacji jest umieszczenie wiadomości M w kanale $C_{i,j}$, a więc wykonanie podstawienia $L_{i,j} := L_{i,j} \cup \{M\}$.



Indywidualne operacje komunikacyjne (2)

receive(P_i, P_j, inM)

Jeżeli kanał C_{ij} nie jest pusty i pewna wiadomość M jest bezpośrednio dostępna ($available(C_{ij})$ ma wartość *True*), to efektem wykonania tej operacji jest pobranie wiadomości M z kanału C_{ij} , a więc wykonanie podstawienia:

$$L_{ij} := L_{ij} \setminus \{M\} \text{ oraz } inM := M \quad (2.9)$$

receive(P_i, P_j, inM) – w procesie odbiorcy P_j , operacja ta jest sparametryzowana przez pojedynczą zmienną inM i identyfikator procesu P_i – **oczekiwanego nadawcy** wiadomości. Jeżeli kanał C_{ij} nie jest pusty i pewna wiadomość M jest bezpośrednio dostępna ($available(C_{ij})$ ma wartość *True*), to efektem wykonania tej operacji jest pobranie wiadomości M z kanału C_{ij} , a więc wykonanie podstawienia $L_{ij} := L_{ij} \setminus \{M\}$ oraz $inM := M$.

Przetwarzanie rozproszone



Grupowe operacje komunikacyjne (1)

$send(P_i, \mathcal{P}_i^R, M)$

Efektom wykonania tej operacji jest umieszczenie wiadomości we wszystkich kanałach $C_{i,j}$, takich że $P_j \in \mathcal{P}_i^R$, a więc podstawienie dla wszystkich tych kanałów:

$$L_{i,j} := L_{i,j} \cup \{M\} \quad (2.10)$$

Przetwarzanie rozproszone (20)

$send(P_i, \mathcal{P}_i^R, M)$ – w procesie nadawcy P_i , operacja ta jest sparametryzowana przez pojedynczą wiadomość M i zbiór procesów $\mathcal{P}_i^R \subseteq \mathcal{P}_i^{OUT}$, będących odbiorcami (adresatami) wiadomości M . Efektom wykonania tej operacji jest umieszczenie wiadomości we wszystkich kanałach $C_{i,j}$, takich że $P_j \in \mathcal{P}_i^R$, a więc podstawienie dla wszystkich tych kanałów: $L_{i,j} := L_{i,j} \cup \{M\}$.



Grupowe operacje komunikacyjne (2)

$$receive(\mathcal{P}_j^S, P_j, sInM)$$

Efektom wykonania operacji $receive(\mathcal{P}_j^S, P_j, sInM)$ jest atomowe pobranie wiadomości M_i od procesów $P_i \in \mathcal{P}_j^S$ i umieszczanie ich w $sInM$. Tym samym, dla każdego procesu $P_i \in \mathcal{P}_j^S$, wykonywane jest kolejno podstawienie

$$L_{i,j} := L_{i,j} \setminus \{M_i\} \text{ oraz } sInM := sInM \cup \{M_i\} \quad (2.11)$$

$receive(\mathcal{P}_j^S, P_j, sInM)$ – w procesie odbiorcy P_j operacja ta jest sparametryzowana przez zmienną $sInM$ typu zbiór wiadomości i zbiór procesów $\mathcal{P}_i^S \subseteq \mathcal{P}_j^{IN}$, będących **oczekiwanymi nadawcami** wiadomości. W ogólności, warunkiem wykonania tej operacji jest jednoczesna dostępność wiadomości od wszystkich procesów $P_i \in \mathcal{P}_j^S$. Jeżeli warunek ten jest spełniony, to efektem wykonania operacji $receive(\mathcal{P}_j^S, P_j, sInM)$ jest atomowe pobranie wiadomości M_i od procesów $P_i \in \mathcal{P}_j^S$ i umieszczanie ich w $sInM$. Tym samym, dla każdego procesu $P_i \in \mathcal{P}_j^S$, wykonywane jest kolejno podstawienie $L_{i,j} := L_{i,j} \setminus \{M_i\}$ oraz $sInM := sInM \cup \{M_i\}$.

Przetwarzanie rozproszone



Rodzaje komunikacji

Kanały o niezerowej pojemności umożliwiają realizację operacji następujących typów komunikacji:

- nieblokowanej
- blokowanej

Przetwarzanie rozproszone (22)

Kanały o niezerowej pojemności umożliwiają realizację nieblokowanych (asynchronicznych) i blokowanych (synchronicznych) operacji komunikacji. W **operacjach komunikacji nieblokowanej**, proces nadający przekazuje komunikat do kanału i natychmiast kontynuuje swoje działanie, a proces odbierający - odczytuje stan kanału wejściowego, lecz nawet gdy kanał jest pusty, proces kontynuuje działanie. W wypadku operacji **komunikacji blokowanej**, nadawca jest wstrzymywany do momentu, gdy wiadomość zostanie odebrana przez adresata, natomiast odbiorca - do momentu, gdy oczekiwana wiadomość pojawi się w jego buforze wejściowym. W tym kontekście, wyróżnia się komunikację synchroniczną i asynchroniczną.

Komunikacja synchroniczna

Przetwarzanie rozproszone



Komunikacja synchroniczna

W komunikacji synchronicznej, nadawca i odbiorca są blokowani aż odpowiedni odbiorca odczyta przesłaną do niego wiadomość (ang. *rendez-vous*).



Przetwarzanie rozproszone (23)

W **komunikacji synchronicznej**, nadawca jest blokowany, aż odpowiedni odbiorca odczyta przesłaną do niego wiadomość, a odbiorca jest blokowany, aż nadana wiadomość do niego dotrze (*rendez-vous*).



Komunikacja asynchroniczna

W przypadku komunikacji asynchronicznej, nadawca lub odbiorca komunikuje się w sposób nieblokowany.



W przypadku **komunikacji asynchronicznej**, nadawca lub odbiorca komunikuje się w sposób nieblokowany.

W dalszej części pracy typ operacji będzie wynikać z kontekstu lub domyślnie przyjmować będziemy, że stosowane są... operacje asynchronicznego nadania i synchronicznego odbioru. Jeśli okaże się to jednak niezbędne, to dla odróżnienia asynchroniczne operacje odbioru będziemy oznaczać przez **receive_a**(P_i, P_j, inM) lub **receive_a**($\mathcal{P}_j^S, P_j, sInM$), a synchroniczne operacje nadania - przez **send_s**(P_i, P_j, M) lub **send_s**($P_i, \mathcal{P}_i^R, M$).



