Na prawach rękopisu

INSTYTUT INFORMATYKI, AUTOMATYKI I ROBOTYKI
POLITECHNIKI WROCŁAWSKIEJ

Raport serii: PREPRINTY nr 24/2005

Analiza zależności czasowych w drzewach niezdatności systemów związanych z bezpieczeństwem

(rozprawa doktorska)

Paweł SKROBANEK

PROMOTOR:
prof. dr hab. inż. Jan Magott

Słowa kluczowe:
- **bezpieczeństwo systemów,**
- **drzewa niezdatności (błędów),**
- **drzewa niezdatności z zależnościami czasowymi,**
- **czasowe sieci Petri'ego,**
- **analiza drzew niezdatności z zależnościami czasowymi**

Wrocław, czerwiec 2005

» Niniejszą pracę dedykuję dwóm osobom, których obecność w moim życiu doprowadziła i umożliwiła jej realizację. «

Ukochanej Żonie - Justynce

oraz

Promotorowi Janowi Magottowi

Spis treści

1. Bezpieczeństwo systemów komputerowych

1.1. Systemy komputerowe

Z rozwojem cywilizacyjnym człowieka nierozerwalnie związane są wynalazki. Jednym z ostatnich są komputery. Ciągle rozszerzający się obszar ich zastosowań oraz obszar zastosowań urządzeń sterowanych przez komputery, oprócz niewątpliwych korzyści, może nieść ze sobą również i niebezpieczeństwa.

Ponieważ bez systemów komputerowych trudno byłoby sobie dziś poradzić, oczekujemy, by były one odpowiednio zaprojektowane i zbudowane - by były *rzetelne*, inaczej *wiarygodne* (*ang. dependability).*

System komputerowy -1) [Ż95] system złożony zwykle z jednego lub kilku komputerów wraz z oprogramowaniem, używających wspólnej pamięci dla całości lub części programu, a także dla całości lub części danych niezbędnych do wykonania programu,

2) [Ur01] "kompletny zestaw sprzętu wraz z posadowionym na nim oprogramowaniem oraz połączonymi bezpośrednio z nim urządzeniami zewnętrznymi (...)".

Rzetelność systemu komputerowego - atrybut (cecha, właściwość) systemu komputerowego pozwalająca na uzasadnione poleganie na usługach oferowanych przez ten system.

Ponieważ nikomu i niczemu nie należy ufać „bezgranicznie" - jak mawiają ludzie, dlatego sprawdzamy rzetelność systemów pod różnymi aspektami. Dokonujemy dekompozycji rzetelności na [Górski00]:

- *Niezawodność (ang. reliability)* - określa zdolność systemu do nieprzerwanej pracy w określonych warunkach otoczenia.
- *Dyspozycyjność (ang. availability)* - atrybut charakteryzujący procent czasu, w którym system jest zdolny do świadczenia oczekiwanych usług.
- *Bezpieczeństwo (ang. safety)* – cecha systemu mówiąca o tym, że system swoim działaniem bezpośrednio lub pośrednio nie spowoduje np. zniszczeń, strat materialnych, zagrożenia życia.

- *Zabezpieczenie (ang. security)* - przed niepowołanym dostępem (np. zabezpieczenie przed atakami terrorystycznymi, kradzieżą danych, ingerencją w funkcjonowanie systemu).

Wymienione wyżej atrybuty podlegają dalszej dekompozycji, a w zależności od pełnionych przez system komputerowy funkcji i jego otoczenia, pełnią mniejszą lub większą rolę w procesie powstawania, a następnie przy użytkowaniu systemu.

Ze względu na zakres pracy, w dalszej części będę zajmował się jedynie *bezpieczeństwem* systemów komputerowych. Omówienie pozostałych atrybutów można znaleźć np. w [Laprie89], [Górski00], [Ż95].

1.2. Bezpieczeństwo systemów

Komputer już od dawna wspomaga pracę człowieka i już dawno, mógł się człowiek przekonać, że nieprzewidziane skutki jego działania oraz niewielki błędy w oprogramowaniu mogą być tragiczne.

Książkowym przykładem w tej materii jest system medyczny Therac-25 [LT93]. Zadaniem systemu było niszczenie tkanki nowotworowej u pacjenta leżącego na stole wiązką promieniowania X. Wiązka ta, by nie powodować skutków ubocznych, musiała być zogniskowana dokładnie na chorej tkance. Ponieważ, ze względów bezpieczeństwa, operator znajdował się podczas zabiegu w innym pomieszczeniu, sterowaniem zajmował się komputer.

W latach 1985 - 1987 miało miejsce sześć wypadków śmiertelnych spowodowanych nadmiernym napromieniowaniem (niewłaściwe zogniskowanie wiązki na skutek nie ustawienia stołu z pacjentem).

Jak pokazała analiza, której krótki zarys można znaleźć w [GW94], wypadki miały miejsce na skutek przepełnienia zmiennej *class3* typu *bajt*. Przy każdym wywołaniu procedury testującej, zwiększana była wartość *class3* o *1*, natomiast wartość *0* oznaczała, że pozycja stołu jest ustawiona. Przepełnienie, jak nietrudno się domyślić, miało miejsce wówczas, gdy do *255* procedura test dodała *1* (zmienna przyjmowała wartość 0). Uruchomienie wówczas terapii mogło i doprowadziło do śmierci ludzi.

Oprogramowanie, samo w sobie, nie stanowi zagrożenia. Niebezpieczne mogą być natomiast skutki jego działania - nawet, jak pokazuje powyższy przykład, z pozoru błahe błędy mogą prowadzić do tragedii. Wymagania dotyczące

oprogramowania w systemach związanych z bezpieczeństwem opisuje [PN-EN 61508-3].

I chociaż bezpieczeństwo jest tylko jednym z atrybutów określających właściwości systemu komputerowego, to niewątpliwie jednym z ważniejszych.

Wypadek (ang. accident) - zdarzenie powodujące w konsekwencji określone straty (np. materialne, zanieczyszczenie środowiska), czy też zranienie, a nawet śmierć ludzi.

Bezpieczeństwo (ang. safety) - brak wypadków (lub bardzo małe prawdopodobieństwo zaistnienia sytuacji, w których mogą wystąpić) spowodowanych użytkowaniem systemu.

System związany z bezpieczeństwem (ang. *safety - related system*) - taki, który bezpośrednio lub poprzez zainicjowanie zdarzeń może prowadzić do wypadku.

Jak zatem projektować system bezpieczny?

Idealnym rozwiązaniem byłoby wykluczyć w ogóle sytuacje związane z hazardem. Tak stanowiło na przykład prawo stanu Kansas (początek ubiegłego wieku), zacytowane w [SPG91]: „Jeśli dwa pociągi zbliżają się do siebie krzyżując swe tory, to oba powinny się całkowicie zatrzymać i nie ruszać ponownie do czasu, aż drugi z nich nie odjedzie". Wówczas do sytuacji niebezpiecznej zapewne nie dojdzie, ale również system nie będzie funkcjonował -pociągi nigdy nie ruszą.

Podobnie możemy zabezpieczyć przejazd drogowy przez tory, tzw. rogatki. Jeśli zamkniemy „na zawsze" rogatki (takiej konstrukcji, by zdenerwowani kierowcy nie mogli ich sforsować), to sytuacja niebezpieczna zdefiniowana jako *przejeżdżający pociąg przy niezamkniętych rogatkach* nigdy nie wystąpi.

Ponieważ powyższe rozwiązanie, ze względów funkcjonalnych nie jest możliwe - mówimy, że system nie może pełnić swojej *misji*, to pozostaje tylko tak zaprojektować, wykonać, a następnie użytkować system, by nie dopuścić do wystąpienia sytuacji niebezpiecznych, tzw. *hazardów*. Wystąpienie takiej sytuacji nie jest równoznaczne z wypadkiem, ale z zaistnieniem warunków, w których może on wystąpić. Zatem:

Hazard (ang. hazard) jest to sytuacja, która może prowadzić do wystąpienia wypadku.

W naszym kraju było w 2003 roku, według NIK (dane podane w [TVP]), było 80% rogatek podatnych na awarie (starszy typ sterowania) prowadzące do sytuacji niebezpiecznej i około 80 tysięcy niezabezpieczonych przejazdów. Liczba osób rannych (wypadki na przejazdach kolejowych) wyniosła 266 osób, a zabitych 63 osoby.

Na etapie projektowania i analizy systemu *hazard* opisujemy jako zbiór określonych warunków, których zaistnienie w określonym środowisku może prowadzić do wypadku.

Ponieważ w większości sytuacji nie jesteśmy w stanie wyeliminować *hazardu*, za system bezpieczny uważa się taki system, w którym stopień zagrożenia, tzw. *ryzyko* jest na akceptowalnym poziomie. Przy czym:

Ryzyko - miara stopnia zagrożenia, wyrażana jako funkcja [Leveson87]:

1) prawdopodobieństwa wystąpienia hazardu,
2) prawdopodobieństwa, że hazard spowoduje wypadek,
3) najgorszych możliwych strat poniesionych w wypadku.

Osiągnięcie określonego poziomu bezpieczeństwa można uzyskać poprzez odpowiednią konstrukcję systemu, dodanie zintegrowanych elementów zabezpieczających lub też zabezpieczeń w postaci niezależnych urządzeń. Normy, dla systemów związanych z bezpieczeństwem wymagają określonego poziomu integralności bezpieczeństwa = integralności zabezpieczeń -w języku polskim stosowane są oba określenia.

Integralność zabezpieczeń (ang. *safety integrity*) - prawdopodobieństwo, że w zaprojektowanym, a następnie wykonanym i uruchomionym systemie elementy zabezpieczające spełnią swoje zadanie.

Norma [IEC1508] wyróżnia cztery poziomy integralności zabezpieczeń zalecając jednocześnie stosowanie określonych metod analizy bezpieczeństwa dla

określonych etapów życia systemu (podczas projektowania, implementacji i wdrożenia oraz podczas eksploatacji).

Omówienie zagadnień związanych z bezpieczeństwem oprogramowania w oparciu o normę IEC 1508 można znaleźć w [Sacha97].

1.3. Metody analizy bezpieczeństwa

W celu uniknięcia błędów w projektowanych systemach, stosuje się odpowiednie techniki analizy.

Ze względu na rodzaj analizy, techniki możemy podzielić na:

- Probabilistyczne, pozwalające na wyznaczenie między innymi prawdopodobieństwa wystąpienia hazardu, gotowości systemu, czy oszacowania oczekiwanego czasu pracy systemu do wystąpienia pierwszej awarii.
- Deterministyczne, odpowiadające w jednoznaczny sposób, czy hazard może w danym systemie wystąpić, ze względu na określone przyczyny.

Norma IEC 1508 zaleca stosowanie analizy *hazardów* już w fazie analizy i specyfikacji wymagań oprogramowania. Tak jak w przypadku różnych systemów (niekoniecznie komputerowych), tak i w przypadku analizy bezpieczeństwa oprogramowania zalecenia dotyczą między innymi takich metod, jak:

1. Diagramy przyczynowo-skutkowe CCD (ang. Cause-Consequence Diagrams), [LJ97], [Ni71],

2. Analiza drzew zdarzeń ETA (ang. Event Trees Analysis) [LJ87],

3. Analiza drzew niezdatności (błędów*) FTA (ang. Fault Trees Analysis) [PN-IEC 1025], [NUREG81],

4. Analiza rodzajów i skutków uszkodzeń FMEA (ang. Failure Mode and Effects Analysis) [IEC 812] oraz jej rozszerzenie o ocenę skutków krytycznych FMECA (ang. Failure Mode, Effects and Criticality Analysis),

5. Analiza hazardu i gotowości systemu HAZOP (ang. Hazard and Operability Analysis) [IEC 61882], [HAZOP].

* W literaturze można spotkać tłumaczenie *Fault Tree Analysis* -jako *Analiza Drzew Błędów* (takie tłumaczenie jest stosowane również przez wiele osób zajmujących się zagadnieniami związanymi z bezpieczeństwem) lub *Analiza Drzew Niezdatności* (tak brzmi tłumaczenie wg normy PN-IEC 1025).

Informacje dotyczące stosowanych technik dla systemów związanych z bezpieczeństwem, można znaleźć w [PN-EN-61508-7].

Jedną z często stosowanych technik jest analiza drzew niezdatności (FTA) [Vesely81], [Górski94], [GW95]. Klasyczna analiza FTA nie obejmuje jednak analizy zależności czasowych pomiędzy zdarzeniami.

Możliwa jest np. taka sytuacja:

Wyznaczymy minimalny zbiór przyczyn, które mogą prowadzić do wystąpienia hazardu. Ze względu jednak na zależności czasowe, niektóre ze zdarzeń nie będą mogły wystąpić (lub wystąpią w takich momentach czasu względem innych, że nie mogą prowadzić do sytuacji niebezpiecznej), a zatem do wystąpienia *hazardu* nie dojdzie.

W celu sprawdzenia takich zależności, stosuje się inne techniki, jako uzupełnienie FTA. Takie podejście możemy znaleźć w pracy [Shimeall91]. Przedstawiono w niej przykład analizy oprogramowania systemu odpalania pocisków *missiles*. FTA została wykorzystana do znalezienia krytycznych (ze względu na bezpieczeństwo) zdarzeń, natomiast do analizy kolejności czasowej zdarzeń w systemie wieloprocesorowym wykorzystano czasową sieć Petri'ego (ang. *TPN –Time Petri Net*). Także w pracy [LS87] odnajdziemy przykłady zastosowania TPN do modelowania i analizy systemów czasu rzeczywistego. W pracy tej została pokazana między innymi metoda wstecznej analizy sieci Petri'ego - nie obejmuje ona jednak analizy czasowej, a jedynie analizę osiągalności pewnych stanów.

Zastosowanie czasowych sieci Petri'ego (ang. *TPNs –Time Petri Nets*) do badania zależności czasowych występujących w drzewach niezdatności, opisano w pracach: [GMW95], [W96]. Jednym z problemów analizy TPNs jest liczba możliwych stanów czasowej sieci Petri'ego (TPN), która dla niewielkiego drzewa niezdatności jest dość duża; np. dla TPN modelującej trzy bramki – dla systemu palnika gazowego [GMW95], otrzymujemy 533 klasy [S97]. Prezentowana metoda w tym opracowaniu pozwala na ograniczenie liczby możliwych klas, poprzez wsteczną (od skutku do przyczyn) analizę TPN.

1.4. Cel, teza pracy

Celem pracy jest opracowanie metody formalnej umożliwiającej analizę zależności czasowych pomiędzy zdarzeniami drzew niezdatności. Cel ten ma zostać

osiągnięty poprzez opracowanie metody analizy czasowej sieci Petri'ego (ang. *time Petri net*) modelującej drzewo niezdatności "od góry do dołu", czyli od skutku do przyczyny.

Tezę pracy można sformułować następująco:

Istnieje formalna metoda analizy drzew niezdatności z zależnościami czasowymi (na podstawie wstecznej analizy czasowych sieci Petri'ego modelujących drzewa niezdatności), która pozwoli na analizę bardziej złożonych drzew niezdatności niż w przypadku klasycznej analizy czasowych sieci Petri'ego modelujących drzewa niezdatności oraz zapewni jednoznaczność, precyzję i automatyzację procesu analizy.

Klasyczna analiza drzew niezdatności pozwala jedynie na identyfikację minimalnych zbiorów przyczyn prowadzących do wystąpienia hazardu. Poszerzenie FTA o analizę zależności czasowych pozwala nie tylko wskazać zbiory zdarzeń, których równoczesne wystąpienie może prowadzić do hazardu, ale również pozwala na określenie zależności czasowych pomiędzy nimi. Znajomość relacji czasowych pozwala z kolei np. na odpowiednią konstrukcję zabezpieczeń lub modyfikację systemu pozwalającą na uniknięcie hazardu.

Rozważmy teoretyczny przykład. Jeśli klasyczna analiza pokaże, że do hazardu prowadzi zainicjowanie dwóch reakcji (obie wydzielają energię cieplną) w dwóch kadziach znajdujących się w jednym izolowanym pomieszczeniu na skutek przekroczenia dopuszczalnej temperatury maksymalnej w tym pomieszczeniu, to (jeśli nie jest możliwe zwiększenie ilości odprowadzanego ciepła w jednostce czasu z pomieszczenia) najprostszym zabezpieczeniem będzie sekwencyjne przeprowadzanie reakcji. Taki sposób pracy powoduje jednak zmniejszenie wydajności. Analiza zależności czasowych może dostarczyć nam wówczas dodatkowej wiedzy, z której może wynikać np., iż wystarczy zainicjować drugą reakcję nie wcześniej niż 10s po zainicjowaniu pierwszej.

Dokładne omówienie obszaru zastosowań i ograniczeń metody znajduje się w rozdziale 8.

Opracowanie formalnej metody analizy zależności czasowych występujących w drzewach niezdatności pozwoliłoby na:

- jednoznaczność i precyzję analizy,

- automatyzację procesu analizy (konstrukcja odpowiednich narzędzi),
- uniezależnienie procesu analizy od ekspertów.

1.5. Struktura opracowania

Rozdział drugi pracy dotyczy sposobu budowy i notacji drzew niezdatności. W pierwszej części, pokazana jest notacja graficzna bramek, zgodnie z [IEC 1025]. W dalszej części rozdziału, pokazana została definicja czterech podstawowych typów bramek logicznych, przyczynowych: XOR i AND oraz uogólniających: XOR i AND, z uwzględnieniem notacji czasu trwania zdarzeń. Końcowa część rozdziału zawiera przykład drzewa niezdatności dla systemu rozjazdu kolejowego.

W rozdziale trzecim zostały zawarte definicje oraz przykłady związane z czasowymi sieciami Petri'ego, natomiast rozdział czwarty, pokazuje zastosowanie czasowych sieci Petri'ego do modelowania drzew niezdatności, zgodnie z [GMW95].

Rozdział piąty zawiera podstawy opracowanej metody INES. Analiza czasowej sieci Petri'ego jest prowadzona „od skutku do przyczyn". Opracowane w tym rozdziale wzory (z wykorzystaniem czasowej sieci Petri'ego) dla każdej z czterech typów bram, pozwalają, jak to zostanie pokazane, na „odejście" od modelu czasowej sieci Petri'ego i analizę formalnie zapisanego drzewa niezdatności.

Rozdział szósty traktuje o formalnej reprezentacji drzewa niezdatności z zależnościami czasowymi.

Algorytm przeprowadzenia analizy drzewa niezdatności z zależnościami czasowymi, przykład takiej analizy oraz wskazanie zastosowań i ograniczeń metody – to kolejne rozdziały: siódmy i ósmy. Uszczegółowienie algorytmów zawiera załącznik A.

Podsumowanie opracowania znajduje się w rozdziale 9.

2. Drzewa niezdatności

2.1. Wprowadzenie

Jedną z najczęściej stosowanych obecnie technik analizy bezpieczeństwa jest analiza drzew niezdatności -FTA (Fault Tree Analysis) [BFS75]. Jak to zaznaczyłem w rozdziale 1, specyfikację FTA możemy znaleźć w międzynarodowych normach [IEC1025], [NUREG81] oraz w polskim tłumaczeniu normy IEC 1025 [PN-IEC1025]. Technika FTA jest również zalecana w normie [IEC1508] jako jedna z technik do analizy bezpieczeństwa oprogramowania.

Ciekawe omówienie zagadnień bezpieczeństwa oprogramowania w normie IEC1508 znajduje się w pracy [Sacha97].

Analiza FTA polega na rozpoznaniu hazardów mogących wystąpić w analizowanym systemie. Na przykład, hazardem dla przejazdu kolejowego jest zdarzenie, które możemy nazwać: *niezabezpieczony przejazd,* rozumiane jako współwystępowanie zdarzeń: *przejeżdżający pociąg, otwarte rogatki*.

Dla każdego hazardu konstruowane jest drzewo przyczyn, które mogą prowadzić do jego wystąpienia. Po zastosowaniu określonej techniki analizy, otrzymujemy wyniki dostarczające nam informacji, czy hazard w rozważanym systemie może wystąpić. W praktyce zarysowują się trzy główne obszary analizy:

a) Klasyczna -pozwalająca na określenie minimalnych zbiorów przyczyn, tzn. takich, że wystąpienie wszystkich zdarzeń ze zbioru jest warunkiem koniecznym i wystarczającym do wystąpienia hazardu.
b) Klasyczna probabilistyczna -umożliwiająca dodatkowo określenie prawdopodobieństwa hazardu i błędów [PP88].
c) Czasowa (w oparciu o klasyczną analizę TPNs [GMW95], [W96], [Sk96] lub w oparciu o wsteczną analizę TPNs modelujących drzewa niezdatności [MS00], [MS02] oraz prezentowana w niniejszym opracowaniu wraz z algorytmem analizy) - umożliwiająca określenie relacji czasowych pomiędzy zdarzeniami prowadzącymi do hazardu.