Model formalny procesu sekwencyjnego

Formalnie, proces sekwencyjny P_i może być opisany (modelowany) przez uporządkowaną czwórkę:

$$P_i = \langle \mathcal{S}_i, \mathcal{S}_i^0, \mathcal{E}_i, \mathcal{F} \rangle \quad (2.12)$$

gdzie

- $\mathcal{S}_i$ jest zbiorem stanów $\mathcal{S}_i$ procesu P_i
- $\mathcal{S}_i^0$ jest zbiorem stanów początkowych, $\mathcal{S}_i^0 \subseteq \mathcal{S}_i$
- $\mathcal{E}_i$ jest zbiorem zdarzeń procesu P_i
- $\mathcal{F}$ jest funkcją tranzycji, taką że: $\mathcal{F}_i \subseteq \mathcal{S}_i \times \mathcal{E}_i \times \mathcal{S}_i$, a
 $\langle S, E, S' \rangle \in \mathcal{F}_i$, (2.13)

jeżeli zajście zdarzenia E w stanie S jest możliwe i prowadzi do zmiany stanu na S'

Formalnie, proces sekwencyjny może być opisany (modelowany) przez uporządkowaną czwórkę $P_i = \langle \mathcal{S}_i, \mathcal{S}_i^0, \mathcal{E}_i, \mathcal{F} \rangle$, gdzie

$\mathcal{S}_i$ – jest **zbiorem stanów** procesu P_i ,

$\mathcal{S}_i^0$ – jest **zbiorem stanów początkowych**, $\mathcal{S}_i^0 \subseteq \mathcal{S}_i$,

$\mathcal{E}_i$ – jest **zbiorem zdarzeń** procesu ,

$\mathcal{F}$ – jest **funkcją tranzycji**, taką że: $\mathcal{F}_i \subseteq \mathcal{S}_i \times \mathcal{E}_i \times \mathcal{S}_i$, a $\langle S, E, S' \rangle \in \mathcal{F}_i$, jeżeli zajście zdarzenia E w stanie S jest możliwe i prowadzi do zmiany stanu na S' .

Stan procesu (1)

Przetwarzanie rozproszone



Stan procesu (1)

Stan $S_i(t)$ procesu w chwili t czasu lokalnego jest w ogólności zbiorem wartości wszystkich zmiennych lokalnych skojarzonych z procesem w chwili t oraz ciągów wiadomości wysłanych (wpisanych) do incydentnych kanałów wyjściowych i ciągów wiadomości odebranych z incydentnych kanałów wejściowych do chwili t .

Przetwarzanie rozproszone (26)

Stan $S_i(t)$ procesu w chwili czasu lokalnego jest w ogólności zbiorem wartości wszystkich zmiennych lokalnych skojarzonych z procesem w chwili oraz ciągów wiadomości wysłanych (wpisanych) do incydentnych kanałów wyjściowych i ciągów wiadomości odebranych z incydentnych kanałów wejściowych do chwili t .

Stan procesu (2)

Przetwarzanie rozproszone



Stan procesu (2)

Dla każdego t , $S_i(t) \in \mathcal{S}_i$. W celu uproszczenia notacji, zależność stanu od czasu można przyjąć za domyślną i jeśli nie prowadzi to do niejednoznaczności, oznaczać stan w pewnej chwili t przez S_i .

Zbiór $\mathcal{S}_i^0$ jest zbiorem stanów początkowych, których wartości są zadawane wstępnie, bądź są wynikiem zajścia wyróżnionego zdarzenia inicjującego E_i^0 .

Przetwarzanie rozproszone (27)

Dla każdego t , $S_i(t) \in \mathcal{S}_i$. W celu uproszczenia notacji, zależność stanu od czasu można przyjąć za domyślną i jeśli nie prowadzi to do niejednoznaczności, oznaczać stan w pewnej chwili t przez S_i .

Zbiór $\mathcal{S}_i^0$ jest **zbiorem stanów początkowych**, których wartości są zadawane wstępnie, bądź są wynikiem zajścia wyróżnionego **zdarzenia inicjującego** E_i^0 .

Definicja zdarzenia

Przetwarzanie rozproszone



Definicja zdarzenia

Zdarzenie E_i^k odpowiada unikalnemu wykonaniu operacji atomowej, zmieniającemu stan S_i procesu i ewentualnie stan incydentnych z procesem kanałów $C_{i,j}$ lub $C_{j,i}$.

Jeżeli operacja odpowiadająca zdarzeniu została wykonana, to powiemy, że **zdarzenie zaszło**.

Przetwarzanie rozproszone (28)

Zdarzenie E_i^k odpowiada unikalnemu wykonaniu operacji atomowej, zmieniającemu stan S_i procesu i ewentualnie stan incydentnych z procesem kanałów $C_{i,j}$ lub $C_{j,i}$. Jeżeli operacja odpowiadająca zdarzeniu została wykonana, to powiemy, że **zdarzenie zaszło**.

Klasy zdarzeń

Przetwarzanie rozproszone



Klasy zdarzeń

- *e_send*
- *e_receive*
- *e_internal*

Przetwarzanie rozproszone (29)

Wyróżnia się trzy podstawowe klasy zdarzeń: *e_send*, *e_receive* i *e_internal*.

Zdarzenie e_send

Przetwarzanie rozproszone



Zdarzenie e_send

$e_send(P_i, P_j, M)$

zachodzi w procesie P_i , w wyniku wykonania przez ten proces operacji $send(P_i, P_j, M)$

$e_send(P_i, \mathcal{P}_i^R, M)$

zachodzi w procesie P_i , w wyniku wykonania przez ten proces operacji $send(P_i, \mathcal{P}_i^R, M)$

Przetwarzanie rozproszone (30)

W szczególności, zdarzenie $e_send(P_i, P_j, M)$ zachodzi w procesie P_i , w wyniku wykonania przez ten proces operacji $send(P_i, P_j, M)$. Analogicznie definiuje się zdarzenie $e_send(P_i, \mathcal{P}_i^R, M)$.

Zdarzenie $e_receive$

Przetwarzanie rozproszone



Zdarzenie $e_receive$

$e_receive(P_i, P_j, M)$

zachodzi w procesie P_j , gdy P_j wykonał operację $receive(P_i, P_j, M)$, a odczytana do zmiennej lokalnej inM wiadomość M pochodziła od procesu P_i

$e_receive(\mathcal{P}_j^S, P_j, \mathcal{M}_j^S)$

zachodzi w procesie P_j , gdy P_j wykonał operację $receive(\mathcal{P}_j^S, P_j, sInM)$, a odczytane do zmiennej lokalnej $sInM$ wiadomości $M_i \in \mathcal{M}_j^S$ pochodzą od procesów $P_i \in \mathcal{P}_j^S$

Przetwarzanie rozproszone (31)

Zdarzenie $e_receive(P_i, P_j, M)$ zachodzi natomiast w procesie P_j , gdy P_j wykonał operację $receive(P_i, P_j, sInM)$, a odczytana do zmiennej lokalnej inM wiadomość M pochodziła od procesu P_i . Analogicznie definiuje się zdarzenie $e_receive(\mathcal{P}_j^S, P_j, \mathcal{M}_j^S)$, odpowiadające wykonaniu operacji. Zbiór $\mathcal{M}_j^S$ jest tu zbiorem jednocześnie odbieranych wiadomości z kanałów wejściowych.