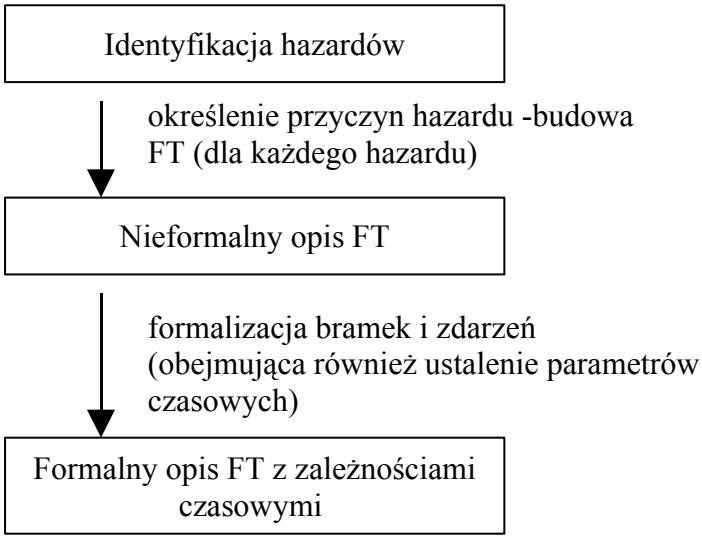
Poszerzenie klasycznej analizy o wyznaczenie zależności czasowych pomiędzy zdarzeniami drzewa niezdatności (ang. *FT*[1] *– Fault Tree*) dla większych drzew zawiera

[1] W dalszej części, jeśli będzie mowa o drzewie niezdatności (innymi słowy: drzewie błędów), będę używał skrótu FT. Ze względu na dwie nazwy stosowane w języku polskim, nie zdecydowałem się przyjąć skrótu np. DN lub DB.

jedynie metoda pokazana w [W96]. Polega ona na uzyskaniu minimalnych zbiorów przyczyn poprzez klasyczną analizę, a następnie wyznaczenie dla każdego takiego zbioru zależności czasowych.

Przedmiotem tej pracy jest metoda definiowania zależności czasowych w drzewie niezdatności oraz analizy całego drzewa z zależnościami czasowymi. Zostanie ona pokazana w rozdziale 6 i 7. Prezentowana metoda pozwala na określenie zależności czasowych zarówno dla zdarzeń z minimalnych zbiorów przyczyn, jak i dla zdarzeń pośredniczących (na ścieżce pomiędzy przyczynami i hazardem).

Proces projektowania drzewa niezdatności pokazuje rysunek 1.



Rys. 1. Proces konstrukcji drzewa niezdatności z zależnościami czasowymi

2.2. Nieformalna reprezentacja

PN-IEC 1025 podaje, iż analizę powinno poprzedzać:

- **Zdefiniowanie zakresu analizy.**

  Taka definicja obejmuje najczęściej: definicję systemu, określenie celu i zakresu analizy, ustalenie założeń (np. dotyczących warunków funkcjonowania systemu, jakości stosowanych elementów i podzespołów, pewnych uproszczeń opisu procesów fizycznych, przyjęcia niezawodności pewnych elementów lub określonego ich stanu w przypadku awarii), ustalenie poziomu szczegółowości, a nawet -nakładów finansowych oraz czasu i potrzebnych zasobów.

- **Zapoznanie się z zadaniami, funkcjonowaniem i otoczeniem projektowanego systemu.**

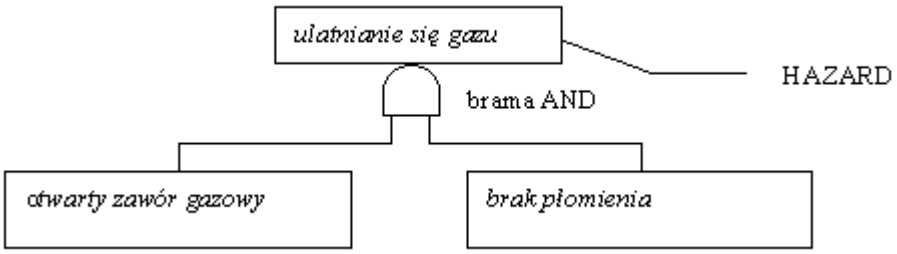
  Znajomość zarówno systemu jak i jego otoczenia jest niezbędna do określenia hazardów i ich przyczyn, innymi słowy: do prawidłowego skonstruowania FT. Na tym etapie pozyskujemy niezbędną wiedzę specjalistyczną na temat systemu oraz określamy wszystkie warunki zewnętrzne mogące mieć wpływ zarówno na sam system, jak i na jego działanie.

Analizę rozpoczynamy od identyfikacji sytuacji niebezpiecznych (hazardów), które mogą wystąpić w projektowanym systemie. Identyfikację taką najczęściej przeprowadzamy z udziałem ekspertów projektujących system.

Znając hazard, określamy jego bezpośrednie przyczyny łącząc je określoną bramką (np. AND, OR, XOR). W kolejnym kroku określamy "przyczyny przyczyn", itd. W każdym kroku "schodzimy" jeden poziom niżej. Konstrukcję FT kończymy, jeśli nie potrafimy już określić przyczyn rozważanego zdarzenia lub osiągnęliśmy wcześniej założony poziomie szczegółowości. Przykład drzewa niezdatności dla rozjazdu kolejowego znajduje się w sekcji 2.4.

Ze względu na to, iż identyfikacja hazardów i ich przyczyn jest objęta tajemnicą (ochrona danych przez firmy), nie ma opracowanych np. banków danych z drzewami niezdatności funkcjonujących systemów i ich analizą. Jest to zatem proces twórczy i w pierwszym etapie może zawierać niejednoznaczności.

Rozważmy następującą sytuację: naszym systemem jest palnik w kuchence gazowej, sytuacja niebezpieczna -*ulatnianie się gazu* jest wynikiem współwystępowania dwóch zdarzeń: o*twarty zawór gazowy* i *brak płomienia*. Graficznie:



Rys. 2. Przyczyny hazardu: *ulatnianie się gazu*

Jest to jednak opis nieprecyzyjny. Taka sytuacja zawsze ma miejsce przez pewien czas. Niebezpieczeństwo pojawia się dopiero, jeśli *ulatnianie się gazu* trwa zbyt długo. Analiza systemu palnika gazowego została pokazana w [GMW95].

2.3. Zapis formalny

2.3.1. Wprowadzenie

Rezultaty analizy są wiarygodne, jeśli model FT jest poprawnie skonstruowany, a stosowane metody analizy - sprawdzone. Dlatego ważne jest, by zarówno opis jak i analizę przeprowadzać metodami formalnymi. Formalny opis drzewa niezdatności pozwala na:

- uniezależnienie opisu od eksperta,
- jednoznaczność i czytelność opisu,
- jednoznaczność uzyskanych wyników,
- niezależną kontrolę procesu analizy,
- opracowanie standardów,
- opracowanie narzędzi wspomagających budowę oraz analizę FT, np. [Relex], [Jarmuż96], INESv1 (prototyp narzędzia opracowany w oparciu o algorytmy opisane w niniejszej pracy),
- porównanie wyników uzyskanych różnymi metodami.

Formalny opis drzewa przy pomocy notacji ECSDM (ang. *Extended Common Safety Description Model*) z uwzględnieniem czasu, pokazany został w [Górski94], notacja graficzna bram i zdarzeń - w [IEC 1025] oraz w [GMW95], przy czym, w [GMW95]z uwzględnieniem parametrów czasowych.

W tej pracy, zostanie pokazane stosowania modelu graficznego. Dodatkowo, posłużę się definicjami bramek z uwzględnieniem zależności czasowych [GMW95]. W bieżącej wersji, metoda, którą nazwałem INES (ang. *Inequalities – Equalities System*) pozwala na analizę drzew niezdatności zbudowanych z czterech typów bram:

- Uogólniającej AND (z ang. *Generalization AND*),
- Uogólniającej XOR (z ang. *Generalization XOR*),
- Przyczynowej AND (z ang. *Causal AND*),
- Przyczynowej XOR (z ang. *Causal XOR*).

Formalizacja zdarzeń drzewa niezdatności może rozpoczynać się od utworzenia opisu wszystkich elementów znajdujących się w systemie i określenia, dla każdego z nich, wszystkich stanów, jakie mogą przyjmować. Proces formalizacji zdarzeń został dokładnie przedstawiony w pracy [GW95], zgodnie z symboliką opisaną w [Jones90]. Formalna notacja dla czterech typów bramek, które wykorzystane zostały w opracowaniu prezentowanej metody, zostanie omówiona w rozdziałach 2.3.3 - 2.3.6.

2.3.2. Bramki - symbole graficzne

W drzewie niezdatności wyróżniamy dwa zasadnicze rodzaje elementów: zdarzenia (ang. *events*) i łączące je bramki (ang. *gates*). Dokładny opis można znaleźć w [IEC 1025]. Zdarzenia opisujemy w prostokątach, natomiast do opisu bramek norma zaleca użycie symboli pokazanych na rysunku 3.

bramka wykluczająca LUB (ang. exclusive OR)	bramka I (ang. AND)	bramka LUB (ang. OR)	bramka NIE (ang. NOT)
zdarzenie na wyjściu zachodzi, gdy zachodzi jedno ze zdarzeń na wejściu, przy czym na wejściu może zachodzić tylko jedno zdarzenie	zdarzenie wyjściowe zachodzi, gdy jednocześnie zachodzą wszystkie zdarzenia na wejściu	zdarzenie na wyjściu zachodzi wówczas, gdy zachodzi przynajmniej jedno zdarzenie na wejściu	zdarzenie wyjściowe jest zaprzeczeniem sytuacji na wejściu

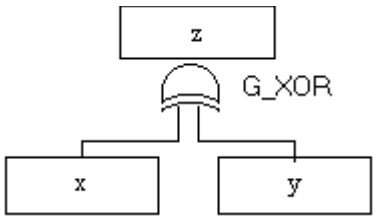
Rys. 3. Symbole graficzne bramek

Zastosowanie powyższych symboli dla potrzeb analizy czasowej nie jest jednak precyzyjne. W kolejnym rozdziale przedstawię definicję czterech podstawowych bramek oraz propozycję formalnego sposobu ich notacji graficznej. Modele bram AND oraz OR zostały zaprezentowane w [GMW95].

Bramki, które zostaną omówione są dwojakiego rodzaju, pierwszy rodzaj to bramka uogólniająca (z ang. *generalization*), drugi - przyczynowa (z ang. *causal*). Zasadnicza różnica jest taka, że czas trwania zdarzenia wyjściowego w bramce

uogólniającej zależy od czasu trwania zdarzeń lub zdarzenia wejściowego (w przypadku bramki XOR), natomiast w przyczynowej – nie zależy, ale ma miejsce opóźnienie zdarzenia wyjściowego względem wejściowych. W szczególnym przypadku opóźnienie może wynosić zero

2.3.3. Bramka uogólniająca XOR



Rys. 4. Bramka uogólniająca XOR

Bramka przedstawiona na rys.4 ma dwa zdarzenia wejściowe: x i y oraz jedno zdarzenie wyjściowe: z. Formalny sposób notacji graficznej bramek został podany w rozdziale 6.3.

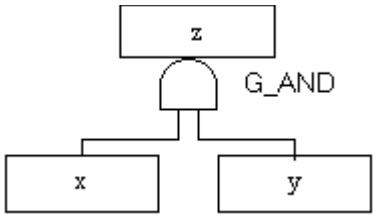
DEFINICJA 1.

$$occur(z) \Rightarrow (occur(x) \wedge x = z) \oplus (occur(y) \wedge y = z) \quad (1)$$

gdzie:

occur(z) - predykat, wystąpienie zdarzenia *z*,
occur(x) - predykat, wystąpienie zdarzenia *x*,
occur(y) - predykat, wystąpienie zdarzenia *y*
$\oplus$ - wykluczające LUB

2.3.4. Bramka uogólniająca AND



Rys. 5. Bramka uogólniająca AND

DEFINICJA 2.

$$occur(z) \Rightarrow occur(x) \wedge occur(y) \wedge overlap(x, y) \wedge \quad (2)$$

$$max(\tau(xs), \tau(ys)) = \tau(zs) \wedge min(\tau(xe), \tau(ye)) = \tau(ze)$$

gdzie:

overlap(x,y) - predykat, zdarzenia *x* i *y* współwystępują razem w czasie,

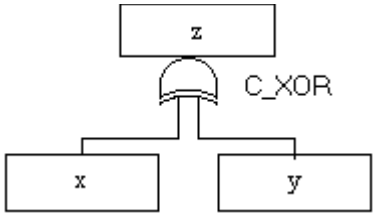
$\tau(xs)$ - moment czasu, w którym rozpoczyna się zdarzenie *x*,

$\tau(ys)$, $\tau(zs)$ - analogicznie do $\tau(xs)$,

$\tau(xe)$ - moment czasu, w którym kończy się zdarzenie *x*,

$\tau(ye)$, $\tau(ze)$ - analogicznie, do $\tau(xe)$.

2.3.5. Bramka przyczynowa XOR



Rys. 6. Bramka przyczynowa XOR

DEFINICJA 3.

$$occur(z) \Rightarrow (\exists x \bullet (duration(x) > t_{d1min} \wedge \tau(xs)+t_{d1min} \leq \tau(zs) \leq \tau(xs)+t_{d1max})) \quad (3)$$
$$\oplus (\exists y \bullet (duration(y) > t_{d2min} \wedge \tau(ys) + t_{d2min} \leq \tau(zs) \leq \tau(ys) + t_{d2max}))$$

gdzie:

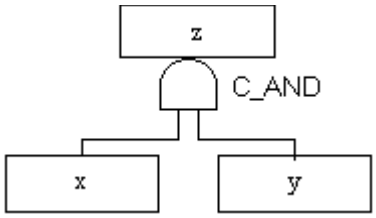
$\bullet$ - spełniające,

duration(x) - czas trwania zdarzenia *x*,

duration(y) - analogicznie, jak *duration(x)*

t_{d1min}, t_{d1max} - odpowiednio, minimalny i maksymalny czas opóźnienia pomiędzy przyczyną (zdarzeniem *x*), a skutkiem (zdarzeniem *z*).

2.3.6. Bramka przyczynowa AND



Rys. 7. Bramka przyczynowa AND

DEFINICJA 4.

$$occur(z) \Rightarrow occur(x) \wedge occur(y) \wedge duration(x \wedge y) \geq t_{dmin} \wedge \quad (4)$$

$$max(\tau(xs), \tau(ys)) + t_{dmin} \leq \tau(hs) \leq max(\tau(xs), \tau(ys)) + t_{dmax}$$

gdzie:

t_{dmin} - minimalny czas współwystępowania zdarzeń *x* i *y*, po którym może wystąpić zdarzenie *z*,

t_{dmax} - maksymalny czas współwystępowania zdarzeń *x* i *y*, po którym na pewno wystąpi zdarzenie *z*.

2.4. Przykład - drzewo niezdatności rozjazdu kolejowego

2.4.1. Założenia

- ruch pociągów na analizowanym fragmencie torów stacji kolejowej odbywa się tylko w kierunku zaznaczonym strzałką na rys.8,
- pociąg na odcinku od punktu A do B jedzie z prędkością nie większą niż 70 km/h i nie mniejszą niż 40 km/h, co wynika z ograniczenia prędkości dla tego odcinka (jeśli pociąg w punkcie A miałby prędkość większą niż 70 km/h, to przyjmujemy, że uruchomiłby automatyczne hamowanie i został zatrzymany); założenie to przyjmiemy dla czytelności przykładu,
- hamulce pociągu działają z maksymalnym opóźnieniem 1000km/h^2,
- do stacji mogą dojechać maksymalnie dwa pociągi; dopiero po odjechaniu jednego z nich może być skierowany kolejny pociąg do rozważanej stacji,
- nadjeżdżający pociąg musi wjechać na wolny tor (by nie doszło do sytuacji niebezpiecznej),
- jeśli kontroler jest sprawny, to "wie", który tor jest zajęty,

- semafor świetlny może sygnalizować jeden z dwóch stanów: otwarty (światło zielone) lub zamknięty (światło czerwone), brak sygnału świetlnego interpretujemy jako światło czerwone.

2.4.2. Opis systemu

Zadaniem systemu jest odpowiednie ustawienie zwrotnicy dla nadjeżdżającego pociągu. Informacja o zbliżającym się pociągu jest przekazywana poprzez sygnał dźwiękowy z lokomotywy oraz sygnał z czujnika A. Ze względów bezpieczeństwa zwrotnica może być ustawiona przez dwa "systemy": zawiadowcę stacji i kontroler.

USTAWIANIE MECHANICZNE ZWROTNICY - KONTROLER

Gdy początek pociągu minie punkt A, wysyłany jest sygnał do Kontrolera. Kontroler ustawia zwrotnicę i otwiera semafor.

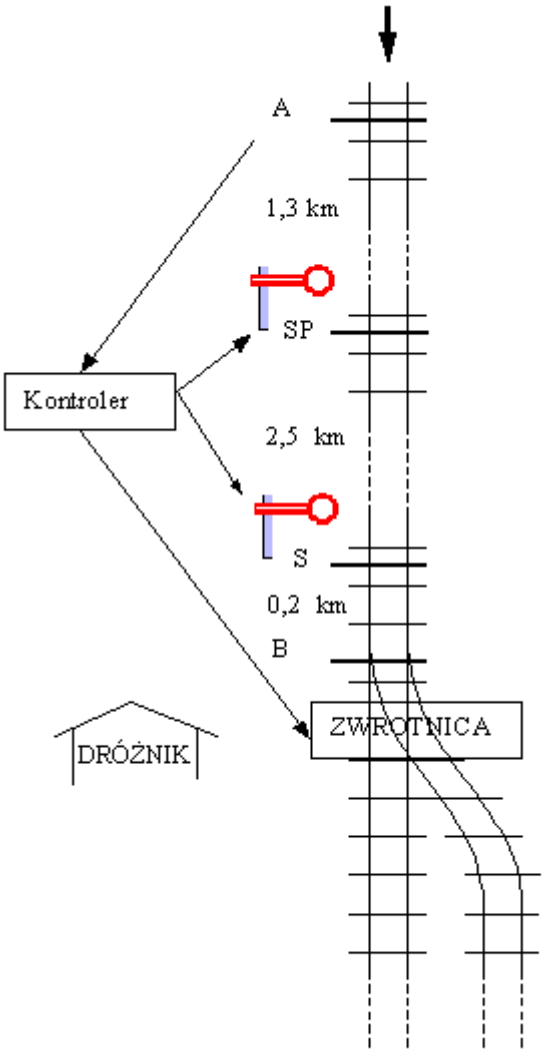
W przypadku, gdy zwrotnica nie zostanie prawidłowo ustawiona przez kontroler, ustawia ją dróżnik, a następnie otwiera semafor (naciskając odpowiedni przycisk).

USTAWIANIE "RĘCZNE" - DRÓŻNIK

Pociąg, zbliżając się do punktu A, wysyła przez maksymalnie 10s sygnał dźwiękowy. Dociera on do dróżnika po minimalnie 13s - dla prędkości rozchodzenia się dźwięku w powietrzu w temperaturze - 30^{o}C. Ze względów bezpieczeństwa, przyjmiemy warunki, gdy dźwięk rozchodzi się wolniej.

Słysząc sygnał (minimalnie po 13s, maksymalnie 13s + 10s = 23s), dróżnik ma obowiązek sprawdzić, czy kontroler ustawił właściwie zwrotnicę. Jeśli nie została ona właściwie ustawiona - ustawia ją ręcznie i otwiera semafor (naciskając odpowiedni przycisk). Ustawianie zajmuje mu od 20s do 90s (załóżmy, że takie wartości wyznaczono doświadczalnie - obserwując pracę dróżnika i dodając margines błędu). Łączny czas liczymy od momentu wysłania sygnału dźwiękowego wynosi, zatem: od 33s do 113s.

Semafor pomocniczy (SP) pokazuje to samo, co semafor (S), by maszynista mógł odpowiednio wcześnie rozpocząć hamowanie. Do zatrzymania pociągu jadącego z szybkością 70 km/h i hamowaniu z opóźnieniem 1000 km/h^2 - określonym w założeniach, pociąg potrzebuje odcinka torów o minimalnej długości 2,45 km.



Rys. 8. Schemat rozjazdu

2.4.3. Drzewo błędów

Budowę drzewa niezdatności rozpoczynamy od identyfikacji hazardu. W naszym przypadku, sytuacja niebezpieczna (hazard) występuje wówczas, gdy wskutek złego ustawienia zwrotnicy nadjeżdżający pociąg zostanie skierowany na tor, który może być uprzednio zajęty przez pociąg stojący na stacji. Oczywiście, nie oznacza to jeszcze kolizji (na stacji może nie być drugiego pociągu, może stać akurat na drugim torze lub wjeżdżający pociąg może zdążyć zahamować). Nie mniej jednak możliwe jest najechania jednego pociągu na drugi (taka sytuacja może prowadzić do wypadku).

z1: Hazard (wjazd pociągu na tor, który może być zajęty)

Rys. 9. FT rozjazd kolejowy - krok1

W kolejnym kroku identyfikujemy przyczyny hazardu oraz bramkę, którą są one połączone. Zauważmy, że zdarzenie *z1* występuje jako skutek dwóch zdarzeń. Oznaczmy te zdarzenia jako *z2* i *z3*.

z2 - "pociąg jedzie z punktu S do B". Punkt B zdefiniujmy jako początek zwrotnicy (pierwszy punkt zwrotnicy od strony nadjeżdżającego pociągu). Zdarzenie *z2* - jako zdarzenie trwające od momentu, gdy pierwszy punkt lokomotywy minie semafor (punkt S) do momentu, gdy pierwszy punkt lokomotywy minie punkt B.

z3 - "nie ustawiona zwrotnica". Zdarzenie rozpoczynające się w momencie, gdy pociąg (pierwszy punkt lokomotywy) minie punkt A, a kończące się w momencie, gdy wskutek działań kontrolera lub zawiadowcy zwrotnica przyjmie prawidłową pozycję.

Moment wystąpienia hazardu to moment, gdy pociąg (pierwszy punkt lokomotywy) minie punkt B.

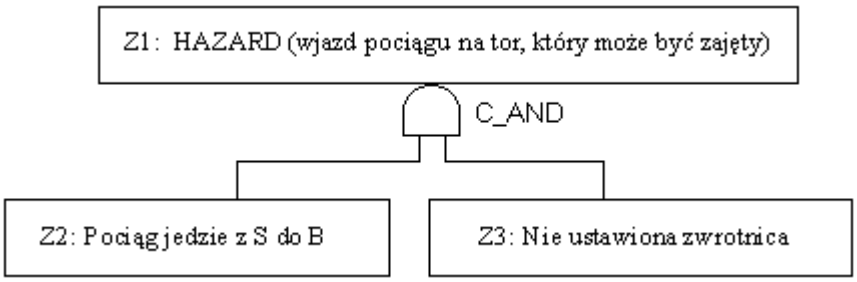
Ponieważ:

- do wystąpienia hazardu muszą wystąpić zdarzenia *z2* i *z3* równocześnie,
- czas trwania hazardu (gdy pociąg znajdzie się na niewłaściwym torze) nie jest już zależny od tego, czy występują zdarzenia *z2* i/lub *z3*,

zatem zdarzenia połączymy bramką przyczynową AND. Graficzną reprezentację zależności pomiędzy zdarzeniami *z2*, *z3* oraz *z1* pokazuje rysunek 7.

W kolejnym kroku identyfikujemy przyczyny zdarzeń *z2* i *z3*, a następnie "przyczyny przyczyn", itd.

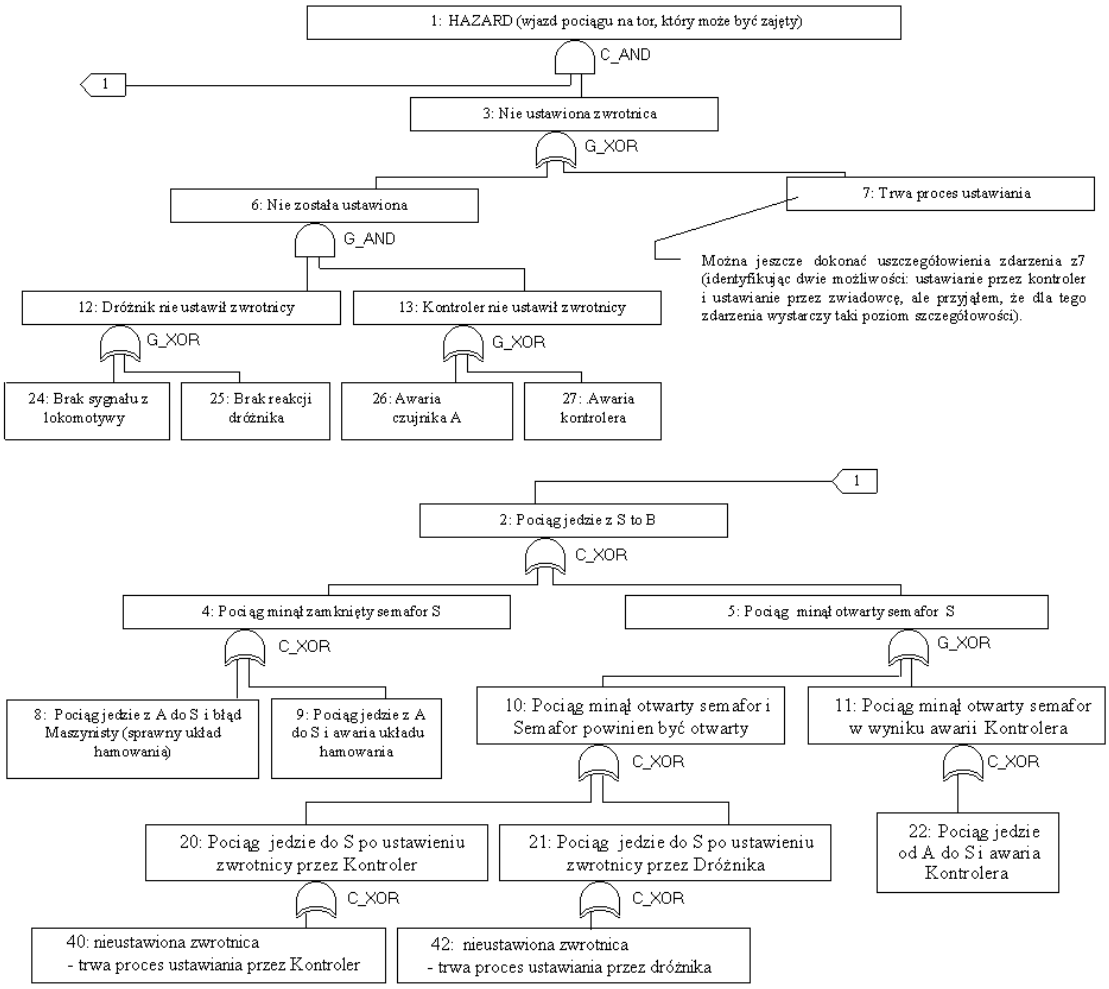
Konstrukcję drzewa kończymy na określonym poziomie szczegółowości lub, jeśli nie potrafimy określić przyczyn danego zdarzenia.



Rys. 10. FT rozjazdu kolejowego - krok2

Dla systemu z rys.8. otrzymamy drzewo błędów przedstawione na rys.11.

Metodę notacji zależności czasowych dla zdarzeń i bramek drzewa niezdatności przedstawię w rozdziale 6.



Rys. 11. Drzewo niezdatności systemu rozjazdu kolejowego

Przedstawiony model rozjazdu stanowi uproszczenie rzeczywistej sytuacji, która ma miejsce na jednotorowych trasach pociągów. Przyjęliśmy założenie (by uczynić przykład czytelniejszym), że pociągi przyjeżdżają z jednego kierunku

– w rzeczywistości nadjeżdżają z dwóch, a stacja jest miejscem ich „mijania”. Liczba torów na stacji również może być większa niż dwa.

Opisaną sytuację możemy spotkać nawet na dużych magistralach kolejowych, choćby jadąc pociągiem relacji Kłodzko – Wrocław. Na pociąg „z przeciwka” często oczekujemy w Ziębicach, gdyż od stacji Kamieniec Ząbkowicki przez Ziębice do stacji Strzelin prowadzi jeden tor.

3. Czasowe sieci Petri'ego

3.1. Definicja czasowej sieci Petri'ego

Definicja czasowej sieci Petri'ego (z ang. *TPN – Time Petri Net*), zgodnie z [BM82], jest następująca:

DEFINICJA 5.

Czasową sieć Petri'ego przedstawiamy jako siódemkę uporządkowaną:

(P, T, B, F, I, M_0, SI),

gdzie:

P (ang. *places*)$=\{p_1,...,p_m\}$ - zbiór miejsc,

T (ang. *transitions*)$=\{t_1, ..., t_n\}$ - zbiór przejść,

$B:PxT\rightarrow N$ funkcja wagi łuku (przypisująca łukowi liczbę naturalną) z miejsca do przejścia, N – zbiór liczb naturalna (ang. *backward transition function*),

$F:TxP\rightarrow N$ funkcja wagi łuku z przejścia do miejsca (ang. *forward transition function*),

$I:PxT\rightarrow\{0,1\}$ jest funkcją łuków hamujących (ang. *inhibitor arcs function*),

$M_0:P\rightarrow N$ jest funkcją znakowania początkowego (ang. *initial marking function*).

(P,T,B,F,I,M_0 -definiują sieć Petri'ego), $SI:T\rightarrow Q_+ \times Q_+\cup\{\infty\}$ jest funkcją przyporządkowującą przejściu statyczny przedział odpaleń (ang. *SI –static interval*), gdzie Q_+ jest zbiorem liczb rzeczywistych nieujemnych.

Zatem, do każdego przejścia t_i zostają przypisane dwie liczby α^S, β^S takie, że:

$SI(t_i) = <\alpha^S, \beta^S>$ oraz $0\le\alpha^S<\infty,\ 0\le\beta^S\le\infty$.

Liczby te określają przedział czasu, w jakim przejście t_i może zostać odpalone (licząc od chwili przygotowania do odpalenia). Zasada odpalania przejść zostanie pokazana w dalszej części pracy. Index: ^{S} oznacza parametry statyczne, a jego brak - dynamiczne.

Zasadnicza różnica pomiędzy parametrami statycznymi oraz dynamicznymi jest taka, że parametry statyczne określają czas odpalenia przejścia wyznaczony na podstawie własności systemu (np. czas otwierania zaworu, czas wykonywania jednej instrukcji w procesorze, czas hamowania samochodu, czas podnoszenia szlabanu). Parametry dynamiczne wyznaczamy podczas analizy TPN.

W celu uniknięcia niejednoznaczności, zdefiniuję parametry dynamiczne w rozumieniu klasycznym (analizy czasowej sieci Petri'ego) oraz w rozumieniu metody INES.

DEFINICJA 6. (definicja klasyczna)

Dynamiczny przedział odpaleń, jest to para liczb $<\alpha_{ti}, \beta_{ti}>$, gdzie:

α_{ti} określa minimalny czas jaki musi upłynąć, by przejście t_i można było odpalić,

β_{ti} określa maksymalny czas jaki może upłynąć do odpalenia przejścia t_i.

Wartości α_{ti} oraz β_{ti} wyznaczane są względem aktualnej chwili czasu..

DEFINICJA 7. (definicja stosowana w analizie INES)

Dynamiczny przedział odpaleń, jest to para liczb $<\alpha_{ti}, \beta_{ti}>$, gdzie:

α_{ti} określa najwcześniejszą chwilę w której przejście t_i może zostać odpalone,

β_{ti} określa najpóźniejszą chwilę w której przejście t_i.może zostać odpalone.

Wartości α_{ti} oraz β_{ti} liczymy względem umownego, ustalonego momentu czasu: "0" (w ramach przyjętego upływu czasu w systemie).

Ponieważ przyjęcie innych oznaczeń literowych nie byłoby celowe ze względu na oznaczenia stosowane w publikacjach oraz ze względu na to iż dotyczą one tej samej własności, w dalszej części pracy wprowadzę indeks ^{k} dla α_{ti}, β_{ti} w rozumieniu Definicji 6.

Zatem:

- $\alpha^k_{ti}, \beta^k_{ti}$ - zgodne z definicją 6,
- α_{ti}, β_{ti} - zgodne z definicją 7.

Zgodnie z powyższym, moment odpalenia dowolnego przejścia t_i mierzony względem umownego, zerowego momentu czasu określa wzór (5).

$$\alpha_{ti} \le \tau(t_i) \le \beta_{ti} \tag{5}$$

gdzie:

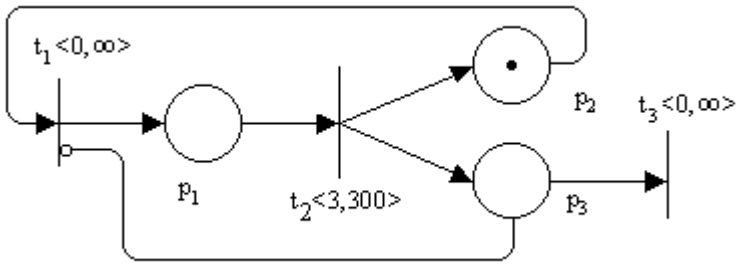
$\tau(t_i)$ - moment odpalenia dowolnego przejścia t_i,

Opis TPN zawierają prace [BD91], [BM82], [Sk96].

Rozważmy bardzo uproszczony przykład: "wykonanie rysunku przez ploter", przy założeniu, że kolejny rysunek możemy rozpocząć dopiero po zabraniu bieżącego z plotera.

Załóżmy, że modelujemy trzy zdarzenia: *p1* -"przesyłanie danych i rysowanie", *p2* -"gotowy rysunek na pulpicie", *p3* -"ploter bezczynny". Do przedstawienia własności, że kolejny rysunek może być wykonany po zabraniu gotowego, posłuży nam łuk hamujący.

TPN modelująca powyższy przykład przedstawia rysunek 12.



Rys. 12. TPN modelująca działanie plotera

$P=\{p_1,p_2,p_3\}$, $T=\{t_1,t_2,t_3\}$, $M_0=\{0,1,0\}$, $SI(t_1)=<0,\infty>$, $SI(t_2)=<3,300>$ $SI(t_3)=<0,\infty>$. Jeśli na łukach nie zostały zaznaczone liczby, oznacza to, że funkcja wagi łuku przyporządkowuje łukowi liczbę 1, w uproszczeniu mówimy, że waga łuku wynosi 1.

Łuk hamujący z miejsca p_3 do przejścia t_1 należy odczytać: dopóki w miejscu p_3 znajduje się znacznik, nie może zostać odpalone przejście t_1. Zasada odpalania przejść zostanie sformalizowana w dalszej części.

Statyczny przedział odpaleń $<0,\infty>$ dla przejść t_1 i t_3 oznacza, odpowiednio:

- początek drukowania może nastąpić w dowolnym momencie czasu, licząc od momentu, gdy ploter jest wolny i poprzednia kartka została zabrana (inaczej, od momentu przygotowania przejścia t_1 do odpalenia)
- kartkę możemy zabrać z plotera w dowolnym momencie po zakończeniu rysowania (od momentu, gdy w miejscu p3 znajdzie się znacznik).

Statyczny przedział odpaleń: $<3,300>$ dla przejścia t_2 oznacza, że ploter jest w stanie wydrukować stronę najszybciej w ciągu trzech jednostek czasu, a najpóźniej -w ciągu 300 jednostek. Czas liczymy od momentu zajścia zdarzenia modelowanego przez miejsce *p1* (od „początku transmisji danych").

Statyczne wartości wyznaczamy doświadczalnie lub na podstawie analizy parametrów fizycznych urządzenia, takich jak np. szybkość transmisji danych, maksymalny (pod względem złożoności i objętości) opis strony, który możemy przesłać, czas rysowania liczony od momentu otrzymania opisu strony, itp. Wyznaczanie parametrów statycznych powinno odbywać się z udziałem ekspertów projektujących lub znających działanie określonego systemu.

3.2. Stany i klasy w TPN

3.2.1. Stany TPN

Stan TPN możemy przedstawić jako parę $S=\{M, I\}$, gdzie M -znakowanie, I – wektor uporządkowanych par liczb nieujemnych, obliczonych dla każdego przygotowanego znakowo do odpalenia przejścia. Przy czym, do odpalenia przygotowane są wszystkie przejścia spełniające zależności określone w definicji 8.

DEFINICJA 8.

Dowolne przejście t_i jest przygotowane do odpalenia wtedy i tylko wtedy, gdy spełniona jest zależność:

$$(\forall p\ (M(p) \geq B(p,t_i))) \ \ and\ ((\forall p\ (I(p,t_i)=1)) \Rightarrow (M(p)=0)) \qquad (6)$$

gdzie: $t_i \in T$

$i = 1, ..., n$

Zatem dowolne przejście t_i jest przygotowane do odpalenia, jeśli w każdym miejscu z którego prowadzi łuk do przejścia t_i jest co najmniej taka liczba znaczników jak waga łuku oraz w każdym miejscu z którego prowadzi łuk hamujący do przejścia t_i nie ma znaczników.

DEFINICJA 9.

Stanem początkowym S_0 TPN nazywamy stan określony dla znakowania początkowego, zatem $S_0=(M_0, I_0)$.

Dla przykładu z rysunku 12 otrzymamy:

$$S_0=(M_0=\{0,1,0\},I=\{\{\alpha^k_{t1}, \beta^k_{t1}\}\})$$

gdzie: α^k_{t1} - najwcześniejszy czas po jakim może zostać odpalone przejście t_1,

β^k_{t1} - najpóźniejszy czas, po jakim przejście t_1 musi zostać odpalone.

Zatem:

$$S_0=(M_0=\{0,1,0\},I=\{\{0,\infty\}\})$$

Czasy α^k_{tl} oraz β^k_{tl} liczymy od momentu przygotowania przejścia t_l do odpalenia. Wyznaczamy je na podstawie wartości statycznych, jeśli przejście zostało przygotowane do odpalenia: $\alpha^k_{tl}=\alpha^S_{tl}$, $\beta^k_{tl}=\beta^S_{tl}$ lub wyliczamy, korzystając z wartości danych dla stanu poprzedniego, co zostanie pokazane w sekcji 3.2.2.

Przedział $<\alpha^k_{tl}, \beta^k_{tl}>$, zgodnie z tym, co zostało podane w sekcji 3.1., nazywamy dynamicznym przedziałem odpaleń.

3.2.2. Przejścia pomiędzy stanami - zasada odpalania przejść

DEFINICJA 10.

Niech $S=\{M, I\}$ będzie dowolnym stanem TPN.

Przejście t_i ($t_i \in T$) może zostać odpalone wtedy i tylko wtedy, gdy:

1) jest przygotowane do odpalenia ze względu na znakowanie M (spełnione są zależności określone wzorem (6)),

2) jest przygotowane do odpalenia ze względu na zależności czasowe pomiędzy nim, a innymi przejściami przygotowanymi do odpalenia ze względu na znakowanie M, zatem jeśli spełniony jest warunek określony wzorem (7):

$$\alpha^k_{ti} \le \theta_{ti} \le min\{\beta^k_{tk}\} \tag{7}$$

gdzie: α^k_{ti} -minimalny czas po jakim przejście t_i może zostać odpalone, licząc od chwili (momentu czasu) przejścia TPN do stanu S.

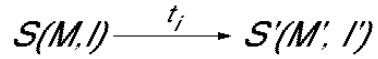
θ_{ti} - względny czas odpalenia przejścia t_i, licząc od chwili przejścia TPN do stanu S,

t_k – dowolne przejście przygotowane do odpalenia ze względu na znakowanie ($t_k \in T$),

β^k_{tk} – najpóźniejszy moment czasu w którym dowolne przejście t_k przygotowane ze względu na znakowanie M do odpalenia, może zostać odpalone.

W dalszej części pracy, przejściami przygotowanymi do odpalenia będę określał przejścia przygotowane ze względu na znakowanie oraz czas.

Niech $S=\{M, I\}$ będzie dowolnym stanem TPN, a przejście t_i dowolnym przejściem przygotowanym do odpalenia. Po odpaleniu przejścia t_i otrzymamy nowy stan S', co graficznie pokazuje rysunek 13.



Rys. 13. Przejście ze stanu S do S' w skutek odpalenia przejścia t_i

Wyliczenie nowego stanu S' odbywa się w dwóch krokach:

K1: wyznaczenie nowego znakowania M':

$$\forall(p)\ M'(p)=M(p)-B(p,t_i)+F(t_i,p) \tag{8}$$

K2: wyznaczenie nowego wektora I':

K2a: $I'=I$,

K2b: usunięcie z wektora I' par liczb związanych z przejściami, które stały się nieprzygotowane do odpalenia przy znakowaniu: $M(p)-B(p,t_i)$,

K2b: przesunięcie o czas θ wartości α^k, β^k dla każdej pary liczb w wektorze I',

K2c: dodanie do wektora I' par liczb związanych z przejściami, które stały się przygotowane do odpalenia dla znakowania M'; dla każdego nowo przygotowanego do odpalenia przejścia t_k przyjmujemy: $\alpha^k_{tk}=\alpha^S_{tk}$, $\beta^k_{tk}=\beta^S_{tk.}$

Dokładny opis zasady odpaleń przejść TPN można znaleźć w [BM82], [BD91], [Sk96].

3.2.3. Opis TPN przy pomocy klas

Ponieważ dziedzina czasu jest ciągła, liczba możliwych stanów TPN jest, poza szczególnymi przypadkami, nieskończona. Dla sieci z rysunku 12, mając stan $S_0=(M_0=\{0,1,0\},I=\{\{\alpha^k_{t2}=0,\ \beta^k_{t2}=\infty\}\})$ przejście t_2 może zostać odpalone dla np. $\theta=0;$ $...;$ $\theta=0{,}001;$ $...;$ $\theta=2{,}3;$ $.....$

W celu osiągnięcia skończonej reprezentacji TPN stosujemy do jej opisu *klasy stanów*.

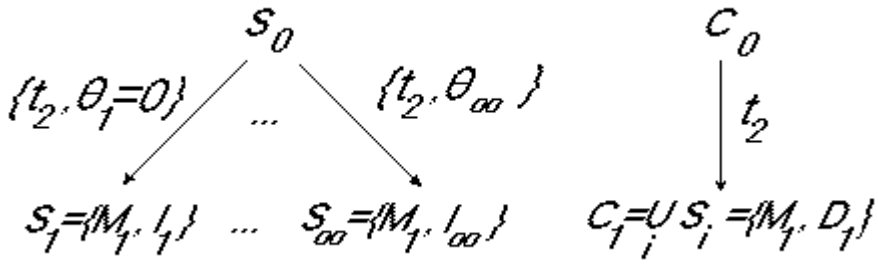
Opis TPN przy pomocy klas możemy znaleźć w [BM82]. Klasę C przedstawiamy jako parę $C=\{M, D\}$, gdzie M jest znakowaniem TPN, a D jest dziedziną odpaleń (ang. *firing domain*). Dziedzina odpaleń jest zbiorem nierówności, po jednej dla każdego przygotowanego do odpalenia przejścia oraz po jednej dla każdej pary

przygotowanych do odpalenia przejść. Nierówności określają przedziały czasu, w którym dane przejście może zostać odpalone oraz zależności pomiędzy chwilami każdej pary przygotowanych do odpalenia przejść. Analizę takiej sieci rozpoczynamy od wyznaczenia klasy początkowej $C_0=\{M_0, D_0\}$.