Zdarzenie $e_internal$

Przetwarzanie rozproszone



Zdarzenie $e_internal$

$e_internal(P_i, P_j, M)$

zachodzi gdy proces P_i wykonał operację, która nie zmienia stanu jego kanałów incydentnych.

Do zdarzeń lokalnych zalicza się między innymi zdarzenia:

$e_init(P_i, S_i^k)$ – które nadaje procesowi P_i stan S_i^k
(w szczególności stan początkowy)

$e_stop(P_i)$ – które kończy wykonywanie procesu

Przetwarzanie rozproszone (32)

Powiemy, że w procesie P_i zaszło zdarzenie **lokalne** $e_internal(P_i)$, gdy proces ten wykonał operację, która nie zmienia stanu kanałów incydentnych procesu. Do zdarzeń lokalnych zalicza się między innymi zdarzenia: $e_init(P_i, S_i^k)$ – które nadaje procesowi P_i stan S_i^k (w szczególności stan początkowy S_i^0), oraz zdarzenie $e_stop(P_i)$ – które kończy wykonywanie procesu P_i .



Dostępność wiadomości

Dostępność wiadomości utożsamiać można z zajściem zdarzeń w środowisku komunikacyjnym:

- *zdarzenie dostarczenia wiadomości* $e_deliver(P_i, P_j, M)$
- *zdarzenia nadejścia wiadomości* $e_arrive(P_i, P_j, M)$

Przez $\mathcal{P}_j^A$ oznaczać będziemy zbiór procesów, których wiadomości dotarły i są dostępne dla P_j .

Należy zaznaczyć, że zdarzenie $e_send(P_i, P_j, M)$ odpowiada na ogół wykonaniu operacji asynchronicznej, która zapisuje wiadomość M do kanału wyjściowego (bufora łącza transmisyjnego). Następnie środowisko rozproszone transmituje wiadomość M do węzła docelowego i zapisuje ją w buforze wejściowym procesu docelowego P_j . Gdy wiadomość dotrze do bufora wejściowego procesu docelowego, to wiadomość staje się dostępna i może być natychmiast pobrana z bufora w wyniku wykonania operacji $receive(P_i, P_j, inM)$. Dostępność wiadomości utożsamiać też można z zajściem zdarzeń w środowisku komunikacyjnym: **zdarzenia dostarczenia wiadomości** $e_deliver(P_i, P_j, M)$, albo **zdarzenia nadejścia wiadomości** – $e_arrive(P_i, P_j, M)$. W tym kontekście, przez $\mathcal{P}_j^A$ oznaczmy zbiór procesów, których wiadomości dotarły i są dostępne dla P_j . Jeżeli proces odbiorcy P_j odczytuje skierowaną do niego wiadomość M , wykonując operację $receive(P_i, P_j, inM)$, wiadomość ta jest przepisywana (przemieszczana) z bufora wyjściowego łącza (bufora wejściowego procesu P_i) do lokalnej zmiennej procesu inM . Jeżeli przepisanie to nastąpiło, to mówimy, że wiadomość **została odebrana** lub, że zaszło **zdarzenie odbioru** $e_receive(P_i, P_j, inM)$.

Warto przypomnieć, że wiadomości mają swoje unikalne identyfikatory, które pozwalają na jednoznaczne kojarzenie zdarzenia nadania konkretnej wiadomości ze zdarzeniem odbioru tej samej wiadomości.



Funkcja tranzycji

Funkcja tranzycji $\mathcal{F}_i \subseteq \mathcal{S}_i \times \mathcal{E}_i \times \mathcal{S}_i$ opisuje reguły zmiany stanu S na S' w wyniku zajścia zdarzenia E .

Elementy $\langle S, E, S' \rangle \in \mathcal{F}_i$ nazwiemy tranzycjami lub krokami. W zależności od zachodzącego zdarzenia E , tranzycję nazwiemy odpowiednio tranzycją wejścia, wyjścia lub lokalną.

Funkcja tranzycji $\mathcal{F}_i \subseteq \mathcal{S}_i \times \mathcal{E}_i \times \mathcal{S}_i$ opisuje reguły zmiany stanu S na S' w wyniku zajścia zdarzenia E . Elementy $\langle S, E, S' \rangle \in \mathcal{F}_i$ nazwiemy **tranzycjami** lub **krokami**. W zależności od zachodzącego zdarzenia E , tranzycję nazwiemy odpowiednio **tranzycją wejścia**, **wyjścia** lub **lokalną**.



Zdarzenia dopuszczalne

Funkcja tranzycji dopuszcza możliwość zajścia zdarzenia E tylko w tych stanach S , dla których $\langle S, E, S' \rangle \in \mathcal{F}_i$. Dlatego też, w wypadku gdy $\langle S, E, S' \rangle \in \mathcal{F}_i$, powiemy że zdarzenie jest **dopuszczalne** (ang. *allowed*) w stanie S .

Wprowadzimy też predykat $allowed(E)$ oznaczający, że w danej chwili zdarzenie E jest dopuszczalne.

Funkcja tranzycji dopuszcza możliwość zajścia zdarzenia E tylko w tych stanach S , dla których $\langle S, E, S' \rangle \in \mathcal{F}_i$. Dlatego też, w wypadku gdy $\langle S, E, S' \rangle \in \mathcal{F}_i$, powiemy że zdarzenie E jest **dopuszczalne** (ang. *allowed*) w stanie S . Wprowadzimy też predykat $allowed(E)$ oznaczający, że w danej chwili zdarzenie E jest dopuszczalne.



Zdarzenia gotowe

Oprócz czynnika wewnętrznego (stanu procesu), zajście zdarzenia może być dodatkowo uwarunkowane stanem kanałów wejściowych (środowiska). Jeśli zdarzenie może zajść ze względu na warunki zewnętrzne (stan kanałów), to powiemy że zdarzenie jest **przygotowane** lub **gotowe** (ang. *ready*).

Fakt gotowości zdarzenia E w danej chwili wyrażać będzie predykat $ready(E)$.

Oprócz czynnika wewnętrznego (stanu procesu), zajście zdarzenia może być dodatkowo uwarunkowane stanem kanałów wejściowych (środowiska). Na przykład, zdarzenie $e_receive(P_i, P_j, M)$ może zajść tylko wówczas, gdy wiadomość M została wcześniej wysłana przez proces P_i i jest aktualnie dostępna w kanale $C_{i,j}$. Podobnie, zdarzenie $e_receive(\mathcal{P}_j^S, P_j, \mathcal{M}_j^S)$ może zajść tylko wówczas, gdy jest spełniony warunek wykonania operacji $receive(\mathcal{P}_j^S, P_j, sInM)$. Jeżeli zdarzenie może zajść ze względu na warunki zewnętrzne (stan kanałów), to powiemy że zdarzenie jest **przygotowane** lub **gotowe** (ang. *ready*). Fakt gotowości zdarzenia w danej chwili wyrażać będzie predykat $ready(E)$.

Predykat *enable*

Przetwarzanie rozproszone



Predykat *enable*

Predykat $enable(E)$, oznacza, że zdarzenie jest aktywne, czyli jednocześnie gotowe i dopuszczalne. Stąd też:

$$enable(E) \equiv ready(E) \wedge allowed(E) \quad (2.14)$$

Przetwarzanie rozproszone (37)

W tym kontekście wprowadzimy jeszcze predykat $enable(E)$, oznaczający, że zdarzenie jest **aktywne** – czyli jednocześnie gotowe i dopuszczalne. Stąd też:

$$enable(E) \equiv ready(E) \wedge allowed(E)$$



Procesy zakończone, wstrzymane

Powiemy, że proces P_i jest w **stanie końcowym** S_i^e , jeżeli zbiór zdarzeń dopuszczalnych w tym stanie jest pusty.

Jeżeli natomiast niepusty zbiór zdarzeń dopuszczalnych zawiera wyłącznie zdarzenia odbioru i żadne z tych zdarzeń nie jest aktywne (gotowe), to powiemy że proces jest **wstrzymany (zablokowany)**.

Powiemy, że proces jest w **stanie końcowym** S_i^e , jeżeli zbiór zdarzeń dopuszczalnych w tym stanie jest pusty. Jeżeli natomiast niepusty zbiór zdarzeń dopuszczalnych zawiera wyłącznie zdarzenia odbioru i żadne z tych zdarzeń nie jest aktywne (gotowe), to powiemy że proces jest **wstrzymany (zablokowany)**.



Procesy aktywne, pasywne

Proces wstrzymany lub zakończony nazwiemy **pasywnym**. Przez proces **aktywny** będziemy natomiast rozumieć proces, który nie jest pasywny.

Przyjmujemy, że w każdej chwili t stan procesu P_i reprezentuje zmienna logiczna $passive_i$, przyjmująca wartość

- *True*, gdy proces P_i jest pasywny
- *False*, gdy proces P_i jest aktywny

Proces wstrzymany lub zakończony nazwiemy **pasywnym**. Przez proces **aktywny** będziemy natomiast rozumieć proces, który nie jest pasywny.

Przyjmijmy jeszcze, że w każdej chwili t stan procesu P_i reprezentuje zmienna logiczna $passive_i$, przyjmująca wartość *True*, gdy proces P_i jest pasywny, a wartość *False*, gdy proces ten jest aktywny.

Proces aktywny

Przetwarzanie rozproszone



Proces aktywny

Aktywny proces P_i ($passive_i=False$) może wysyłać i odbierać wiadomości, wykonywać tranzycje lokalne, a więc potencjalnie może również spontanicznie (w dowolnej chwili) zmienić swój stan na pasywny.

Przetwarzanie rozproszone (40)

W wielu praktycznych zastosowaniach, do analizy interesujących nas właściwości procesów wystarcza uproszczony model, w którym wyróżnia się tylko dwa stany procesów: aktywny i pasywny. **Aktywny** proces P_i ($passive_i=False$) może wysyłać i odbierać wiadomości, wykonywać tranzycje lokalne, a więc potencjalnie może również spontanicznie (w dowolnej chwili) zmienić swój stan na pasywny.

Proces pasywny

Przetwarzanie rozproszone



Proces pasywny

W stanie **pasywnym** procesu P_i ($passive_i = True$) dopuszczalne są natomiast co najwyżej zdarzenia odbioru.

Zmiana stanu procesu z pasywnego na aktywny uwarunkowana jest osiągnięciem gotowości przez choćby jedno z dopuszczalnych zdarzeń odbioru, czyli spełnieniem tak zwanego **warunku uaktywnienia**.

Przetwarzanie rozproszone (41)

W stanie **pasywnym** procesu P_i ($passive_i = True$) dopuszczalne są natomiast co najwyżej zdarzenia odbioru. Stąd zmiana stanu procesu z pasywnego na aktywny uwarunkowana jest osiągnięciem gotowości przez choćby jedno z dopuszczalnych zdarzeń odbioru, czyli spełnieniem tak zwanego **warunku uaktywnienia**.

Warunek uaktywnienia

Przetwarzanie rozproszone



Warunek uaktywnienia

Warunek uaktywnienia (ang. *activation condition*) procesu P_i związany jest ze zbiorem warunkującym $\mathcal{D}_i$, zbiorem $\mathcal{P}_i^A$, oraz predykatem $activate_i(\mathcal{X})$.

Przetwarzanie rozproszone (42)

Warunek uaktywnienia (ang. *activation condition*) P_i związany jest ze zbiorem warunkującym $\mathcal{D}_i$, zbiorem $\mathcal{P}_i^A$, oraz predykatem $activate_i(\mathcal{X})$.

Zbiór warunkujący

Przetwarzanie rozproszone



Zbiór warunkujący

Zbiór warunkujący (ang. *dependent set*), jest sumą mnogościową zbiorów $\mathcal{P}_i^S$ wszystkich zdarzeń odbioru dopuszczalnych w danej chwili.

Przetwarzanie rozproszone (43)

Zbiór warunkujący (ang. *dependent set*), jest sumą mnogościową zbiorów $\mathcal{P}_i^S$ wszystkich zdarzeń odbioru dopuszczalnych w danej chwili.

Predykat activate

Przetwarzanie rozproszone



Predykat *activate*

Predykat $activate_i(\mathcal{X})$ zdefiniowany jest w sposób następujący:

1. jeżeli $\mathcal{X} = \mathcal{D}_i$, to $activate_i(\mathcal{X}) = True$
2. jeżeli $\mathcal{X} = \emptyset$, to $activate_i(\mathcal{X}) = False$
3. jeżeli $\mathcal{X} \subset \mathcal{D}_i$ i $\mathcal{X} \neq \emptyset$, to:

$$activate_i(\mathcal{X}) \equiv \exists \mathcal{X}' :: \mathcal{X}' \neq \emptyset \wedge \mathcal{X}' \subseteq \mathcal{X} \wedge (\mathcal{P}_i^A = \mathcal{X}' \Rightarrow (passive_i \leadsto \neg passive_i))$$

gdzie

(2.15)

$passive_i \leadsto \neg passive_i$ oznacza, że pasywny proces P_i zmieni swój stan na aktywny w skończonym choć nieprzewidywalnym czasie.