Dla sieci z rysunku 12 otrzymamy:

$C_0=\{M_0=\{0,1,0\}\ D_0=\{0\leq \tau(t_2)\leq\infty\}\}$, gdzie $\tau(t_2)$ - moment odpalenia przejścia t_2

Przejście z opisu TPN przy pomocy *stanów* do opisu przy pomocy *klas stanów* dla TPN z rysunku 12 pokazuje rysunek 14.



Rys. 14. Przejście z opisu przy pomocy stanów do opisu przy pomocy klas stanów

Jak to zostało pokazane na rysunku 9.3, klasa obejmuje wszystkie stany jakie mogą zostać wygenerowane po odpaleniu tego samego przejścia z pewnego stanu dla wszystkich możliwych względnych czasów odpaleń θ_i.

Kanoniczne postać systemu nierówności zawartego w domenie D jest następująca [BM82]:

$$\begin{cases} \alpha^k_i \leq \tau(t_i) \leq \beta^k_i \text{ dla każdego przygotowanego do odpalenia } t_i \\ \tau(t_i) - \tau(t_j) \leq \gamma_{i,j} \text{ dla każdej pary przygotowanych do odpalenia przejść} \\ t_i, t_j,\ i \neq j \end{cases} \quad (9)$$

Wartości α^k_i, β^k_i określają dynamiczny przedział odpaleń, natomiast wartości $\gamma_{i,j}$ opisują zależności czasowe pomiędzy momentami odpalania przejść.

3.2.4. Wyznaczanie klasy C_0

Zasada odpalania przejść została opisana w [BM82, BD91] oraz [Sk96], przy czym dwie pierwsze pozycje nie obejmują modelu zawierającego łuki hamujące.

ETAP I. Wyznaczenie klasy początkowej $C_0=\{M_0, D_0\}$

Znakowanie początkowe M_0 przyjmujemy jak dla zwykłej sieci Petri'ego, natomiast domenę D_0 wyznaczamy w następujący sposób:

1) dla każdego przygotowanego do odpalenia przejścia t_i, najwcześniejszy (α^k_i) oraz najpóźniejszy (β^k_i) czas odpalenia przejścia wyznaczamy korzystając z parametrów statycznych:

$$\alpha^k_i = \alpha^S_i \qquad (10)$$
$$\beta^k_i = \beta^S_i$$

2) zależności pomiędzy czasami odpaleń przejść przygotowanych do odpalenia $\gamma_{j,k}$ *dla* $j \neq k$, obliczamy jako różnicę:

$$\gamma_{j,k} = \beta^S_j - \alpha^S_k \qquad (11)$$

3.2.5. Zasada odpaleń przejść - wyznaczanie nowej klasy

Niech odpalaną klasą będzie pewna klasa $C=\{M, D\}$. Niech t_f będzie odpalanym przejściem. Po przyjęciu przejścia t_f jako nieprzygotowanego do odpalenia, klasę $C' = \{M', D'\}$ liczymy w trzech krokach:

Krok I: wykonujemy poniższe obliczenia dla każdego przygotowanego do odpalenia przejścia t_i:

$$\alpha^k_i = max\{0, -\gamma_{f,i}, \alpha^k_i - \beta^k_f\} \qquad (12)$$
(OP1)
$$\beta^k_i = min\{\gamma_{i,f}, \beta^k_i - \alpha^k_f\}$$
$$\gamma_{i,j} = min\{\gamma_{i,j}, \beta^k_i - \alpha^k_j\}$$

Krok II: dokonujemy eliminacji przejść, które były przygotowane do odpalenia ze względu na znakowanie, a stały się nieprzygotowane do odpalenia dla znakowania M″ - wyznaczonego po odpaleniu przejścia t_f.

W tym celu liczymy znakowanie M'':

$$(\forall p) M''(p) = M(p) - B(p, t_f) \qquad (13)$$

Następnie, dla każdego przejścia, które było przygotowane do odpalenia ze względu na znakowanie, a stało się nieprzygotowane, dokonujemy następującej operacji:

Niech t_e odpowiada eliminowanemu przejściu, zatem:

$$\alpha^k_i = max\{\alpha^k_i, \alpha^k_i - \gamma_{e,i}\} \tag{14}$$

(OP2)
$$\beta^k_i = min\{\beta^k_i, \beta^k_i + \gamma_{i,e}\}$$
$$\gamma_{j,k} = min\{\gamma_{j,k}, \gamma_{j,e} + \gamma_{e,k}\}$$

Krok III: dodanie do systemu nierówności odpowiadających nowo przygotowanym do odpalenia przejściom. W tym celu liczymy znakowanie:

$$(\forall p) M'(p) = M''(p) + F(t_f, p) \tag{15}$$

Następnie, przyjmując t_n - nowo przygotowane do odpalenia przejście ze względu na znakowanie wykonujemy:

(OP3)

dla wszystkich zmiennych nie związanych z $\tau(t_n)$ *nie wprowadzamy zmian* (16)

dla nowego przejścia t_n *przyjmujemy:* $\alpha^k_n = \alpha^S_n$, $\beta^k_n = \beta^S_n$

zależności pomiędzy momentem odpalenia przejścia t_n *i momentami odpaleń pozostałych przejść:*

$\tau(t_n) - \tau(t_k) \le \gamma_{n.k}$ *dla każdego* $k \neq n$

$\tau(t_j) - \tau(t_n) \le \gamma_{j,n}$ *dla każdego* $j \neq n$

wartości $\gamma_{n.k}$ *oraz* $\gamma_{j,n}$ *wyliczamy z równań:*

$\gamma_{n.k} = \beta^S_n - \alpha^k_k$

$\gamma_{j,n} = \beta^k_j - \alpha^S_n$

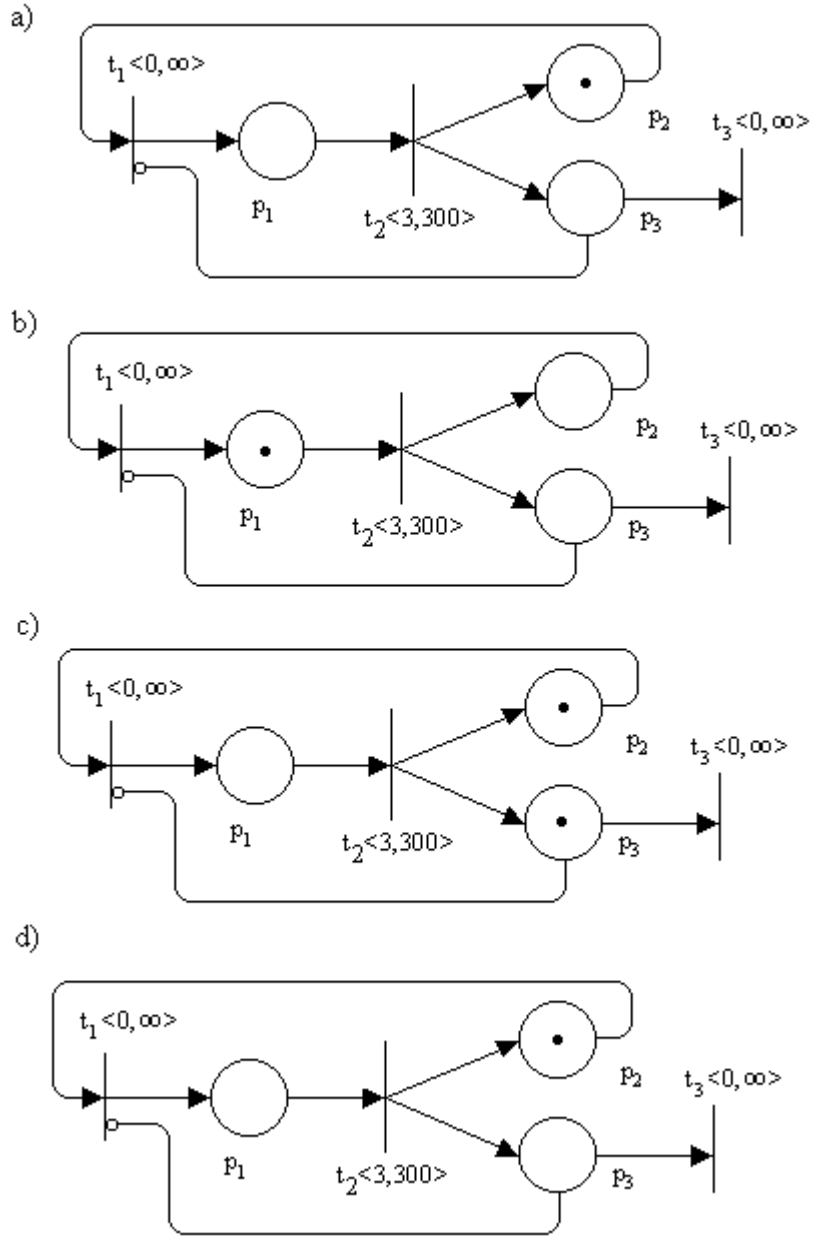
W wyniku powyższych operacji: (OP1), (OP2) oraz (OP3) i wzorów (13), (15) wyznaczamy, na podstawie pewnej klasy C, klasę C'. Zatem, mając daną z definicji klasę C_0 możemy wyznaczyć pozostałe klasy TPN.

Zachowanie TPN możemy opisać poprzez podanie klas dla danej TPN oraz sekwencji odpaleń pomiędzy klasami. Analogicznie do opisu za pomocą stanów, używając zasady odpaleń, możemy zbudować drzewo klas (korzeń -klasa inicjująca C_0), w którym łuki są etykietowane przejściami. Z danej klasy wychodzi tyle łuków, ile odpalanych przejść (przykład w kolejnym rozdziale).

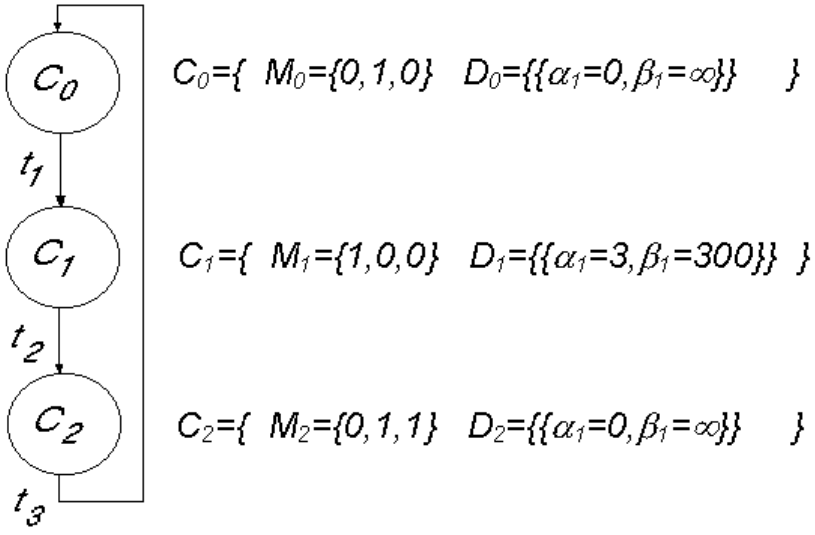
3.3. Analiza dynamiczna TPN

Ponieważ TPN nie jest przedmiotem tego opracowania, zilustruję jedynie zasadę generowania przestrzeni stanów na przykładzie TPN z rys. 12.

Zasadę odpalania przejść oraz zmianę znakowania pokazuje rysunek 15, natomiast diagram klas - rysunek 16.



Rys. 15. TPN -zasada odpalania przejść a) ploter gotowy do pracy, b) trwa proces rysowania, c) zakończono rysowanie i kartka znajduje się w podajniku (łuk hamujący z miejsca p_3 do przejścia t_1 modeluje tą własność, iż nie można rozpocząć wydruku kolejnej kartki, dopóki nie zostanie zabrana bieżąca kartka z podajnika), d) zabrano kartkę (ploter gotowy do kolejnego wydruku)



Rys. 16. Diagram klas dla TPN z rysunku 12.

Mając daną klasę C_0 wyznaczamy wszystkie pozostałe klasy, przy czym może zdarzyć się sytuacja, że liczba klas dla pewnej sieci Petri'ego będzie nieskończona. W pracy [BM82] zostały omówione takie przypadki, lecz dla TPN modelujących drzewa niezdatności, ze względu na przyjęte modele bram, takiego niebezpieczeństwa nie ma.

Podczas analizy sprawdzamy również, czy wyznaczona klasa jest równa jednej z poprzednio otrzymanych. Jak pokazuje rysunek 16, system znajdując się w stanie opisanym przez klasę C_2, po zajściu zdarzenia "zabrano kartkę" (odpaleniu przejścia t_3), przejdzie do stanu opisanego przez klasę C_0.

Algorytm analizy TPN modelującej FT z zależnościami czasowymi opisałem w [Sk96].

4. Zastosowanie czasowych sieci Petri'ego do modelowania oraz analizy drzew niezdatności

4.1. Wprowadzenie

Zastosowanie czasowych sieci Petri'ego (TPNs - *ang. Time Petri Nets*) do modelowania oraz analizy drzew niezdatności polega na zastąpienia drzewa niezdatności (FT - *ang. Fault Tree*), równoważną siecią Petri'ego. Konstrukcja takiej sieci polega na zastąpieniu bramek (wraz ze zdarzeniami wejściowymi oraz zdarzeniem wyjściowym) w FT odpowiadającymi im strukturami TPN. Metoda ta została przedstawiona w [GMW95]. Podczas analizy TPN modelującej FT generowana jest jednak bardzo duża liczba klas, co ogranicza zastosowanie klasycznej analizy do niewielkich drzew niezdatności [Sk96], [W96] lub do analizy wybranych fragmentów drzewa niezdatności, w tym również wprowadzonych zabezpieczeń [Sk97].

Ponieważ metoda transformacji FT w TPN stanowi podstawę do opracowania metody INES (*ang. INequalities - Equalities System*), zostanie ona omówiona w tym rozdziale.

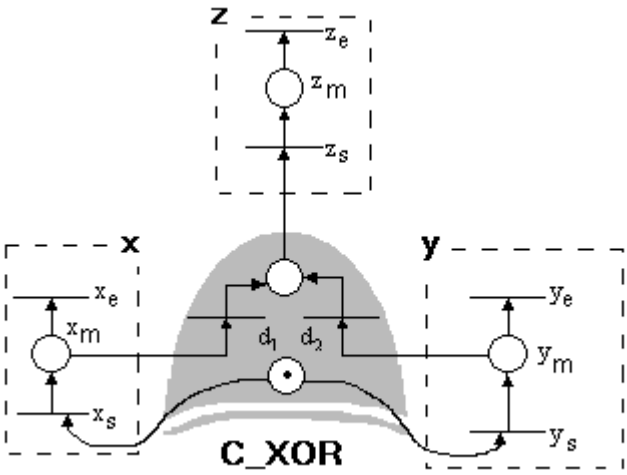
4.2. Model bramki przyczynowej XOR (ang. Causal XOR)

W metodzie zaprezentowanej w [GMW96] przyjęto poniższą zasadę modelowania zdarzeń:

- miejsca odpowiadają zdarzeniom,
- start oraz koniec zdarzenia reprezentowany jest przez przejście (występowanie znacznika w miejscu związanym z jakimś zdarzeniem oznacza jego „trwanie").

Rysunek 17 przedstawia model TPN dla bramki przyczynowej XOR. Na rysunku widoczne są trzy fragmenty sieci modelujące zdarzenia x, y, z (obszary zaznaczone linią przerywaną) oraz fragment TPN modelujący bramę (na szarym tle bramki).

Przejścia x_s, y_s, z_s reprezentują odpowiednio: strat zdarzenia x, start zdarzenia y, start zdarzenia z. Przejścia x_e, y_e, z_e - reprezentują zakończenia zdarzeń. Jeśli w miejscu x_m znajduje się znacznik to oznacza, iż w ramach przyjętego upływu czasu zdarzenie x trwa. Analogicznie, jeśli znacznik znajdzie się w miejscu y_m lub z_m oznacza to trwanie zdarzeń y, z.



Rys. 17. TPN modelująca bramę przyczynową XOR

Przejście d_1 reprezentuje opóźnienie, po jakim może wystąpić start zdarzenia *z* licząc od startu zdarzenia *x*, czyli - po jakim czasie od wystąpienia przyczyny może wystąpić skutek.

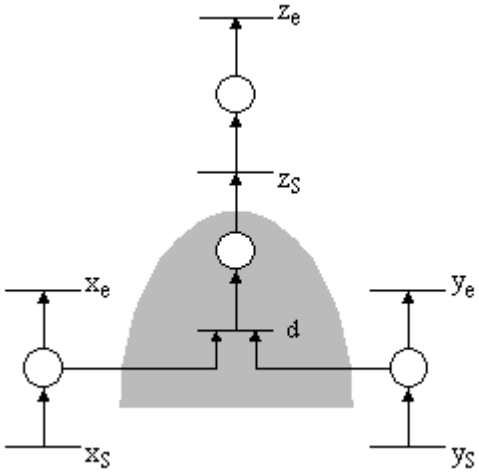
Analogicznie, d_2 reprezentuje opóźnienie pomiędzy zdarzeniem *y* oraz *z*.

Dla bramki przyczynowej XOR określamy następujące parametry czasowe:

- $SI(x_S) = \langle 0,0 \rangle$, $SI(y_S) = \langle 0,0 \rangle$, $SI(z_S) = \langle 0,0 \rangle$ gdzie *SI* - statyczny przedział odpaleń (patrz rozdział 3.)
- $SI(x_e) = \langle \alpha^S_{xe}, \beta^S_{xe} \rangle$, $SI(y_e) = \langle \alpha^S_{ye}, \beta^S_{ye} \rangle$, $SI(z_e) = \langle \alpha^S_{ze}, \beta^S_{ze} \rangle$, przy czym przedziały czasowe specyfikują czas trwania zdarzeń *x, y, z*
- $SI(d_1) = \langle \alpha^S_{d1}, \beta^S_{d1} \rangle$, $SI(d_2) = \langle \alpha^S_{d2}, \beta^S_{d2} \rangle$. Gdzie: α^S_{d1} – najwcześniejszy czas, po którym może wystąpić skutek (start zdarzenie *z*) licząc od wystąpienia przyczyny (startu zdarzenia *x*); β^S_{d2} - najpóźniejszy czas, po którym na pewno wystąpi skutek (start zdarzenie *z*), jeśli będzie nadal występowała przyczyna (zdarzenie *x*) licząc od startu przyczyny. W ten sam sposób interpretujemy α^S_{d2}, β^S_{d2} dla przyczyny *y*.

4.3. Model bramki przyczynowej AND

Model bramki przyczynowej AND (*ang. Causal AND*) przedstawia rysunek 18.



Rys. 18. TPN modelująca bramkę przyczynową AND

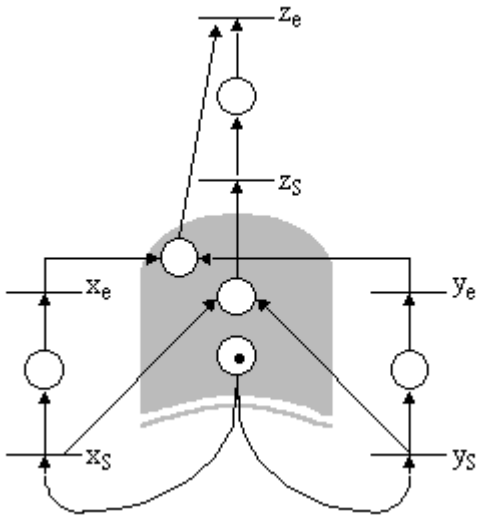
Znaczenie oznaczeń z rysunku 18. jest następujące:

- x_s i x_e – przejścia reprezentujące odpowiednio start i koniec zdarzenia x (analogicznie y_s, y_e, z_s, z_e dla zdarzeń y i z),
- $SI(x_S) = \langle 0,0 \rangle$, $SI(y_S) = \langle 0,0 \rangle$, $SI(z_S) = \langle 0,0 \rangle$ - przedziały czasu przypisane do przejść modelujących start zdarzeń, odpowiednio, x, y, z; wartości $\langle 0,0 \rangle$ oznaczają, że przejścia te zostaną odpalone natychmiast po przygotowaniu do odpalenia, (*SI*-statyczny przedział odpaleń)
- $SI(x_e) = \langle t_{x\ min}, t_{x\ max} \rangle$, $SI(y_e) = \langle t_{y\ min}, t_{y\ max} \rangle$, $SI(z_e) = \langle t_{z\ min}, t_{z\ max} \rangle$, - przedziały czasu przypisane do przejść modelujących zakończenie zdarzenia; wartości $t_{x\ min}$, $t_{x\ max}$, $t_{y\ min}$, $t_{y\ max}$, $t_{z\ min}$, $t_{z\ max}$ specyfikują odpowiednio czasy trwania zdarzeń x, y, z.
- d jest przejściem reprezentującym opóźnienie czasowe pomiędzy wystąpieniem przyczyn a skutkiem, który mogą spowodować,
- $SI(d) = \langle t_{d\ min}, t_{d\ max} \rangle$ -przedział czasu, określający jak długo muszą współwystępować przyczyny, by wywołać skutek.