Przetwarzanie rozproszone (44)

Z kolei predykat $activate_i(\mathcal{X})$ zdefiniowany jest w sposób następujący:

1. jeżeli $\mathcal{X} = \mathcal{D}_i$, to $activate_i(\mathcal{X}) = True$
2. jeżeli $\mathcal{X} = \emptyset$, to $activate_i(\mathcal{X}) = False$
3. jeżeli $\mathcal{X} \subset \mathcal{D}_i$ i $\mathcal{X} \neq \emptyset$, to:

$$activate_i(\mathcal{X}) \equiv \exists \mathcal{X}' :: \mathcal{X}' \neq \emptyset \wedge \mathcal{X}' \subseteq \mathcal{X} \wedge (\mathcal{P}_i^A = \mathcal{X}' \Rightarrow (passive_i \leadsto \neg passive_i))$$

gdzie $passive_i \leadsto \neg passive_i$ oznacza, że pasywny proces P_i zmieni swój stan na aktywny w skończonym choć nieprzewidywalnym czasie.

Predykat $activate_i(\mathcal{X})$ zachodzi zatem wtedy i tylko wtedy, gdy w związku z nadejściem (dostępnością) wiadomości od procesów tworzących zbiór $\mathcal{X}' \subseteq \mathcal{X}$ ($\mathcal{X}' = \mathcal{P}_i^A$), gotowe stanie się którekolwiek z dopuszczalnych zdarzeń odbioru. Innymi słowy predykat $activate_i(\mathcal{X})$ zachodzi, jeżeli dostarczenie wiadomości od wszystkich procesów $P_j \in \mathcal{X}$ umożliwia uaktywnienie procesu P_i . Jak łatwo zauważyć, predykat ten posiada właściwość monotoniczności: jeżeli $\mathcal{X} \subseteq \mathcal{Y}$ i $activate_i(\mathcal{X}) = True$, to $activate_i(\mathcal{Y}) = True$.

Predykat *ready*

Przetwarzanie rozproszone



Predykat *ready*

Warunek uaktywnienia procesu P_i formalnie wyraża predykat $ready_i(\mathcal{X})$:

$$ready_i(\mathcal{X}) \equiv (\mathcal{P}_i^A \supseteq \mathcal{X}) \wedge activate_i(\mathcal{X}) \quad (2.17)$$

Gdy proces jest uaktywniany, to wiadomości, których dostarczanie doprowadziło do spełnienia warunku uaktywnienia, są atomowo pobierane z buforów wejściowych i dalej przetwarzane.

Przetwarzanie rozproszone (45)

W tym kontekście, **warunek uaktywnienia** procesu może być formalnie wyrażony przez następujący predykat $ready_i(\mathcal{X})$:

$$ready_i(\mathcal{X}) \equiv (\mathcal{P}_i^A \supseteq \mathcal{X}) \wedge activate_i(\mathcal{X})$$

Gdy proces jest uaktywniany, to wiadomości, których dostarczanie doprowadziło do spełnienia warunku uaktywnienia, są atomowo pobierane z buforów wejściowych i dalej przetwarzane.

Modele żądań

Przetwarzanie rozproszone

Modele żądań

- model jednostkowy
- model AND
- model OR
- podstawowy model „ k spośród r ”
- model OR-AND
- dysjunkcyjny model „ k spośród r ”
- model predykatowy

Przetwarzanie rozproszone (46)

W praktyce, warunki uaktywnienia wyrażane są często w kategoriach tak zwanego *modelu żądań* (ang. *request model*), pozwalającego na zwięźle sformułowanie warunków uaktywnienia specyficznych dla określonej klasy zastosowań. Podejście takie odpowiada rozwiązaniom, w których procesy w trakcie wykonywania kierują do innych procesów żądania przydziału ogólnie rozumianych zasobów, lub podjęcia określonych działań. Spełnienie żądania może być potwierdzone przez przesłanie stosownej wiadomości. Tym samym, istotą modelu żądań jest wysyłanie żądań w formie wiadomości i oczekiwanie na określony zbiór wiadomości potwierdzających realizację żądania.

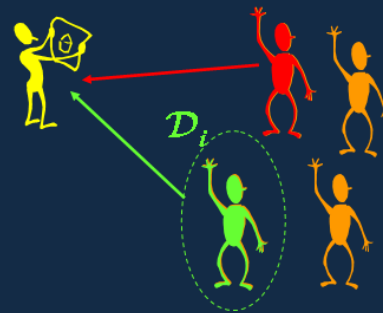


Model jednostkowy

W modelu jednostkowym warunkiem uaktywnienia pasywnego procesu jest przybycie wiadomości od jednego, ściśle określonego nadawcy.

W tym przypadku $|\mathcal{D}_i| = 1$, dla każdego naturalnego i , $1 \leq i \leq n$.

Model ten odpowiada szerokiej klasie systemów, w których procesy żądają kolejno po jednym tylko zasobie.



W **modelu jednostkowym** warunkiem uaktywnienia pasywnego procesu P_i jest przybycie wiadomości od jednego, ściśle określonego nadawcy. W tym przypadku $|\mathcal{D}_i| = 1$, dla każdego naturalnego i , $1 \leq i \leq n$. Model ten odpowiada szerokiej klasie systemów, w których procesy żądają kolejno po jednym tylko zasobie. Przykładowo może to być przydział konkretnego bufora jednostkowego w kolejnym węźle sieci komunikacyjnej z komutacją pakietów, czy potwierdzenie założenia blokady na pojedynczym obiekcie (relacji) w rozproszonej bazie danych.

Model AND

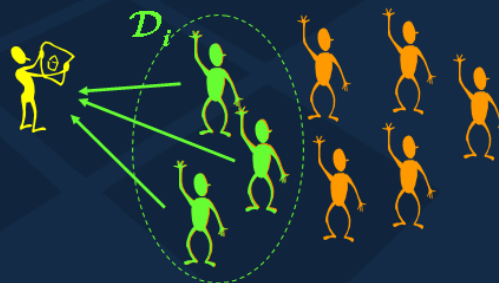
Przetwarzanie rozproszone



Model AND

W modelu AND proces pasywny staje się aktywnym, jeżeli dotarły wiadomości od wszystkich procesów tworzących zbiór warunkujący.

Model ten nazywany jest również **modelem zasobowym**.