.

4.4. Model bramki uogólniającej XOR

Model bramki uogólniającej XOR (*ang. Generalization XOR*) przedstawia rysunek 19.



Rys. 19. TPN modelująca bramkę uogólniającą XOR

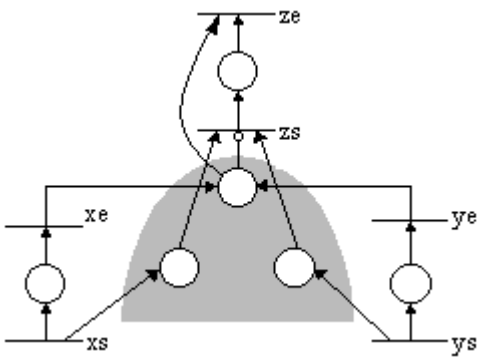
Przedziały czasu przypisane do przejść z rys.19. są następujące:

- $SI(x_S) = \langle 0,0 \rangle$, $SI(y_S) = \langle 0,0 \rangle$, $SI(z_S) = \langle 0,0 \rangle$
- $SI(x_e) = \langle t_{x\,min}, t_{x\,max} \rangle$, $SI(y_e) = \langle t_{y\,min}, t_{y\,max} \rangle$, $SI(z_e) = \langle 0,0 \rangle$

Oznaczeń na rysunku 19 są takie same jak w poprzednich modelach bram.

4.5. Model bramki uogólniającej AND

Model bramki uogólniającej AND (*ang. Generalization AND*) przedstawia rysunek 20.



Rys. 20. TPN modelująca bramkę uogólniającą AND

Przedziały czasu przypisane do przejść z rys.20., są następujące:

- $SI(x_S) = \langle 0,0 \rangle$, $SI(y_S) = \langle 0,0 \rangle$, $SI(z_S) = \langle 0,0 \rangle$,
- $SI(x_e) = \langle t_{x\,min}, t_{x\,max} \rangle$, $SI(y_e) = \langle t_{y\,min}, t_{y\,max} \rangle$, $SI(z_e) = \langle 0,0 \rangle$

Jeśli w miejscu, z którego wychodzi łuk hamujący, znajduje się znacznik to uniemożliwia on odpalenie przejścia z_s. Łuk ten modeluje następującą własność: „jeśli zakończy się zdarzenie *x* lub *y* przed rozpoczęciem zdarzenia *z*, to zdarzenie *z* nie wystąpi".

4.6. Modelowanie drzewa niezdatności przy pomocy czasowej sieci Petri'ego

Ze względu na opracowane modele, metoda ta może być obecnie stosowana do FT zbudowanych z wykorzystaniem dwuwejściowych bram: uogólniającej AND, przyczynowej AND, uogólniającej XOR, przyczynowej OR [GMW96], uogólniającej OR [W96] oraz przyczynowej XOR (sekcja 4.2). Możliwe jest jednak modelowanie wielowejściowych bramek z wykorzystaniem istniejących modeli.

Jeśli chcemy zbudować TPN modelującą FT, możemy to uczynić od razu lub zbudować FT, a następnie przekształcić je w TPN. Ponieważ technika analizy drzew niezdatności jest często stosowana, a konstrukcja drzewa prostsza od budowy TPN, drugie rozwiązanie jest wygodniejsze. W związku z tym, bardzo praktycznym jest, wspomaganie pracy człowieka przez komputer. Pożądany proces analizy przebiega według schematu:

1. Człowiek buduje drzewo niezdatności. Sam proces tworzenia wymaga wiedzy
   i doświadczenia ekspertów. Do tej pory nie powstały ogólnodostępne bazy wiedzy, czy systemy ekspertowe, z wykorzystaniem których byłoby możliwe np. przeprowadzenie automatycznej konstrukcji drzewa niezdatności dla prostych systemów określonego typu. Nie ma też banków informacji
   z drzewami niezdatności dla elektrowni atomowych, samolotów itp. W literaturze możemy spotkać jedynie proste przykłady lub fragmenty dużych drzew niezdatności. Sytuacja taka, jak można wnioskować, nie zmieni się

w najbliższym czasie. Drzewo niezdatności samo w sobie niesie zbyt wiele informacji dla konkurencji firmy, a także niepowołanych osób (np. terrorystów), by mogło być powszechnie udostępniane.

Co do procesu konstrukcji drzewa niezdatności, to może być on wspomagany „komputerem", np. [Relex].

W prezentowanej metodzie, w odróżnieniu od klasycznej analizy, muszą zostać wyznaczone również parametry statyczne, określające czas trwania zdarzeń w drzewie niezdatności.

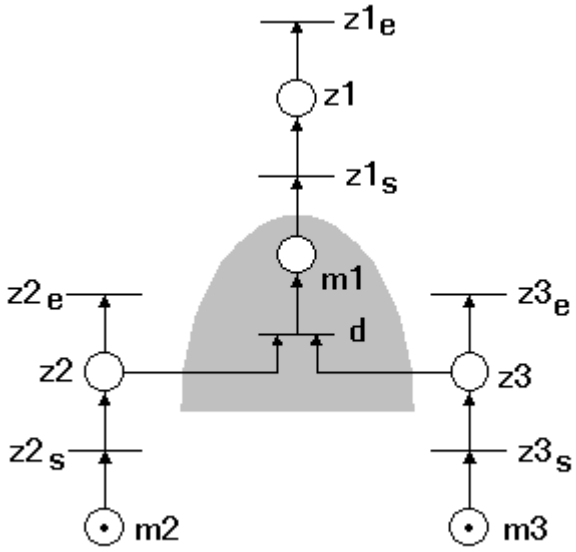
2. Komputer przeprowadza analizę i generuje wyniki, [Sk96][2].
3. Człowiek interpretuje wyniki. Jeśli stwierdzimy, że w rozważanym systemie,

   z wyznaczonymi parametrami czasu, hazard nie wystąpi, to kończymy analizę.

   W przeciwnym razie, mając na uwadze wyniki analizy, wprowadzamy np. zabezpieczenia i/lub zmieniamy konstrukcję systemu i/lub stosujemy precyzyjniejsze urządzenia. Po tych operacjach, przechodzimy do kroku 1 – nasze zmiany będą wymagały, co najmniej, modyfikacji niektórych parametrów czasowych, a mogą także powodować konieczność modyfikacji struktury FT.

Przyjmijmy, że w kroku 1 zdefiniowaliśmy fragment drzewa niezdatności, który poddamy analizie. Niech będzie to FT z rys. 10. Równoważny model TPN przedstawia rysunek 21.

[2] Nie spotkałem jednak do tej pory narzędzia, które dokonuje transformacji FT w TPN. Program dołączony do pracy [Sk96] również wymaga podania danych wejściowych w postaci TPN.
Podanie opisu FT i uzyskanie wyników w postaci minimalnych zbiorów przyczyn rozszerzonych o czas, można spotkać jednak w narzędziu [Jarmuż96] oraz w narzędziu powstałym w oparciu o niniejszą pracę, ale nie jest to, sensu stricte, analiza TPN.



Rys. 21. TPN modelująca FT z rysunku 10.

Określmy teraz parametry czasowe dla TPN z rysunku 21.

Ze względu na przyjęte ograniczenia (sekcji 2.4.2) co do maksymalnej prędkości pociągu, minimalny czas trwania zdarzenia *z2* (pociąg jedzie z S do B) wynosi 10 sekund, natomiast maksymalny: ∞ (pociąg może nigdy nie dojechać, zepsuć się i zatrzymać). W praktyce zamiast ∞, należy przyjąć odpowiednio długi czas.

Może to być np. liczba sekund odpowiadająca dobie (zakładając, że zgodnie z obowiązującymi procedurami oraz możliwościami technicznymi, służby porządkowe muszą w ciągu doby usunąć pociąg w przypadku awarii lub wykolejenia).

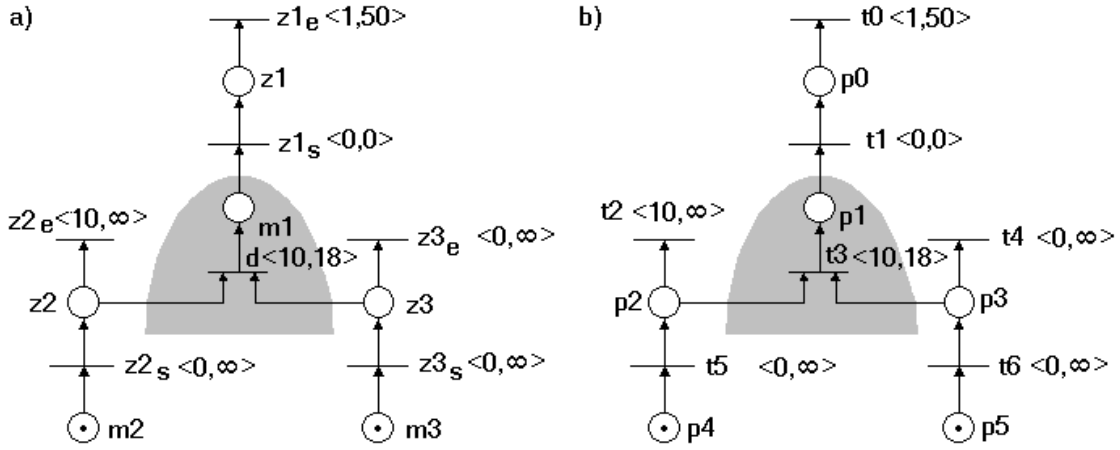
Minimalny czas ustawiania zwrotnicy wynosi: 0 (w sytuacji, gdy np. była ustawiona wcześniej), natomiast maksymalny: ∞ (w przypadku awarii), a zatem zdarzenie *z3* (nieustawiona zwrotnica) może trwać najkrócej 0, a najdłużej ∞.

Pozostaje jeszcze określenie minimalnego i maksymalnego opóźnienia pomiędzy wystąpieniem zdarzeń *z2* oraz *z3*, a wystąpieniem hazardu *z1*.

Ponieważ hazardem jest sytuacja, gdy pociąg wjeżdża na tor przy nie ustawionej zwrotnicy (wówczas mogą się znaleźć dwa pociągi na jednym torze), sytuacja taka najwcześniej nastąpi po 10 sekundach, natomiast nie później, jeśli pociąg jedzie zgodnie

z założeniami (powyżej 40 km/h), niż po 18s. Można oczywiści przyjąć, że po ∞, ale ze względów na precyzyjniejszą analizę przyjmiemy mniejszy czas.

TPN z zaznaczonymi zależnościami czasowymi pokazuje rys. 22. Przedział czasu: <0,∞> dla przejść $z2_S$ oraz $z3_S$ oznacza, że zdarzenia te mogą wystąpić w dowolnym momencie czasu.



Rys. 22. TPN z zaznaczonymi statycznymi parametrami czasowymi, modelująca FT z rysunku 10 a) model z zaznaczonymi zdarzeniami, b) model z ponumerowanymi miejscami i przejściami

W celu przejrzystości analizy oraz zgodności z wynikami wygenerowanymi w zastosowanym narzędziu, przyjęto skróty literowe: *p* -miejsce (ang. *place*), *t* –przejście (ang. *transition*).

Wyniki analizy zostały omówione w kolejnym rozdziale.

4.7. Analiza czasowej sieci Petri'ego modelującej drzewo niezdatności

Wyniki analizy przeprowadzone dla TPN z rysunku 22 są następujące:

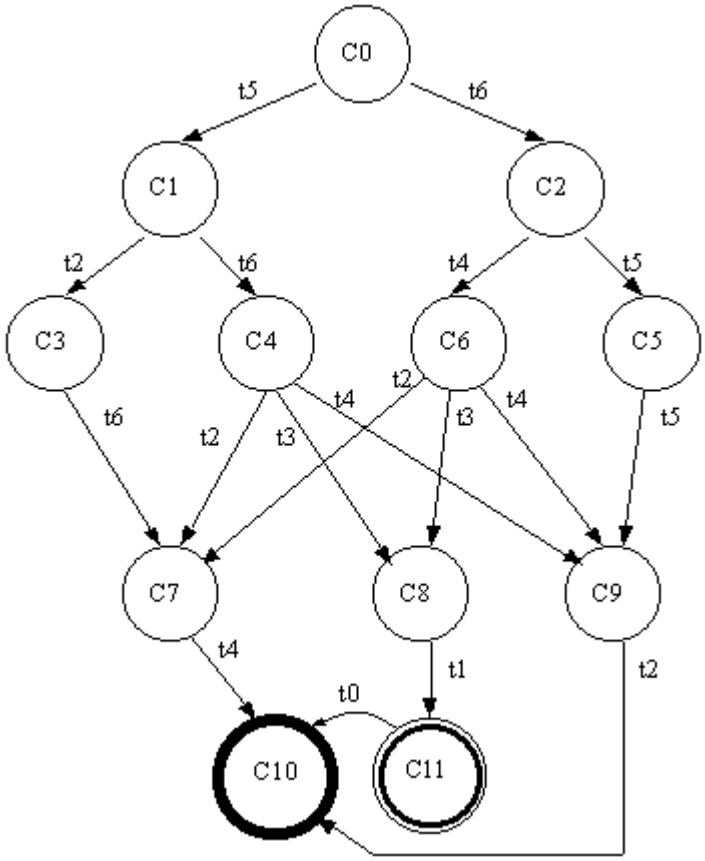
```
KLASA C0
M 0 0 0 0 1 1 *)
0<= τ(t5) <= ∞
0<= τ(t6) <= ∞
γ5,6 = ∞
γ6,5 = ∞

KLASA C1
M 0 0 1 0 0 1

KLASA C3
M 0 0 0 0 0 1
0<= τ(t6) <= ∞

KLASA C4
M 0 0 1 1 0 0
0<= τ(t2) <= ∞
10<= τ(t3) <= 18
0<= τ(t4) <= ∞

KLASA C6
M 0 0 1 1 0 0
10<= τ(t2) <= ∞
10<= τ(t3) <= 18
0<= τ(t4) <= ∞
γ2,3 = ∞
γ2,4 = ∞
γ3,2 = 8
γ3,4 = 18

KLASA C9
M 0 0 1 0 0 0
0<= τ(t2) <= ∞

KLASA C10
M 0 0 0 0 0 0

KLASA C11
M 1 0 0 0 0 0
0<= τ(t0) <= 49
```

```
10<= τ(t2) <= ∞
0<= τ(t6) <= ∞
γ2,6 = ∞
γ6,2 = ∞

KLASA C2
M 0 0 0 1 1 0
0<= τ(t4) <= ∞
0<= τ(t5) <= ∞
γ4,5 = ∞
γ5,4 = ∞

γ2,3 = ∞
γ2,4 = ∞
γ3,2 = 18
γ3,4 = 18
γ4,2 = ∞
γ4,3 = ∞

KLASA C5
M 0 0 0 0 1 0
0<= τ(t5) <= ∞

γ4,2 = ∞
γ4,3 = ∞

KLASA C7
M 0 0 0 1 0 0
0<= τ(t4) <= ∞

KLASA C8
M 0 1 0 0 0 0
0<= τ(t1) <= 0

C0 ->t5/s=0/C1, t6/s=0/C2,
C1 ->t2/s=0/C3, t6/s=0/C4,
C2 ->t4/s=0/C5, t5/s=0/C6,
C3 ->t6/s=0/C7,
C4 ->t2/s=0/C7, t3/s=0/C8, t4/s=0/C9,
C5 ->t5/s=10**)/C9,
C6 ->t2/s=0/C7, t3/s=0/C8, t4/s=0/C9,
C7 ->t4/s=0/C10,
C8 ->t1/s=1/C11,
C9 ->t2/s=0/C10,
C11 ->t0/s=0/C10
```

*) Zapis M 0 0 0 0 1 1 określa liczbę znaczników kolejno w miejscach p0 p1 p2 p3 p4 p5 (w miejscach p4 i p5 jest zatem po jednym znaczniku, a w pozostałych - brak),

**) s (ang. shift)=10 oznacza przesunięcie upływu czasu o 10 jednostek. Takiego przesunięcia, jeśli jest możliwe, dokonuje się zawsze po wyznaczeniu klasy. Pozwala to na zminimalizowanie liczby klas. Wartość przesunięcia czasu jest to wielkość liczona od momentu czasu, w którym klasa została wyznaczona, do momentu, w którym może zostać odpalone jakiekolwiek z przejść przygotowanych ze względu na znakowanie do odpalenia. Patrz [BM82], [Sk96].
Zapis t5/s=10*/C9 należy odczytać: po odpaleniu przejścia t5 i upływie 10 jednostek czasu otrzymamy klasę C9 (w "upływających" 10 jednostkach czasu, żadne z przejść nie jest przygotowane do odpalenia ze względu na czas).

Powyższe wyniki zostały wygenerowane przy pomocy narzędzia dołączonego do opracowania [Sk96], zgodnie ze wzorami OP1, (13), OP2, (15), OP3. Dla większej przejrzystości przejścia pomiędzy klasami pokazuje graf na rysunku 23. Wyniki analizy posłużą w dalszej części opracowania do porównania z analizą metodą INES.



Rys. 23. Diagram klas dla TPN z rysunku 22.

Jak możemy zauważyć, klasa C11 jest związana ze znacznikiem w miejscu, które modeluje występowanie hazardu, zatem hazard może wystąpić w rozważanym systemie. Następnym naszym działaniem jest modyfikacja projektu systemu lub wprowadzenie stosownych zabezpieczeń, by nie mogło dojść do hazardu. Ze względu jednak na to, iż analizie został poddany tylko niewielki fragment drzewa niezdatności, zabezpieczanie systemu zostanie omówione po przeprowadzeniu analizy całego drzewa metodą INES, w załączniku A.

Klasa C10 jest klasą końcową.

4.8. Podsumowanie

Zastosowanie TPN do analizy drzew niezdatności pozwala na przeprowadzenie analizy dynamicznej modelowanego systemu, ukierunkowanej na analizę osiągalności hazardu oraz identyfikację minimalnych zbiorów przyczyn wraz z relacjami czasowymi pomiędzy zdarzeniami.

Ze względu na dużą złożoność obliczeniową oraz dużą liczbę generowanych klas, analiza czasowej sieci Petri'ego modelującej drzewa niezdatności, może być stosowana do analizy:

- drzew niezdatności o niewielkiej ilości bramek i zdarzeń [Sk96], [MS00],
- fragmentów drzew niezdatności z wprowadzonymi zabezpieczenia oraz działania zabezpieczeń w połączeniu ze zdarzeniami drzewa niezdatności [Sk97],
- analizy wybranych ścieżek (fragmentów) drzew niezdatności.

5. Metoda INES (INequalities - Equalities System) analizy drzew niezdatności poszerzonych o zależności czasowe

5.1. Wprowadzenie

Klasyczna analizy TPN modelującej drzewa niezdatności generuje przestrzeń wszystkich możliwych stanów TPN, uściślając -wszystkich zbiorów stanów, zwanych klasami. Z punktu widzenia bezpieczeństwa, nie jest istotna aż tak dokładna informacja.

Analizując zachowanie systemu (pod względem bezpieczeństwa) interesują nas jedynie te stany systemu, które mogą prowadzić do hazardu lub informacja o ich braku. Zatem, przeprowadzając analizę TPN modelującej FT, będziemy szukać tylko tych stanów, które mogą prowadzić do wystąpienia zdarzenia związanego z hazardem. Do takiej analizy została opracowana metoda INES.

Mimo, iż podstawę do opracowania metody INES stanowi TPN modelująca drzewa niezdatności, to korzystanie z samego systemu równań i nierówności nie wymaga znajomości TPN. Jak to zostanie pokazane, z wykorzystaniem TPN został opracowany system równań i nierówności dla każdej z bram, pozwalający na analizę FT

z odpowiednio określonymi parametrami czasowymi dla zdarzeń [MS00].

W metodzie INES przeprowadzamy analizę TPN z "góry na dół". Przyjmujemy pewien umowny moment czasu jako moment 0 (najwygodniej za moment 0 przyjąć start hazardu) i wyznaczamy przedział czasu, który określa występowanie hazardu w ramach przyjętego upływu czasu (w szczególnym przypadku, hazard może trwać nieskończenie długo). Następnie, idąc "w dół" drzewa niezdatności, wyznaczamy czasy trwania dla kolejnych zdarzeń -tylko tych, których występowanie może prowadzić do hazardu.

Zasadnicza różnica pomiędzy analizą klasyczną TPN, a analizą prezentowaną w niniejszym opracowaniu jest taka, że metoda klasyczna wymaga analizy wszystkich lub prawie wszystkich stanów w celu stwierdzenia osiągalności hazardu, natomiast w prezentowanej metodzie -analizujemy tylko te, które mogą prowadzić do hazardu.