Przetwarzanie rozproszone (48)

W **modelu AND** proces pasywny staje się aktywnym, jeżeli dotarły wiadomości od wszystkich procesów tworzących zbiór warunkujący. Model ten nazywany jest również **modelem zasobowym**. Jest on stosowany w rozwiązaniach, w których procesy ubiegają się o pewne wieloelementowe zbiory zasobów. Przykładowo, mogą to być konkretne bufory w kolejnych węzłach sieci komunikacyjnej tworzących kanał wirtualny, czy też blokady związane jednocześnie z wieloma obiektami.

Model OR

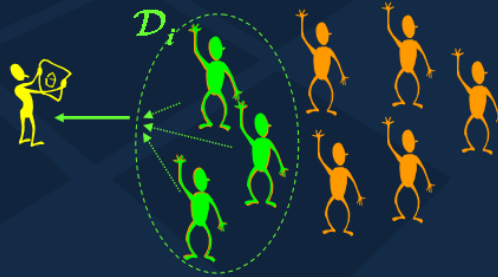
Przetwarzanie rozproszone



Model OR

W modelu OR do uaktywnienia procesu wystarczy jedna wiadomość od któregośkolwiek z procesów ze zbioru warunkującego.

Model ten nazywany jest również modelem komunikacyjnym.



Przetwarzanie rozproszone (49)

W **modelu OR** do uaktywnienia procesu wystarczy jedna wiadomość od któregośkolwiek z procesów ze zbioru warunkującego. Model ten nazywany jest również **modelem komunikacyjnym**. Odpowiada on alternatywnym strukturom programowym dostępnym między innymi w językach *CSP* i *ADA*, a także – ubieganiu się o zasoby z pewnej puli zasobów równoważnych (alternatywnych). Przykładowo, zasobami takimi mogą być pojedyncze buforów z puli buforów dostępnych w kolejnym węźle, czy też repliki obiektów. Wówczas, dostęp do jednego z zasobów z określonej puli (dostęp do jednej repliki) jest wystarczający dla kontynuowania procesu.

Podstawowy model k spośród r

Przetwarzanie rozproszone



Podstawowy model k spośród r

W podstawowym modelu k spośród r , z pasywnym procesem P_i skojarzony jest zbiór warunkujący $\mathcal{D}_i$, liczba naturalna k_i , $1 \leq k_i \leq |\mathcal{D}_i|$, oraz liczba $r_i = |\mathcal{D}_i|$.

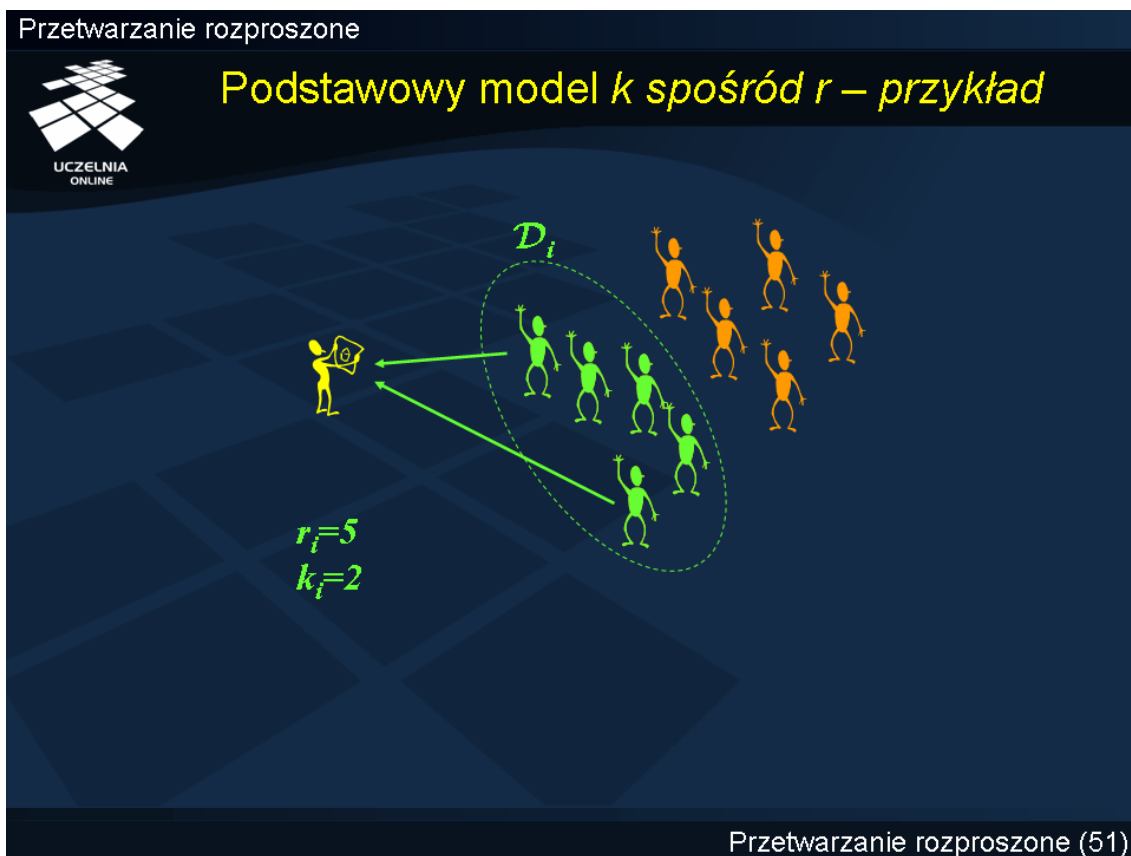
W modelu tym proces staje się aktywny tylko wówczas, gdy uzyska wiadomości od co najmniej k_i różnych procesów ze zbioru warunkującego $\mathcal{D}_i$.

Przetwarzanie rozproszone (50)

W **podstawowym modelu k spośród r** , z pasywnym procesem P_i skojarzony jest zbiór warunkujący $\mathcal{D}_i$, liczba naturalna k_i , $1 \leq k_i \leq |\mathcal{D}_i|$, oraz liczba $r_i = |\mathcal{D}_i|$. W modelu tym proces staje się aktywny tylko wówczas, gdy uzyska wiadomości od co najmniej k_i różnych procesów ze zbioru warunkującego $\mathcal{D}_i$.