W metodzie INES występują dwa rodzaje warunków:

1. Warunki statyczne (ang. *SC - static conditions*), których niespełnienie gwarantuje, że hazard nie może wystąpić.

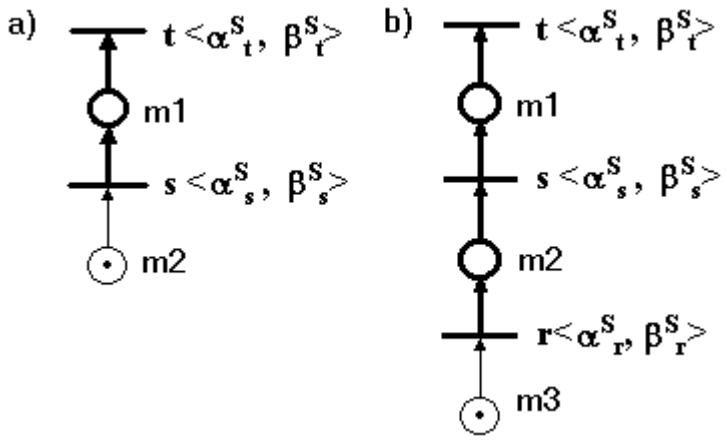
2. Warunki dynamiczne (ang. *DC - dynamic conditions*), określające takie zależności na chwile rozpoczęcia i zakończenia zdarzeń, których spełnienie jest równoważne z możliwością wystąpienia hazardu.

Warunki statyczne są związane z analizą parametrów statycznych systemu, natomiast warunki dynamiczne dotyczą analizy występowania zdarzeń w czasie, co zostanie pokazane.

5.2. Analiza "top-down" wybranych struktur czasowej sieci Petri'ego

Rozważane w tej sekcji struktury występują w czasowych sieciach Petri'ego modelujących drzewa niezdatności. Ich analiza zostanie wykorzystana w kolejnych rozdziałach do opracowania wzorów dla potrzeb metody INES.

5.2.1. Struktura liniowa



Rys. 24. TPN - struktura liniowa złożona: a) z dwóch miejsc i dwóch przejść, b) z trzech miejsc i trzech przejść

Jeśli chcemy przeprowadzić analizę wsteczną (od ostatniego odpalonego przejścia) TPN z rysunku 24a, to interesuje nas wyznaczenie najwcześniejszego i najpóźniejszego momentu odpalenia przejścia *s*. Wartości te wyznaczymy przyjmując jako dane: α_t i β_t (najwcześniejszy i najpóźniejszy moment odpalenia przejścia *t* w ramach przyjętego upływu czasu, patrz rozdział 3).

Dla struktury liniowej z rysunku 24a otrzymamy:

$$\alpha_s=\alpha_t-\beta^S_t \ \wedge \ \beta_s=\beta_t-\alpha^S_t \tag{17}$$

Z własności sieci Petri'ego, przejście *t* zostanie odpalone nie wcześniej niż po upływie α^{S}_{t}, i nie później niż po upływie β^{S}_{t} jednostek czasu od chwili pojawienia się znacznika w miejscu *m1*, co zapisujemy:

$$\tau(s) + \alpha^{S}_{t} \leq \tau(t) \wedge \tau(t) \leq \tau(s) + \beta^{S}_{t}$$

gdzie:

$\tau(s)$ - moment odpalenia przejścia *s*,

$\tau(t)$ - moment odpalenia przejścia *t*

Zatem:

$$-\tau(t) + \alpha^{S}_{t} \leq -\tau(s) \wedge -\tau(s) \leq -\tau(t) + \beta^{S}_{t}$$

Po wymnożeniu obu stron nierówności przez "-1":

$$\tau(t) - \alpha^{S}_{t} \geq \tau(s) \wedge \tau(s) \geq \tau(t) - \beta^{S}_{t}$$

$$\tau(t) - \alpha^{S}_{t} \geq \tau(s) \geq \tau(t) - \beta^{S}_{t}$$

ponieważ, ze wzoru (5): $\beta_t \geq \tau(t) \geq \alpha_t$, zatem:

$$\beta_t - \alpha^{S}_{t} \geq \tau(t) - \alpha^{S}_{t} \geq \tau(s) \geq \tau(t) - \beta^{S}_{t} \geq \alpha_t - \beta^{S}_{t}$$

$$\beta_t - \alpha^{S}_{t} \geq \tau(s) \geq \alpha_t - \beta^{S}_{t}$$

ponieważ, ze wzoru (5): $\beta_s \geq \tau(s) \geq \alpha_s$, zatem:

$$\alpha_s = \alpha_t - \beta^{S}_{t} \quad \wedge \quad \beta_s = \beta_t - \alpha^{S}_{t}$$

c. b. d. o

Dla TPN z rysunku 24b, otrzymamy:

$$\alpha_r = \alpha_t - \beta^{S}_{t} - \beta^{S}_{s} \wedge \beta_r = \beta_t - \alpha^{S}_{t} - \alpha^{S}_{s} \tag{18}$$

WYPROWADZENIE

Korzystając ze wzoru (17), czasy odpalenia przejść *r* oraz *s* określają zależności:

$$\alpha_s = \alpha_t - \beta^{S}_{t} \quad \wedge \quad \beta_s = \beta_t - \alpha^{S}_{t}$$

$$\alpha_r = \alpha_s - \beta^{S}_{s} \quad \wedge \quad \beta_r = \beta_s - \alpha^{S}_{s}$$

zatem:

$$\alpha_r = \alpha_t - \beta^{S}_{t} - \beta^{S}_{s} \quad \wedge \quad \beta_r = \beta_t - \alpha^{S}_{t} - \alpha^{S}_{s}$$

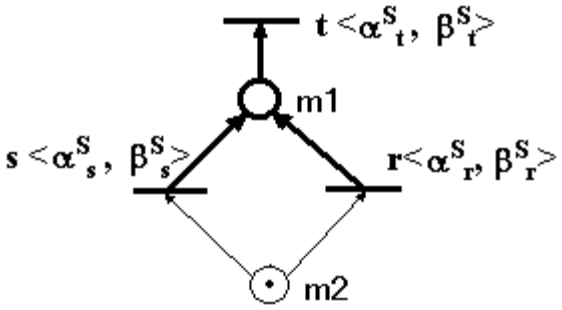
c. b. d. o.

Liczba znaczników w miejscu *m2* na rysunku 24a oraz *m3* na rysunku 24b ograniczają liczbę odpaleń każdego z przejść do jednego razu.

Ograniczenie to wynika z techniki analizy drzew niezdatności. Omawiana analiza "z góry do dołu" zostanie wykorzystana do analizy drzew niezdatności, a dokładniej do wyznaczenia minimalnych zbiorów przyczyn. Nie ma potrzeby rozważania sytuacji,

w których występują dwie przyczyny, mimo iż do wystąpienia hazardu wystarczy tylko jedna z nich, gdyż nie jest to zgodne z ideą minimalnych zbiorów przyczyn, a prowadzi do znacznego skomplikowania analizy.

5.2.2 Struktura "odwrócone Y"



Rys. 25. Sieć Petri'ego -struktura "odwrócone Y"

Zauważmy, że przeprowadzając analizę TPN z rysunku 25, mamy dwa wzajemnie wykluczające się przypadki. Przyjmując, że odpalone zostało przejście *t* mamy: 1. przejście *t* zostało odpalone po odpaleniu przejścia *s* (nie mogło wówczas zostać odpalone przejście *r* ze względu na jeden znacznik w miejscu *m2* oraz zasadę odpalania przejść w TPN),

2. przejście *t* zostało odpalone po odpaleniu przejścia *r* (nie mogło wówczas zostać odpalone przejście *s*).

W pierwszym przypadku mamy strukturę liniową postaci: przejście *t*, miejsce *m1*, przejście *s*, miejsce *m2*; w drugim również, postaci: przejście *t*, miejsce *m1*, przejście *r*, miejsce *m2*. Korzystając ze wzoru (17) oraz uwzględniając dwa wzajemnie wykluczające przypadki, otrzymamy:

Z własności sieci Petri'ego:

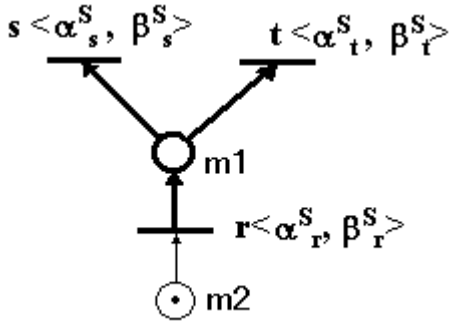
$$(\alpha_s=\alpha_t-\beta^S{}_t \ \wedge \ \beta_s=\beta_t-\alpha^S{}_t) \tag{19}$$

$$\oplus\ (\alpha_r=\alpha_t-\beta^S{}_t \ \wedge \ \beta_r=\beta_t-\alpha^S{}_t)$$

gdzie: $\oplus$ -wzajemne wykluczanie (wyłącznie lub)

Znając moment odpalenia przejścia *t*, możemy wyznaczyć przedział czasu w jakim musiało zostać odpalone przejście *s* lub w jakim musiało zostać odpalone przejście *r*.

5.2.3. Struktura "Y"



Rys. 26. Sieć Petri'ego - struktura "Y"

Założenia.

1. Odpalone zostanie przejście *t*.
2. Dane jest: α_t oraz β_t (parametry odpalenia przejścia *t*).
3. Wyznaczamy SC (warunki statyczne) odpalenia przejścia *t*.
4. Wyznaczymy DC (warunki dynamiczne) - parametry czasowe dla przejść *r* oraz *s*.

Dla TPN z rysunku 26 otrzymamy:

SC

$$\alpha^S_t \leq \beta^S_s \tag{20}$$

DC

$$\alpha_r = \alpha_t - min\{\beta^S_t, \beta^S_s\} \tag{21}$$
$$\wedge\ \beta_r = \beta_t - \alpha^S_t$$
$$\wedge\ \alpha_s = \alpha_t$$
$$\wedge\ \beta_s = \beta_r + \beta^S_s$$

WYPROWADZENIE

Ad SC

Ponieważ przejścia *s* i *t* zostaną przygotowane w tym samym momencie czasu do odpalenia, to zgodnie ze wzorem (10) oraz warunkiem określonym wzorem (7) przejście *t* mogło zostać odpalone tylko wtedy, gdy:

$$\alpha^S_t \leq \beta^S_s$$

c. b. d. o.

Ad DC

Z własności sieci Petri'ego:

$$\tau(t) \leq \tau(s) \wedge \tau(r) + \alpha^S_t \leq \tau(t) \leq \tau(r) + \beta^S_t$$

Korzystając ze wzoru (17) dla odpalenia przejścia $\tau(t)$ otrzymamy:

$$\tau(t) \leq \tau(s) \wedge \alpha_t - \beta^S_t \leq \tau(r) \leq \beta_t - \alpha^S_t$$

Ponieważ:

$\tau(t) \leq \tau(s)$ oraz $\tau(s) \leq \tau(r) + \beta^S_S$ (z własności sieci Petri'ego), zatem z przechodniości nierówności:

$$\tau(t) \leq \tau(r) + \beta^S_s$$

$$\tau(t) - \beta^S_s \leq \tau(r)$$

Wiedząc, że $\tau(t) \geq \alpha_t$:

$$\alpha_t - \beta^S_s \leq \tau(r)$$

Ponieważ β^S_t i β^S_s są z definicji liczbami nieujemnymi, możemy zapisać:

$$\alpha_t - min\{\beta^S_s, \beta^S_t\} \leq \tau(r) \leq \beta_t - \alpha^S_t$$

Korzystając ze wzoru (5): $\beta_r \geq \tau(r) \geq \alpha_r$, zatem:

$$\underline{\alpha_r = \alpha_t - min\{\beta^S_t, \beta^S_s\} \wedge \beta_r = \beta_t - \alpha^S_t}$$

Rozważmy teraz warunki na odpalenie przejścia *s*.

Ponieważ $\tau(s) \geq \tau(t)$ oraz $\tau(t) \geq \alpha_t$ zatem, z przechodniości nierówności:

$$\tau(s) \geq \alpha_t \text{ , to}$$

$$\underline{\alpha_s = \alpha_t}$$

Z własności TPN:

$$\tau(s) \leq \tau(r) + \beta^S_s$$

Wiedząc, że $\tau(r) \leq \beta_r$ otrzymamy:

$$\tau(s) \leq \beta_r + \beta^S_s$$

Zatem:

$$\underline{\beta_s = \beta_r + \beta^S_s}$$

c. b. d. o.

Warunki $\alpha^S_t \leq \beta^S_s$ oraz $\alpha^S_s \leq \beta^S_t$ (jeśli rozważalibyśmy możliwość odpalenia przejścia *s*) są związane z parametrami statycznymi. Ich sprawdzenie jest stosunkowo proste (efektywny algorytm analizy), a ich niespełnienie gwarantuje, że dane przejście nie może zostać odpalone.

W celu przybliżenia wymowy warunków statycznych, rozważmy bardzo uproszczony przykład logowania użytkownika na serwerze.

Przyjmijmy, że proces logowania rozpoczynamy po ustanowieniu połączenia z serwerem i może się on zakończyć tylko na dwa sposoby: normalne zakończenie procesu z komunikatem (o sukcesie lub błędzie) lub poprzez „zabicie" procesu przez proces nadzorujący na skutek przekroczenia limitu czasu.

Niech:

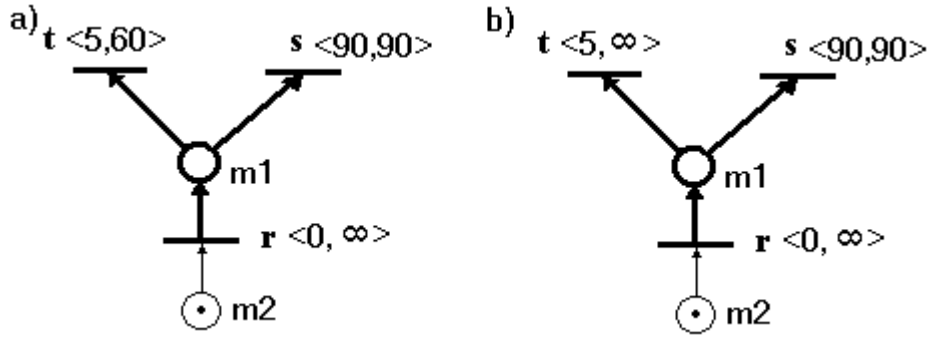
1) miejsce *m1* odpowiada zdarzeniu "proces logowania do serwera",

2) przejście *r* odpowiada startowi procesu logowania,

3) przejście *t* - zakończenie procesu logowania,

4) przejście *s* - „zabicie" procesu logowania (upłynął limit czasu).

Przykładowe parametry czasowe dla tak zdefiniowanych zdarzeń pokazuje rysunek 27.



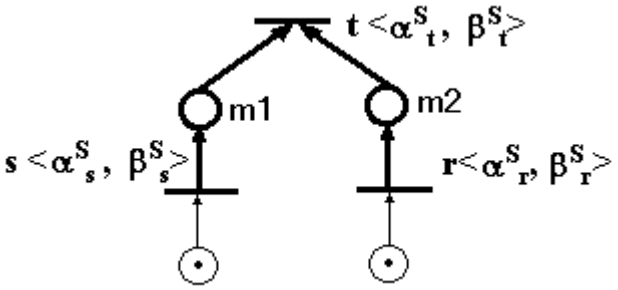
Rys. 27. TPN modelująca "proces logowania do serwera" a) konstrukcja oprogramowania gwarantuje zakończenie procesu logowania nie później niż po 60 jednostkach czasu, b) konstrukcja oprogramowania nie pozwala na oszacowanie

maksymalnego czasu potrzebnego do zakończenia logowania (może trwać nieskończenie długo)

Jeśli oprogramowanie jest tak napisane, że proces logowania zawsze zakończy się nie później niż po 60 jednostkach czasu, to w rozważanym systemie nigdy nie nastąpi „zabicie" procesu logowania (rysynek 26a). Nigdy nie zostanie odpalone przejście *s*, gdyż nie jest spełniony warunek: $\alpha^S_s \leq \beta^S_t$.

W przeciwnym razie, jeśli nie potrafimy lub nie możemy określić maksymalnego czasu logowania (jak na rysunku 26 b) - gdyż może trwać dowolną ilość czasu, to zakończenie procesu logowania może mieć miejsce również na skutek przekroczenia limitu czasu.

5.2.4. Struktura pomocnicza "odwrócone V"



Rys. 28. Sieć Petri'ego -struktura "odwrócone V"

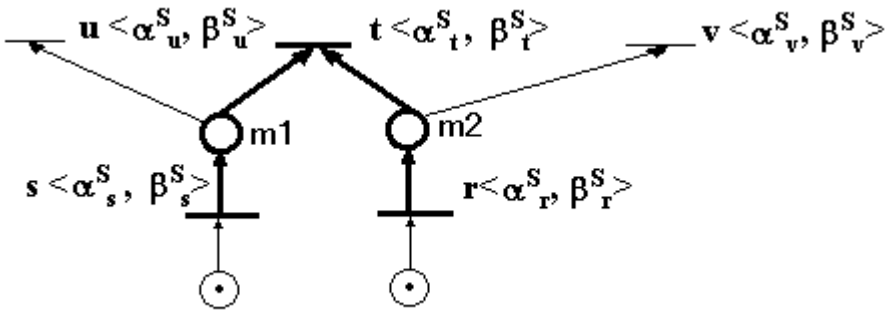
Z własności sieci Petri'ego:

$$max\{\tau(s),\ \tau(r)\} + \alpha^S_t \leq \tau(t) \leq max\{\tau(s),\ \tau(r)\} + \beta^S_t \qquad (22)$$

Ze względu na wybór $max\{\tau(s),\ \tau(r)\}$ przy wyznaczaniu czasów, w jakich musiały zostać odpalone przejścia *s* i *r* otrzymamy dwa przypadki. Pierwszy dla $\tau(s) \leq \tau(r)$, natomiast drugi dla $\tau(r) \leq \tau(s)$:

1. $\tau(s) \leq \tau(r) \wedge \tau(t) - \beta^S_t \leq \tau(r) \leq \tau(t) - \alpha^S_t$ (23a)
2. $\tau(r) \leq \tau(s) \wedge \tau(t) - \beta^S_t \leq \tau(s) \leq \tau(t) - \alpha^S_t$ (23b)

5.2.5. Struktura "W"



Rys. 29. Sieć Petri'ego - struktura "W"

Struktura przedstawiona na rysunku 29 jest poszerzeniem TPN przedstawionej w sekcji 5.2.4.

Przyjmijmy, że dane są: α_t oraz β_t (parametry odpalenia przejścia *t*). Przyjmując, że przejście *t* zostało odpalone, wyznaczymy parametry czasowe dla pozostałych przejść w TPN.

SC:

Z własności sieci Petri'ego wiemy, że przejście *t* może zostać odpalone tylko wtedy, gdy jest spełniony warunek:

$$\beta^S_u \geq \alpha^S_t \ \wedge \ \beta^S_v \geq \alpha^S_t \tag{24}$$

DC:

Jeśli warunek (24) będzie spełniony, to mamy dwa przypadki (wzór ((23a), oraz (23b)). Ponieważ (z założenia) odpalone zostało przejście *t,* dodatkowo musimy uwzględnić warunki: $\tau(t) \leq \tau(u) \wedge \tau(t) \leq \tau(v)$

Rozważmy przypadek, gdy: $\boldsymbol{\tau(r) \geq \tau(s)}$.

Otrzymamy:

a) dla przejścia *r*:

$$\alpha_r = \alpha_t - min\{\beta^S_t, \beta^S_u, \beta^S_v\} \ \wedge \beta_r = \beta_t - \alpha^S_t \tag{25}$$

b) dla przejścia *s*:

$$\alpha_s = \alpha_t - \beta^S_u \wedge \beta_s = \beta_r \tag{26}$$

c) dla przejścia *v*:

$$\alpha_v = \alpha_t \wedge \beta_v = \beta_r + \beta^S_v \tag{27}$$

d) dla przejścia *u*:

$$\alpha_u = \alpha_t \wedge \beta_u = \beta_s + \beta^S_u \tag{28}$$

WYPROWADZENIE

Ad a)

Z własności sieci Petri'ego:

$$\tau(u) \le \tau(s) + \beta^S_u$$

$$\tau(v) \le \tau(r) + \beta^S_v$$

ze wzorów (23a) oraz uwzględniając $\tau(t) \le \tau(u) \wedge \tau(t) \le \tau(v)$:

$$\tau(t) \le \tau(u) \wedge \tau(t) \le \tau(v) \wedge \tau(s) \le \tau(r) \wedge \tau(t) - \beta^S_t \le \tau(r) \le \tau(t) - \alpha^S_t$$

to:

$$\tau(t) \le \tau(u) \le \tau(s) + \beta^S_u \le \tau(r) + \beta^S_u \quad \Rightarrow \tau(t) - \beta^S_u \le \tau(r)$$

$$\wedge \quad \tau(t) \le \tau(v) \le \tau(r) + \beta^S_v \Rightarrow \tau(t) - \beta^S_v \le \tau(r)$$

$$\wedge \quad \tau(t) - \beta^S_t \le \tau(r)$$

$$\wedge \quad \tau(r) \le \tau(t) - \alpha^S_t$$

Ponieważ: β^S_t i β^S_u i $\beta^S_v \in Q_+$, zatem:

$$\tau(t) - min\{\beta^S_t, \beta^S_u, \beta^S_v\} \le \tau(r) \le \tau(t) - \alpha^S_t$$

Wiedząc z założenia -wzór nr (5), że $\alpha_t \le \tau(t) \le \beta_t$ oraz korzystając z przechodniości nierówności:

$$\alpha_t - min\{\beta^S_t, \beta^S_u, \beta^S_v\} \le \tau(r) \le \beta_t - \alpha^S_t$$

Ze wzoru (5): $\alpha_r \le \tau(r) \le \beta_r$, zatem:

$$\underline{\alpha_r = \alpha_t - min\{\beta^S_t, \beta^S_u, \beta^S_v\} \wedge \beta_r = \beta_t - \alpha^S_t}$$

c. b. d. o.