Model spośród **k spośród r** jest uogólnieniem modelu AND oraz modelu OR. Istotnie, model ten redukuje się do modelu AND, gdy $k_i = r_i$, a do modelu OR, gdy $k_i = 1$. Model k spośród r , odpowiada na przykład ubieganiu się o bufory w sieciach komunikacyjnych, w których przesyłane wiadomości składają się z wielu (k) pakietów. Jeżeli w tym wypadku pojedynczy pakiet pamiętany może być w buforze jednostkowym, to przesłanie wiadomości wymaga zarezerwowania w następnym węźle puli k równoważnych buforów z ogólnej ich liczby r , w celu przesłania do nich wiadomości składającej się z k pakietów. Model k spośród r wykorzystywany jest również w systemach replikowanych baz danych, w których stosowany jest algorytm głosowania do zarządzania współbieżnością (ang. *quorum based concurrency control algorithm*). W algorytmach tych transakcje realizujące odczyt muszą odczytać k^r kopii spośród r , a transakcje realizujące zapis, muszą uzyskać dostęp do k^w kopii spośród r , przy czym $k^r + k^w > r$ oraz $2k^w > r$.

Podstawowy model k spośród r – przykład



Na slajdzie ilustrującym model k spośród r widzimy przykład, w którym do aktywacji procesu wystarczą wiadomości od dwóch z pięciu procesów należących do zbioru warunkującego.



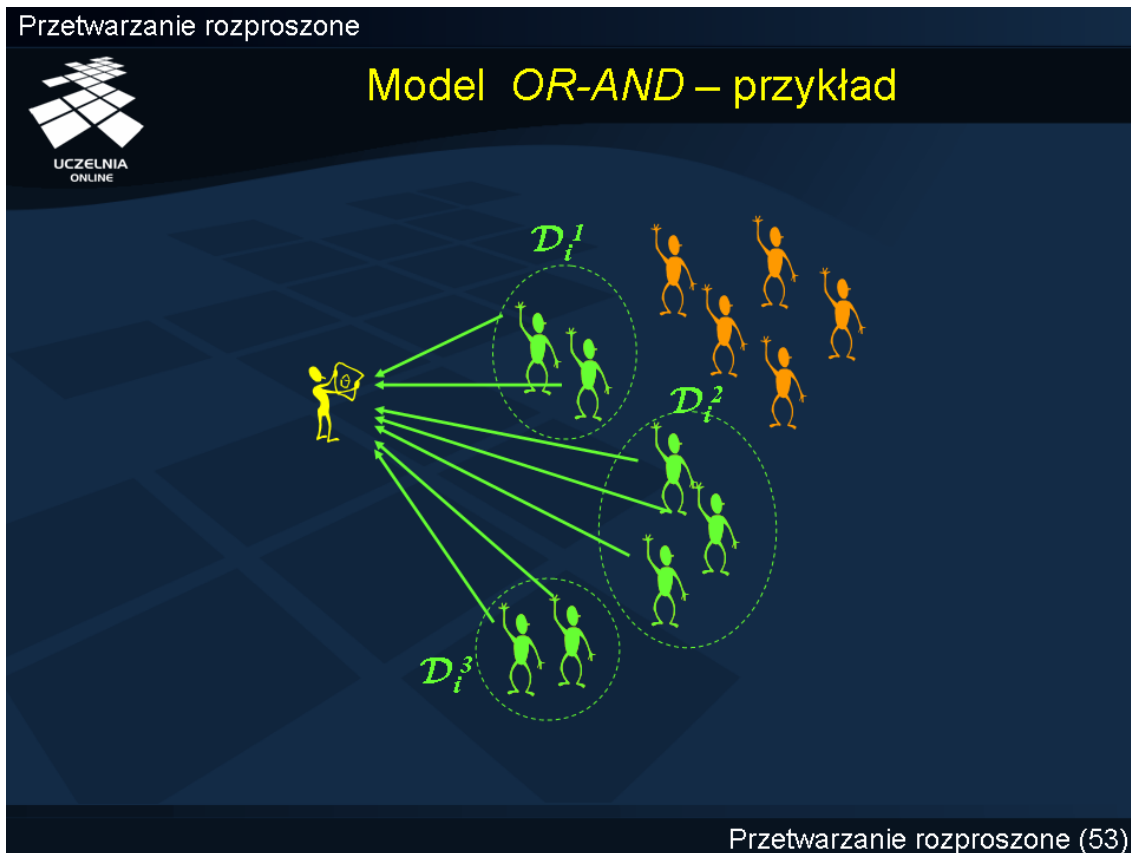
Model **OR-AND**

W modelu **OR-AND** zbiór warunkujący $\mathcal{D}_i$ pasywnego procesu P_i jest zdefiniowany jako $\mathcal{D}_i^1 \cup \mathcal{D}_i^2 \cup \dots \cup \mathcal{D}_i^{q_i}$, gdzie dla każdego naturalnego u , $1 \leq u \leq q_i$, $\mathcal{D}_i^u \subseteq \mathcal{P}$.

Proces staje się aktywny po otrzymaniu wiadomości od każdego z procesów tworzących zbiór $\mathcal{D}_i^1$ lub od każdego z procesów tworzących zbiór $\mathcal{D}_i^2$ lub ... lub od każdego z procesów tworzących zbiór $\mathcal{D}_i^{q_i}$.

W modelu **OR-AND** zbiór warunkujący $\mathcal{D}_i$ pasywnego procesu P_i jest zdefiniowany jako $\mathcal{D}_i^1 \cup \mathcal{D}_i^2 \cup \dots \cup \mathcal{D}_i^{q_i}$, gdzie dla każdego naturalnego u , $1 \leq u \leq q_i$, $\mathcal{D}_i^u \subseteq \mathcal{P}$. Proces staje się aktywny po otrzymaniu wiadomości od każdego z procesów tworzących zbiór $\mathcal{D}_i^1$ lub od każdego z procesów tworzących zbiór $\mathcal{D}_i^2$ lub ... lub od każdego z procesów tworzących zbiór $\mathcal{D}_i^{q_i}$.

Model OR-AND – przykład



Rozważmy przykładowy warunek uaktywnienia procesu P_i , sformułowany następująco: proces P_i zostanie uaktywniony po otrzymaniu wiadomości jednocześnie od procesów P_1 i P_2 , lub jednocześnie od procesów P_3, P_4 i P_5 , lub jednocześnie od procesów P_6 , i P_7 .

W tym wypadku możemy zatem przyjąć, że:

$$\mathcal{D}_i^1 = \{P_1, P_2\}$$

$$\mathcal{D}_i^2 = \{P_3, P_4, P_5\}$$

$$\mathcal{D}_i^3 = \{P_6, P_7\}$$

Model OR-AND jest ogólniejszy od podstawowego modelu k spośród r . Model ten redukuje się do modelu k spośród r , gdy podzbiory $\mathcal{D}_i$ odpowiadają wszystkim możliwym podzbiорom o liczności k zbioru warunkującego $\mathcal{D}_i$ o liczności r .