Ad b)

Wiedząc, że $\tau(s) \le \tau(r)$ -z założenia dla rozważanego przypadku oraz $\alpha_r \le \tau(r) \le \beta_r$ - ze wzoru (5), otrzymamy:

$$\tau(s) \le \tau(r) \le \beta_r \Rightarrow \tau(s) \le \beta_r$$

Zatem możemy wyznaczyć najpóźniejszy czas odpalenia przejścia *s*:

$$\underline{\beta_s = \beta_r}$$

Szukając najwcześniejszego momentu czasu, w którym musi zostać odpalone przejście *s*, skorzystamy z nierówności:

$\tau(t) \le \tau(u)$ - z założenia dla rozważanego przypadku

$\wedge \alpha_t \le \tau(t)$

$\wedge \tau(u) \le \tau(s) + \beta^S_u$ - z własności sieci Petri'ego,

stąd:

$$\alpha_t \le \tau(s) + \beta^S_u \Rightarrow \alpha_t - \beta^S_u \le \tau(s)$$

to najwcześniejszy moment odpalenia przejścia *s* wynosi:

$$\underline{\alpha_s = \alpha_t - \beta^S_u}$$

c. b. d. o.

Ad c)

Ponieważ $\tau(t) \leq \tau(v)$ oraz $\alpha_t \leq \tau(t)$, zatem (z przechodniości nierówności):

$$\alpha_t \leq \tau(v)$$

Zatem:

$$\underline{\alpha_v = \alpha_t}$$

Z własności sieci Petri'ego:

$$\underline{\beta_v = \beta_t + \beta^S_v}$$

c. b. d. o.

Ad d)

Analogicznie do przypadku c), otrzymamy:

$$\underline{\alpha_u = \alpha_t}$$

$$\underline{\beta_u = \beta_s + \beta^S_u}$$

c. b. d. o.

Postępując w analogiczny sposób dla $\tau(s) \geq \tau(r)$ otrzymamy:

$$\alpha_r = \alpha_t - \beta^S_v \wedge \beta_r = \beta_s \tag{29}$$

$$\wedge\ \alpha_s = \alpha_t - min\{\beta^S_t, \beta^S_u, \beta^S_v\} \wedge \beta_s = \beta_t - \alpha^S_t$$

$$\wedge\ \alpha_v = \alpha_t \wedge \beta_v = \beta_r + \beta^S_v$$

$$\wedge\ \alpha_u = \alpha_t \wedge \beta_u = \beta_s + \beta^S_u$$

5.3. Wyznaczenie parametrów hazardu

W procesie analizy wyznaczamy parametry dynamiczne dla zdarzeń występujących w drzewie niezdatności. Parametry te określają, w jakich momentach czasu muszą wystąpić zdarzenia, by prowadziły do hazardu. Parametry dynamiczne dla dowolnego zdarzenia *z* zapisujemy:

- *zs* $<\alpha_{zs}, \beta_{zs}>$ najwcześniejszy i najpóźniejszy moment czasu startu zdarzenia (momenty czasu liczymy w ramach przyjętego upływu czasu w systemie),
- *ze* $<\alpha_{ze}, \beta_{ze}>$ najwcześniejszy i najpóźniejszy moment czasu zakończenia zdarzenia (momenty czasu liczymy w ramach przyjętego upływu czasu w systemie).

Dla przypomnienia: α_{ze} -parametr dynamiczny, α^S_{ze} -parametr statyczny.

Analizę rozpoczynamy od przyjęcia umownego momentu "0". Za taki moment możemy przyjąć np. najpóźniejszy moment startu hazardu:

$$\beta_{hs}=0$$

Najwcześniejszy moment jego startu wyznaczymy korzystając z parametrów statycznych.

Z własności sieci Petri'ego wiemy, że czas odpalenia przejścia *τ(hs)*, liczony od momentu przygotowania przejścia do odpalenia określa, nierówność:

$$\alpha^S_{hs} \leq \tau(hs) \leq \beta^S_{hs}$$

Przyjmijmy na chwilę, że moment przygotowania przejścia *hs* do odpalenia jest umownym momentem „0", wówczas najwcześniejszy i najpóźniejszy moment czasu, w którym przejście *hs* może zostać odpalone są równe wartościom statycznym:

$$\alpha_{hs}= \alpha^S_{hs},\ \beta_{hs}=\beta^S_{hs}$$

„Przesuńmy" teraz umowny moment „0" o wartość $-\beta^S_{hs}$, wówczas:

$$\alpha_{hs}= \alpha^S_{hs}-\beta^S_{hs},\ \beta_{hs}=0 \tag{30}$$

Czas zakończenia hazardu wyznaczamy z własności sieci Petri'ego:

$$<\alpha_{he},\beta_{he}>=<\alpha_{hs}+\alpha^S_{he},\ \beta_{hs}+\beta^S_{he}>=<\alpha^S_{hs}-\beta^S_{hs}+\alpha^S_{he},\beta^S_{he}> \tag{31}$$

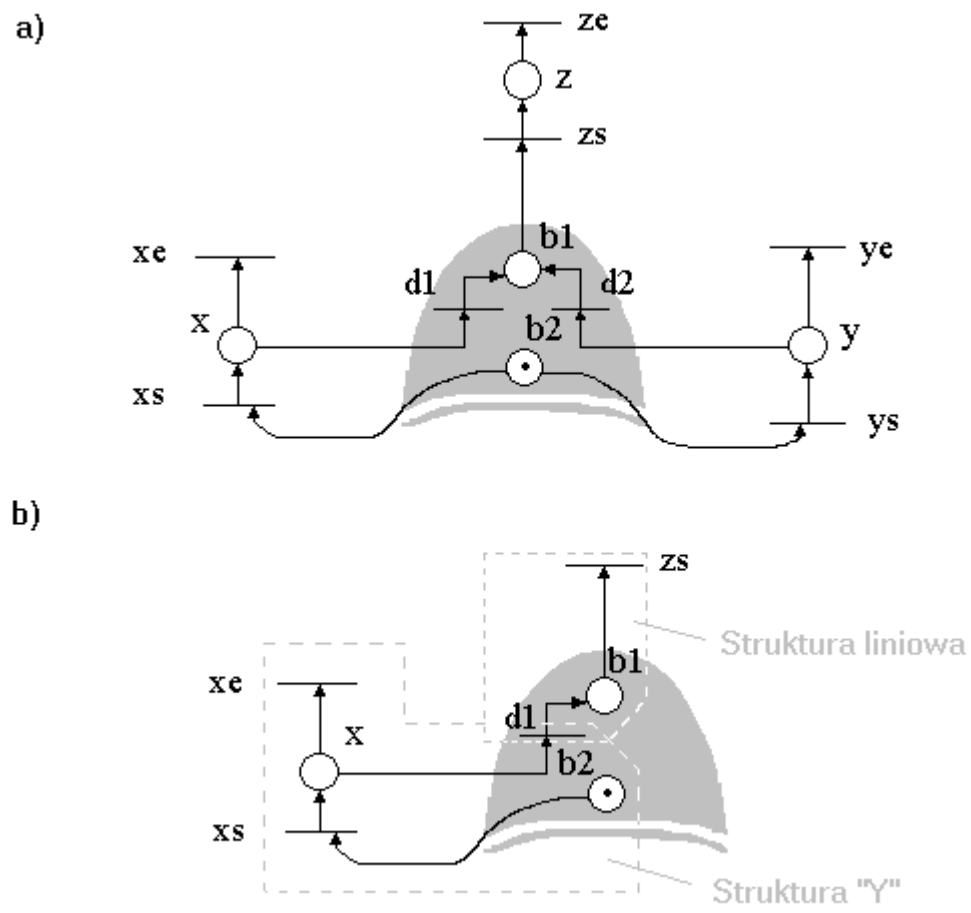
Mając wyznaczony umowny moment startu hazardu oraz wyznaczony czas jego zakończenia, możemy rozpocząć analizę TPN z "góry do dołu".

5.4. System nierówności i równań dla bramek występujących w drzewach niezdatności

5.4.1. Bramka przyczynowa XOR

Model bramki z zaznaczonymi zdarzeniami pokazuje rysunek 23. Przyjmujemy, że dane jest *zs* oraz *ze* i wyznaczamy parametry czasowe zdarzeń *xs*, *xe*, *ys*, *ye*.

Parametry czasowe *zs* i *ze* możemy przyjąć jako dane, gdyż zostaną one wyznaczone jako parametry hazardu (dla bramki w najwyższym poziomie drzewa) lub zostaną wyznaczone w jednym z wcześniejszych etapów analizy (dla wszystkich pozostałych bramek).



Rys. 30. Bramka przyczynowa XOR, a) model bramki z zaznaczonymi przejściami oraz parametrami dynamicznymi, b) fragment bramki związany z wyznaczeniem parametrów czasowych dla przejść *xs* oraz *xe*

Ze względu na to, że w miejscu b2 znajduje się jeden znacznik oraz ze względu na zasadę odpaleń przejść otrzymamy dla TPN modelującej bramkę XOR dwa wzajemnie wykluczające przypadki:

1. przejście *zs* zostało odpalone po odpaleniu przejścia *xs*,
2. przejście *zs* zostało odpalone po odpaleniu przejścia *ys*.

Wyznaczmy teraz parametry dla zdarzeń *x* oraz *y* przyjmując poniższe założenia.

Założenia:

Dane jest: α_{zs} oraz β_{zs} (parametry odpalenia przejścia *zs*).

Zatem dla bramki XOR otrzymamy dwa przypadki:

$$\begin{cases} \alpha_{d1}^{S} \le \beta_{xe}^{S} \\ \alpha_{xs} = \alpha_{zs} - min\{\beta_{d1}^{S}, \beta_{xe}^{S}\}, \quad \beta_{xs} = \beta_{zs} - \alpha_{d1}^{S} \\ \alpha_{xe} = \alpha_{zs}, \beta_{xe} = \beta_{xs} + \beta_{xe}^{S} \end{cases} \tag{32a}$$

(32)

$$\oplus \tag{32b}$$

$$\begin{cases} \alpha_{d2}^{S} \le \beta_{ye}^{S} \\ \alpha_{ys} = \alpha_{zs} - min\{\beta_{d2}^{S}, \beta_{ye}^{S}\}, \ \beta_{ys} = \beta_{zs} - \alpha_{d2}^{s} \\ \alpha_{ye} = \alpha_{zs}, \ \beta_{ye} = \beta_{ys} + \beta_{ye}^{S} \end{cases}$$

WYPROWADZENIE

Ad 1.

TPN związana z przypadkiem pierwszym została przedstawiona na rysunku 30.

Struktura liniowa.

Ponieważ: *zs* <*0,0*> -z definicji modelu TPN dla bramki XOR (sekcja 4.2), ze wzoru (17) (przyjmując *zs*=*t*, *d1*=*s*) otrzymamy:

$$\alpha_{d1}=\alpha_{zs} \ \wedge \ \beta_{d1}=\beta_{zs}$$

Struktura "Y".

Ze wzorów (20) oraz (21), przyjmując *d1*=*t*, *xe*=*s*, *xs*=*r*, otrzymamy:

SC

$$\alpha^{S}_{d1} \le \beta^{S}_{xe}$$

DC

$$\alpha_{xs} = \alpha_{d1} - min\{\beta^S_{d1}, \beta^S_{xe}\} \wedge \beta_{xs} = \beta_{d1} - \alpha^S_{d1}$$
$$\wedge\ \alpha_{xe} = \alpha_{d1} \wedge \beta_{xe} = \beta_{xs} + \beta^S_{xe}$$

Korzystając z obliczeń dla struktury liniowej, ostatecznie:

SC

$$\alpha^S_{d1} \leq \beta^S_{xe}$$

DC

$$\alpha_{xs} = \alpha_{zs} - min\{\beta^S_{d1}, \beta^S_{xe}\} \wedge \beta_{xs} = \beta_{zs} - \alpha^S_{d1}$$
$$\wedge\ \alpha_{xe} = \alpha_{zs} \wedge \beta_{xe} = \beta_{xs} + \beta^S_{xe}$$

c. b. d. o.

AD 2.

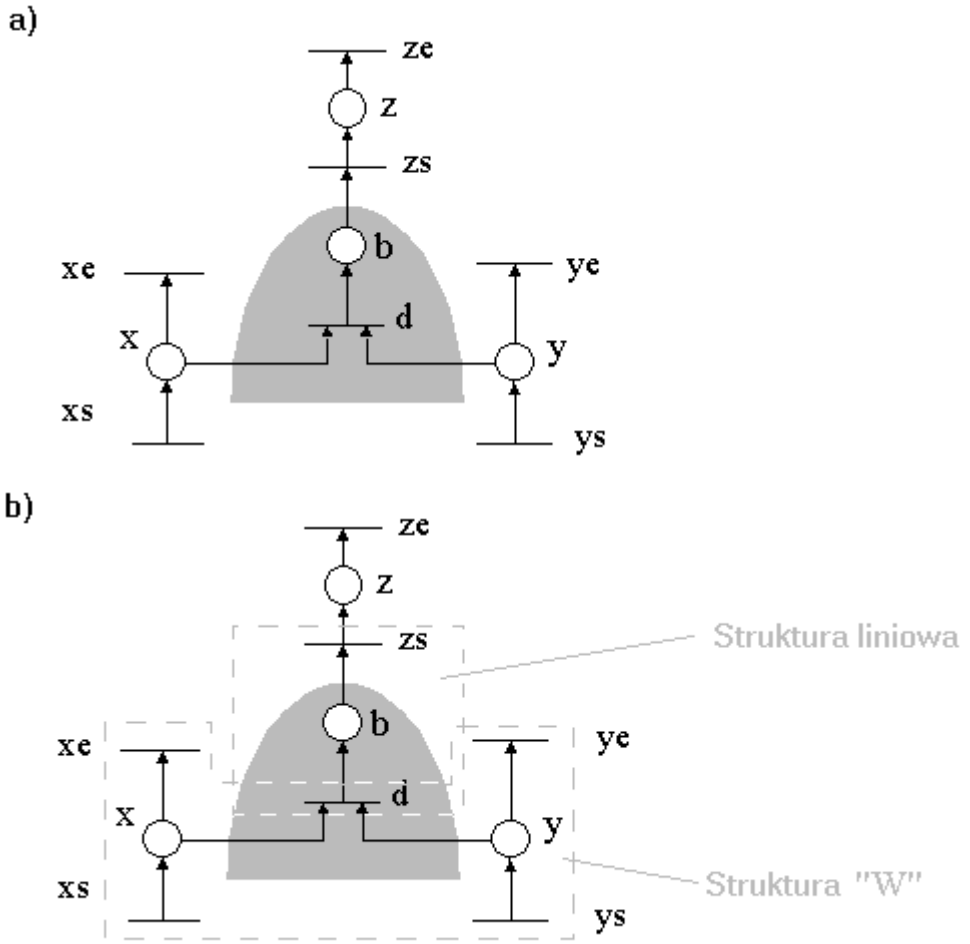
Analogicznie do przypadku 1.

Podsumowanie.

Dla bramki przyczynowej XOR otrzymaliśmy dwa przypadki. Pierwszy, gdy zdarzenie *z* jest skutkiem zdarzenia *x*, drugi -zdarzenie *z* jest skutkiem zdarzenia *y*.

Znając czasy odpalenia przejścia $z < \alpha_{zs}, \beta_{zs} >$, możemy wyznaczyć parametry czasowe dla zdarzeń *x* oraz *y*. Parametry te opisuje wzór (32).

5.4.2. Bramka przyczynowa AND



Rys. 31. Bramka przyczynowa AND, a) model bramki b) model bramki z zaznaczoną strukturą liniową oraz strukturą "W"

Podobnie jak w przypadku bramki XOR przyjmijmy, że dane są: α_{zs}, β_{zs}. Otrzymamy wówczas system nierówności dany wzorem (33). Nierówności te pozwalają na wyznaczenie parametrów czasowych zdarzeń *x* i *y*.

$$\begin{cases} \alpha_d^S \le \beta_{xe}^S & \text{(33a)} \\ \alpha_d^S \le \beta_{ye}^S \end{cases} \qquad \text{(33)}$$

and

$$\begin{cases} \begin{cases} \alpha_{xs} = \alpha_{zs} - \min\{\beta_d^S, \beta_{xe}^S, \beta_{ye}^S\},\ \beta_{xs} = \beta_{zs} - \alpha_d^S & \text{(33b)} \\ \alpha_{ys} = \alpha_{zs} - \beta_{ye}^S,\ \beta_{ys} = \beta_{xs} \\ \alpha_{xe} = \alpha_{zs},\ \beta_{xe} = \beta_{xs} + \beta_{xe}^S \\ \alpha_{ye} = \alpha_{zs},\ \beta_{ye} = \beta_{ys} + \beta_{ye}^S \end{cases} \\ \text{or} \\ \begin{cases} \alpha_{ys} = \alpha_{zs} - \min\{\beta_d^S, \beta_{xe}^S, \beta_{ye}^S\},\ \beta_{ys} = \beta_{zs} - \alpha_d^S & \text{(33c)} \\ \alpha_{xs} = \alpha_{zs} - \beta_{xe}^S,\ \beta_{xs} = \beta_{ys} \\ \alpha_{xe} = \alpha_{zs},\ \beta_{xe} = \beta_{xs} + \beta_{xe}^S \\ \alpha_{ye} = \alpha_{zs},\ \beta_{ye} = \beta_{ys} + \beta_{ye}^S \end{cases} \end{cases}$$

WYPROWADZENIE

Struktura liniowa.

Ponieważ: zs <0,0> -z definicji modelu TPN dla bramki przyczynowej AND (sekcja 4.3), zatem ze wzoru (17) (przyjmując *zs*=*t*, *d*=*s*):

$$\alpha_d = \alpha_{zs} \wedge \beta_d = \beta_{zs}$$

Struktura "W".

Przyjmijmy: *s*=*xs*, *u*=*xe*, *t*=*d*, *r*=*ys*, *v*=*ye*.

SC

Ze wzoru (24):

$$\underline{\beta^S_{xe} \ge \alpha^S_d \wedge \beta^S_{ye} \ge \alpha^S_d}$$

DC

Ze wzorów (25) - (28) pierwszy przypadek:

$$\alpha_{ys} = \alpha_d - min\{\beta^S_d, \beta^S_{xe}, \beta^S_{ye}\} \wedge \beta_{ys} = \beta_d - \alpha^S_d \wedge \alpha_{xs} = \alpha_d - \beta^S_{xe} \wedge \beta_{xs} = \beta_{ys}$$
$$\wedge \alpha_{ye} = \alpha_{ys} \wedge \beta_{ye} = \beta_{ys} + \beta^S_{ye} \wedge \alpha_{xe} = \alpha_{ys} \wedge \beta_{xe} = \beta_{xs} + \beta^S_{xe}$$

lub, ze wzoru (29), drugi:

$$\alpha_{ys} = \alpha_d - \beta^S_{ye} \wedge \beta_{ys} = \beta_{xs} \wedge \alpha_{xs} = \alpha_d - min\{\beta^S_d, \beta^S_{xe}, \beta^S_{ye}\} \wedge \beta_{xs} = \beta_d - \alpha^S_d$$
$$\wedge \alpha_{ye} = \alpha_{ys} \wedge \beta_{ye} = \beta_{ys} + \beta^S_{ye} \wedge \alpha_{xe} = \alpha_{ys} \wedge \beta_{xe} = \beta_{xs} + \beta^S_{xe}$$

Podstawiając za $\alpha_d = \alpha_{zs}$ *i* $\beta_d = \beta_{zs}$ (zgodnie z analizą dla struktury liniowej):

$$\alpha_{ys} = \alpha_{zs} - min\{\beta^S_d, \beta^S_{xe}, \beta^S_{ye}\} \wedge \beta_{ys} = \beta_{zs} - \alpha^S_d \wedge \alpha_{xs} = \alpha_{zs} - \beta^S_{xe} \wedge \beta_{xs} = \beta_{ys}$$
$$\wedge \alpha_{ye} = \alpha_{ys} \wedge \beta_{ye} = \beta_{ys} + \beta^S_{ye} \wedge \alpha_{xe} = \alpha_{ys} \wedge \beta_{xe} = \beta_{xs} + \beta^S_{xe}$$

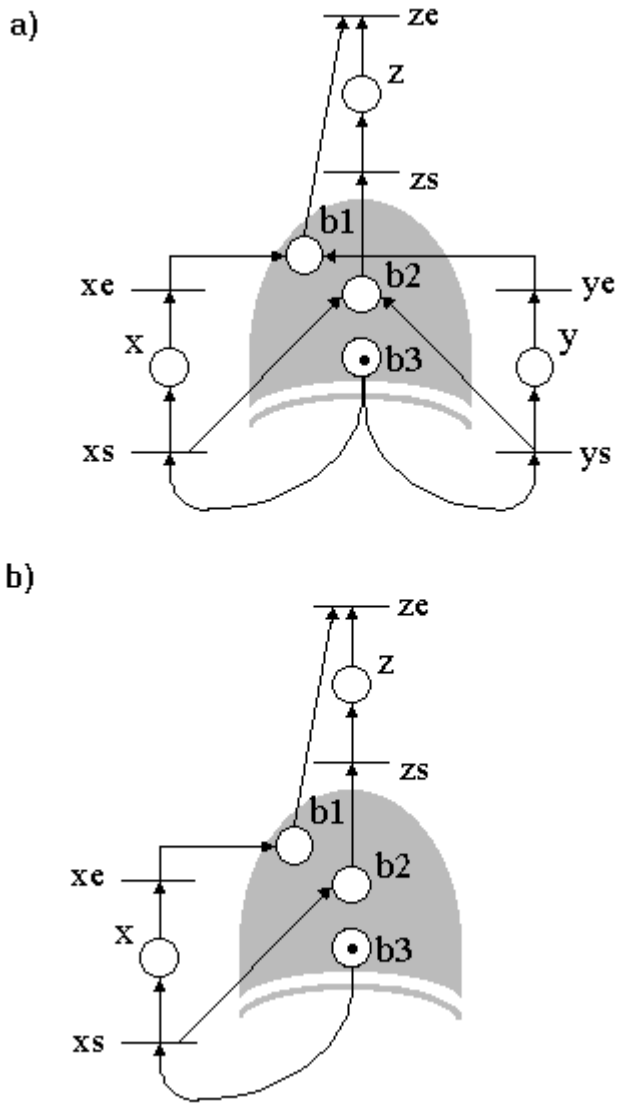
lub

$$\alpha_{ys} = \alpha_{zs} - \beta^S_{ye} \wedge \beta_{ys} = \beta_{xs} \wedge \alpha_{xs} = \alpha_{zs} - min\{\beta^S_d, \beta^S_{xe}, \beta^S_{ye}\} \wedge \beta_{xs} = \beta_{zs} - \alpha^S_d$$

$$\wedge \alpha_{ye} = \alpha_{ys} \wedge \beta_{ye} = \beta_{ys} + \beta^S_{ye} \wedge \alpha_{xe} = \alpha_{ys} \wedge \beta_{xe} = \beta_{xs} + \beta^S_{xe}$$

c. b. d. o.