Przykładowo, jeżeli $\mathcal{D}_i = \{P_1, P_2, P_3\}$ i P_i formułuje warunek 2 spośród 3, czyli 2 spośród $\{P_1, P_2, P_3\}$, to $\mathcal{D}_i$ można wyrazić jako $\{P_1, P_2\} \cup \{P_2, P_3\} \cup \{P_1, P_3\}$, a więc jako sumę wszystkich dwuelementowych podzbiorów zbioru $\mathcal{D}_i$.



Dysjunkcyjny model k spośród r

W modelu dysjunkcyjnym k spośród r z każdym pasywnym procesem P_i skojarzony jest zbiór warunkujący $\mathcal{D}_i = \mathcal{D}_i^1 \cup \mathcal{D}_i^2 \cup \dots \cup \mathcal{D}_i^{q_i}$, liczby naturalne $k_i^1, k_i^2, \dots, k_i^{q_i}$, oraz liczby naturalne $r_i^1, r_i^2, \dots, r_i^{q_i}$, gdzie dla każdego naturalnego u , $1 \leq u \leq q_i$, $\mathcal{D}_i^u \subseteq \mathcal{P}$, $1 \leq k_i^u \leq r_i^u = |\mathcal{D}_i^u|$.

Proces staje się aktywny po otrzymaniu wiadomości od k_i^1 różnych procesów ze zbioru $\mathcal{D}_i^1$, lub k_i^2 wiadomości od różnych procesów ze zbioru $\mathcal{D}_i^2$, lub ... lub k_i^u wiadomości od różnych procesów ze zbioru $\mathcal{D}_i^{q_i}$.

W modelu dysjunkcyjnym k spośród r z każdym pasywnym procesem P_i skojarzony jest zbiór warunkujący $\mathcal{D}_i = \mathcal{D}_i^1 \cup \mathcal{D}_i^2 \cup \dots \cup \mathcal{D}_i^{q_i}$, liczby naturalne $k_i^1, k_i^2, \dots, k_i^{q_i}$, oraz liczby naturalne $r_i^1, r_i^2, \dots, r_i^{q_i}$, gdzie dla każdego naturalnego u , $1 \leq u \leq q_i$, $\mathcal{D}_i^u \subseteq \mathcal{P}$, $1 \leq k_i^u \leq r_i^u = |\mathcal{D}_i^u|$. Proces staje się aktywny po otrzymaniu wiadomości od k_i^1 różnych procesów ze zbioru $\mathcal{D}_i^1$, lub k_i^2 wiadomości od różnych procesów ze zbioru $\mathcal{D}_i^2$, lub ... lub k_i^u wiadomości od różnych procesów ze zbioru $\mathcal{D}_i^{q_i}$.

Dysjunkcyjny model k spośród r – przykład

Przetwarzanie rozproszone



Dysjunkcyjny model k spośród r

Model dysjunkcyjny k spośród r redukuje się do:

- modelu OR-AND, gdy $k_i^u = |\mathcal{D}_i^u|$ dla każdego u
- modelu podstawowego k spośród r , gdy $q_i=1$
- modelu AND, gdy $q_i=1$ i $k_i^1 = |\mathcal{D}_i^1|$
- modelu OR, gdy $q_i=1$ i $k_i^1=1$

Przetwarzanie rozproszone (55)

Należy zauważyć, że model dysjunkcyjny k spośród r redukuje się do modelu *OR-AND* (gdy $k_i^u = |\mathcal{D}_i^u|$ dla każdego u) – i w konsekwencji do innych modeli. W istocie, model podstawowy k spośród r uzyskuje się gdy $q_i=1$; model *AND* – gdy $q_i=1$ i $k_i^1 = |\mathcal{D}_i^1|$; a w końcu model *OR* - gdy $q_i=1$ i $k_i^1=1$.

Przetwarzanie rozproszone

Model predykatowy



W modelu predykatowym, dla każdego pasywnego procesu P_i ze zbiorem warunkującym $\mathcal{D}_i$ określony jest predykat $activate_i(\mathcal{X})$, gdzie $\mathcal{X} \subseteq \mathcal{P}$.

Jak łatwo zauważyć, stosownie definiując predykat $activate_i(\mathcal{X})$ można oczywiście uzyskać wszystkie wcześniej omówione modele żądań.

Przetwarzanie rozproszone (56)

W **modelu predykatowym**, dla każdego pasywnego procesu P_i ze zbiorem warunkującym $\mathcal{D}_i$ określony jest predykat $activate_i(\mathcal{X})$, gdzie $\mathcal{X} \subseteq \mathcal{P}$.

Jak łatwo zauważyć, stosownie definiując predykat $activate_i(\mathcal{X})$ można oczywiście uzyskać wszystkie wcześniej omówione modele żądań.

Rozważmy na początek dysjunkcyjny model k spośród r , w którym żądania procesu P_i są zdefiniowane w kategoriach podzbiorów $\mathcal{D}_i^1, \mathcal{D}_i^2, \dots, \mathcal{D}_i^{q_i}$ oraz stałych naturalnych $k_i^1, k_i^2, \dots, k_i^{q_i}$ i $r_i^1, r_i^2, \dots, r_i^{q_i}$. Wówczas, predykat można wyrazić następująco:

$$activate_i(\mathcal{X}) \equiv (\exists u :: 1 \leq u \leq q_i :: |\mathcal{D}_i^u \cap \mathcal{X}| \geq k_i^u)$$

Definicje predykatu dla innych modeli można wyprowadzić z powyższego sformułowania w sposób następujący:

model podstawowy k spośród r – przyjmując, że $q_i=1$, $\mathcal{D}_i^1 = \mathcal{D}_i$ i dalej:

$$activate_i(\mathcal{X}) \equiv |\mathcal{D}_i \cap \mathcal{X}| \geq k_i^1$$

model OR-AND – przyjmując, że $\forall u :: 1 \leq u \leq q_i :: k_i^u = |\mathcal{D}_i^u|$ i dalej:

$$activate_i(\mathcal{X}) \equiv (\exists u :: 1 \leq u \leq q_i :: \mathcal{D}_i^u \subseteq \mathcal{X})$$

model OR – przyjmując, że $\forall u:: 1 \leq u \leq q_i :: |\mathcal{D}_i^u| = 1 \wedge k_i^u = 1 \wedge \mathcal{D}_i = \mathcal{D}_i^1 \cup \mathcal{D}_i^2 \cup \dots \cup \mathcal{D}_i^{q_i}$ i dalej:

$$activate_i(\mathcal{X}) \equiv \mathcal{D}_i^u \cap \mathcal{X} \neq \emptyset$$

model AND - przyjmując, że $q_i = 1$, $\mathcal{D}_i^1 = \mathcal{D}_i$, $k_i^1 = r_i^1 = |\mathcal{D}_i^1| = |\mathcal{D}_i|$ i dalej

$$activate_i(\mathcal{X}) \equiv \mathcal{D}_i \subseteq \mathcal{X}$$