5.4.3 Bramka uogólniająca XOR



Rys. 32. Brama uogólniająca XOR, a) model bramki b) fragment modelu bramki związany z odpaleniem przejścia *xs*

Dla bramki uogólniającej XOR otrzymamy dwa przypadki:

$$\begin{cases} \boldsymbol{\alpha}_{xs} = \boldsymbol{\alpha}_{zs}, \boldsymbol{\beta}_{xs} = \boldsymbol{\beta}_{zs} \\ \boldsymbol{\alpha}_{xe} = \boldsymbol{\alpha}_{ze}, \boldsymbol{\beta}_{xe} = \boldsymbol{\beta}_{ze} \end{cases} \quad (34)$$
$$\oplus$$
$$\begin{cases} \boldsymbol{\alpha}_{ys} = \boldsymbol{\alpha}_{zs}, \boldsymbol{\beta}_{ys} = \boldsymbol{\beta}_{zs} \\ \boldsymbol{\alpha}_{ye} = \boldsymbol{\alpha}_{ze}, \boldsymbol{\beta}_{ye} = \boldsymbol{\beta}_{ze} \end{cases}$$

WYPROWADZENIE

Założenie:

Dane są: α_{zs}, β_{zs}, α_{ze}, β_{ze} oraz $\alpha^S_{zs}=0$, $\beta^S_{zs}=0$, $\alpha^S_{ze}=0$, $\beta^S_{ze}=0$ z definicji bramki uogólniającej.

Rozważmy TPN z rysunku 32b.

Odpalenie przejścia *zs* względem *xs* określa wzór (17), zatem:

$$\alpha_{xs}=\alpha_{zs}-\beta^S_{zs} \ \wedge \ \beta_{xs}=\beta_{zs}-\alpha^S_{zs}$$
$$\alpha_{xs}=\alpha_{zs}-0 \wedge \ \beta_{xs}=\beta_{zs}-0$$
$$\underline{\alpha_{xs}=\alpha_{zs} \wedge \ \beta_{xs}=\beta_{zs}}$$

Ponieważ $\tau(zs)=\tau(xs)$ oraz z własności sieci Petri'ego wiemy, że najpierw musi być odpalone przejście *xs*, by można było odpalić przejście *xe*, zatem $\tau(xe)\geq\tau(xs)$, to:

$$\tau(xe)\geq\tau(zs)$$

Ze wzoru (23b) otrzymamy ($t=ze$, $s=xe$):

$$\tau(ze)-\beta^S_{ze} \leq \tau(xe) \leq \tau(ze)-\alpha^S_{ze}$$
$$\tau(ze)-0 \leq \tau(xe) \leq \tau(ze)-0$$
$$\tau(xe)=\tau(ze)$$

Zatem:

$$\underline{\alpha_{xe}=\alpha_{ze} \wedge \ \beta_{xe}=\beta_{ze}}$$

c. b. d. o.

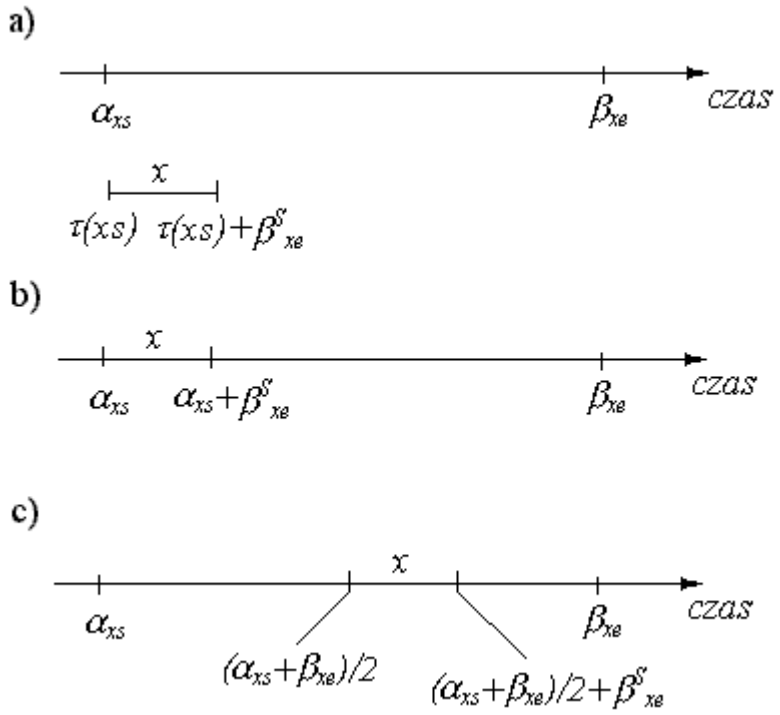
W tym miejscu wyjaśnienia wymaga również zależność pomiędzy *xs* oraz *xe*. Na rysunku 33 przedstawiłem jedną z wielu możliwych sytuacji.

Na rysunek 33a przedstawiony jest przedział czasu $<\alpha_{xs}, \beta_{xe}>$ oraz odcinek, który pokazuje maksymalną „długość" zdarzenia *x*. Jeśli zdarzenie *x* prowadzi do hazardu, to musi się rozpocząć nie wcześniej niż α_{xs} i zakończyć nie później niż β_{xe}, czyli zawiera się w przedziale $<\alpha_{xs}, \beta_{xe}>$.

Maksymalny czas trwania zdarzenia *x* określa wartość parametru statycznego β^S_{xe}.

Wielkości dynamiczne (w rozumieniu metody INES) nie określają czasu trwania zdarzenia, lecz warunki, kiedy i po jakim czasie jego wystąpienie może prowadzić do hazardu. Biorąc pod uwagę wszystkie możliwe zdarzenia – warunki występowania zdarzeń oraz ich korelacja czasowa, która może spowodować hazard.

Przykład przedstawiony na rysunku, jak zaznaczyłem wcześniej, jest tylko jedną z wielu możliwości, na przykład inna sytuacja to taka, gdy maksymalny czas trwania zdarzenia będzie dłuższy niż przedział czasu określony parametrami α_{xs}, β_{xe}.



Rys. 33. Przykład zależność między najwcześniejszym i najpóźniejszym momentem czasu odpalenia przejścia *xs*, a maksymalnym czasem trwania zdarzenia *x* a) przykładowe wielkości przedziałów czasowych, gdzie *x*-maksymalny przedział czasu określający trwanie zdarzenia *x*; wielkość przedziału $=\beta^S_{xe}$, b) położenie przedziałów czasu dla $\tau(xs)=\alpha_{xs}$, c) położenie przedziałów czasu dla $\tau(xs)=(\alpha_{xs}+\beta_{xe})/2$

5.4.4. Bramka uogólniająca AND

Dla bramki uogólniającej AND otrzymamy cztery przypadki określone wzorem (35).

$$
\begin{cases}
\begin{cases} \alpha_{xs} = \alpha_{zs},\ \beta_{xs} = \beta_{zs},\ \alpha_{xe} = \alpha_{ye},\ \beta_{xe} = \beta_{xs} + \beta^{S}_{xe} \\ \alpha_{ys} = \alpha_{xs} - \beta^{S}_{ye},\ \beta_{ys} = \beta_{xs},\ \alpha_{ye} = \alpha_{ze},\ \beta_{ye} = \beta_{ze}, \end{cases} & \text{(35a)} \\
xor & \\
\begin{cases} \alpha^{S}_{ye} \le \beta^{S}_{xe} \\ \alpha_{ys} = \alpha_{zs},\ \beta_{ys} = \beta_{zs},\ \alpha_{ye} = \alpha_{ze},\ \beta_{ye} = \beta_{ze} \\ \alpha_{xs} = \alpha_{ye} - \beta^{S}_{xe},\ \beta_{xs} = \beta_{ys},\ \alpha_{xe} = \alpha_{ye},\ \beta_{xe} = \beta_{xs} + \beta^{S}_{xe} \end{cases} & \text{(35b)} \\
xor & \\
\begin{cases} \alpha_{ys} = \alpha_{zs},\ \beta_{ys} = \beta_{zs},\ \alpha_{ye} = \alpha_{xe},\ \beta_{ye} = \beta_{ys} + \beta^{S}_{ye} \\ \alpha_{xs} = \alpha_{ys} - \beta^{S}_{xe},\ \beta_{xs} = \beta_{ys},\ \alpha_{xe} = \alpha_{ze},\ \beta_{xe} = \beta_{ze}, \end{cases} & \text{(35c)} \\
xor & \\
\begin{cases} \alpha^{S}_{xe} \le \beta^{S}_{ye} \\ \alpha_{xs} = \alpha_{zs},\ \beta_{xs} = \beta_{zs},\ \alpha_{xe} = \alpha_{ze},\ \beta_{xe} = \beta_{ze} \\ \alpha_{ys} = \alpha_{xe} - \beta^{S}_{ye},\ \beta_{ys} = \beta_{xs},\ \alpha_{ye} = \alpha_{xe},\ \beta_{ye} = \beta_{ys} + \beta^{S}_{ye} \end{cases} & \text{(35d)}
\end{cases}
\qquad (35)
$$

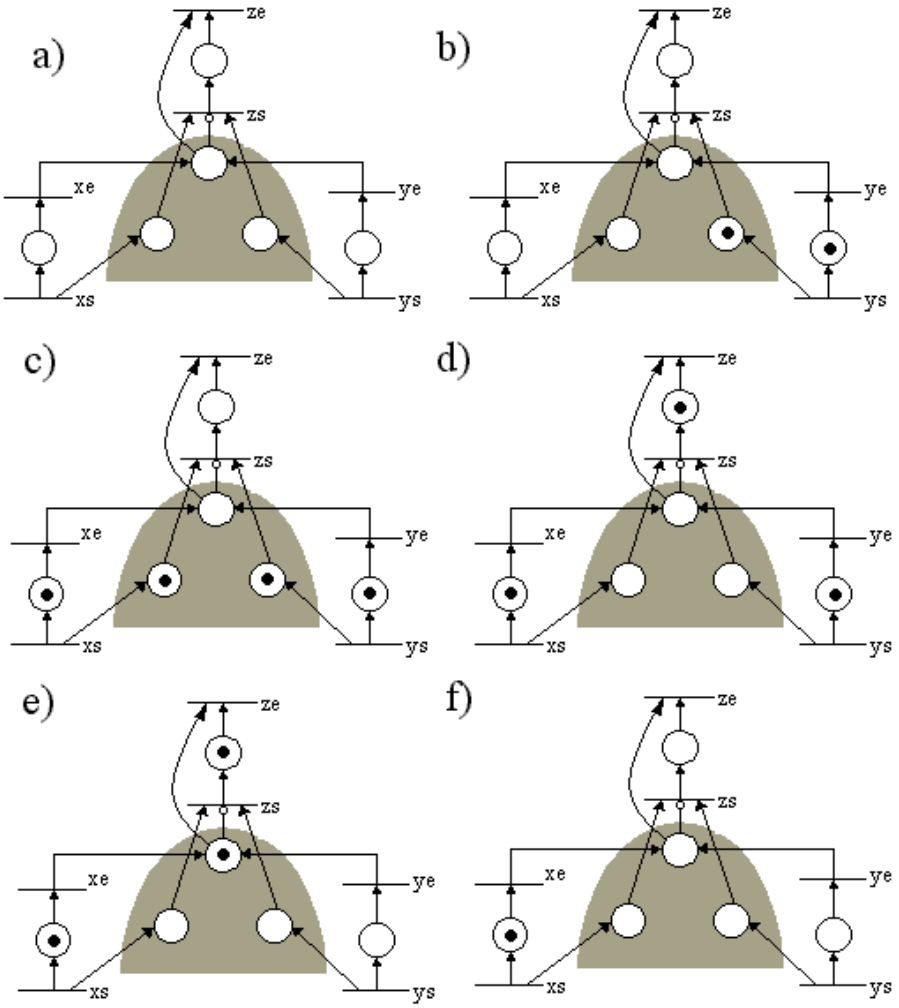
WYPROWADZENIE

Dla bramki uogólniającej AND (rys. 20), ze względu na relacje czasowe związane z kolejnością występowania zdarzeń *x* oraz *y* w czasie, rozważymy cztery przypadki (związane z wystąpieniem zdarzenia *z)*:

1. $\tau(ys) \le \tau(xs) \le \tau(ye) \le \tau(xe)$ – jako pierwsze rozpoczyna się zdarzenie y, następnie rozpoczyna się zdarzenie x, kończy się zdarzenie y i kończy się zdarzenie x,
2. $\tau(xs) \le \tau(ys) \le \tau(ye) \le \tau(xe)$ – zdarzenie y rozpoczyna się i kończy w trakcie trwania zdarzenia x (po jego rozpoczęciu i przed jego zakończeniem),
3. $\tau(xs) \le \tau(ys) \le \tau(xe) \le \tau(ye)$ – jako pierwsze rozpoczyna się zdarzenie x, następnie rozpoczyna się zdarzenie y, kończy się zdarzenie x i kończy się zdarzenie y,
4. $\tau(ys) \le \tau(xs) \le \tau(xe) \le \tau(ye)$ – zdarzenie x rozpoczyna się i kończy w czasie, gdy ma miejsce zdarzenie y.

Ad 1. $\tau(ys) \le \tau(xs) \le \tau(ye) \le \tau(xe)$

Sekwencję odpalania przejść dla tego przypadku pokazuje rysunek 34.



Rys. 34. Model bramki generalization AND ilustrujący odpalanie przejść w kolejności: a) przed odpaleniem przejść, b) *ys*, c) *xs*, d) *zs*, e) *ye* f) *ze*.

Rozważany przypadek dotyczy sytuacji, w której wystąpiły zdarzenia *x* i *y* oraz start zdarzenia *y* jest równy startowi zdarzenia *z*, a koniec zdarzenia *y* jest równy końcowi zdarzenia *z*. Zatem z definicji otrzymamy:

$$\underline{\alpha_{xs}=\alpha_{zs},\ \beta_{xs}=\beta_{zs}} \text{ oraz } \underline{\alpha_{ye}=\alpha_{ze},\ \beta_{ye}=\beta_{ze}}$$

Powyższe zależności uzyskamy również przeprowadzając rozumowanie w oparciu o własności sieci Petri'ego oraz nierówności podane w definicji tego przypadku:

Przejście *zs* staje się przygotowane do odpalenia po odpaleniu przejścia *xs* (rysunek 34c). Z własności sieci Petri'ego otrzymamy:

$$\tau(xs)+\alpha^{S}_{zs}\le\tau(zs)\le\tau(xs)+\beta^{S}_{zs}$$

dla analizowanej bramki, z definicji: $\alpha^{S}_{zs}=0$ i $\beta^{S}_{zs}=0$, zatem:

$$\tau(xs)\le\tau(zs)\le\tau(xs)$$

$$\tau(xs)=\tau(zs)$$

Ostatecznie:

$$\underline{\alpha_{xs}=\alpha_{zs},\ \beta_{xs}=\beta_{zs}}$$

Przejście *ze* staje się przygotowane do odpalenia po odpaleniu przejścia *xe* (rysunek 34e), zatem:

$$\tau(ye)+\alpha^{S}_{ze}\le\tau(ze)\le\tau(ye)+\beta^{S}_{ze}$$

z definicji analizowanej bramki wiemy (rozdział 4.5), że $\alpha^{S}_{ze}=0$ i $\beta^{S}_{ze}=0$, to:

$$\tau(ye)\le\tau(ze)\le\tau(ye)$$

$$\tau(ye)=\tau(ze)$$

Ostatecznie:

$$\underline{\alpha_{ye}=\alpha_{ze},\ \beta_{ye}=\beta_{ze}}$$

Wyznaczmy kolejne parametry.

Mając dane: $\tau(ye)\le\tau(xe)$ oraz $\tau(ye)\ge\alpha_{ye}$, $\tau(xe)\ge\alpha_{xe}$ oraz korzystając z przechodniości nierówności i tego, że wyznaczyliśmy α_{ye} otrzymamy:

$$\alpha_{ye}\le\tau(ye)\le\tau(xe)$$

Zatem, najwcześniejszy czas, w jakim może zostać odpalone przejście *xe* wynosi:

$$\underline{\alpha_{xe}=\alpha_{ye}}$$

Najpóźniejszy moment zakończenia zdarzenia *x* wyznaczymy z własności sieci Petri'ego:

$$\tau(xs)+\beta^{S}_{xe}\le\tau(xe)\ \wedge\ \tau(xe)\le\beta_{xe}$$

to:

$$\tau(xs)+\beta^{S}_{xe}\le\beta_{xe}$$

$$\tau(xs)\le\beta_{xe}-\beta^{S}_{xe}$$

wiedząc, że $\tau(xs) \leq \beta_{xs}$ możemy wyznaczyć β_{xe}:

$$\underline{\beta_{xe} = \beta_{xs} + \beta^{S}_{xe}}$$

Wyznaczymy jeszcze warunki dotyczące startu zdarzenia *y*. Z definicji rozważanego przypadku: $\tau(ys) \leq \tau(xs)$.

Ponieważ wiemy, że $\tau(xs) \leq \beta_{xs}$ oraz β_{xs} zostało wcześniej wyznaczone, zatem:

$$\tau(ys) \leq \tau(xs) \leq \beta_{xs}$$

to, korzystając z przechodniości nierówności, najpóźniejszy moment startu zdarzenia *y* wynosi:

$$\underline{\beta_{ys} = \beta_{xs}}$$

Ponieważ z systemu nierówności nie wynika wprost ograniczenie związane z najwcześniejszym momentem czasu, w którym może wystąpić zdarzenie *y*, przeprowadzimy pewne rozważanie.

Odpowiedzmy na pytanie, kiedy zdarzenie *y* będzie najbardziej przesunięte w czasie (w stronę malejących jednostek) od zdarzenia *x*, a zatem, jaka będzie wartość α_{ys}?

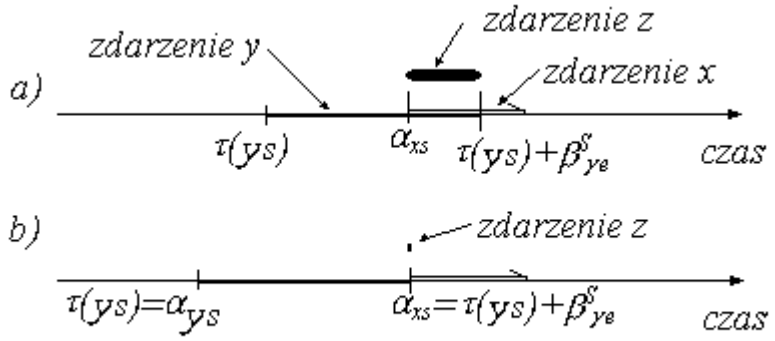
Zauważmy, że zdarzenie *x* jest jednoznacznie usytuowane w czasie (wyznaczyliśmy już α_{xs}, β_{xs}, α_{xe}, β_{xe}). Ponadto dana jest zależność: $\tau(xs) \leq \tau(ye)$ oraz wiemy, że $\alpha_{xs} \leq \tau(xs)$.

Jeśli będziemy coraz bardziej „przesuwać" w lewo na osi czasu start zdarzenia *y*, to dojdziemy do sytuacji, w której najwcześniejszy moment startu zdarzenia x, będzie najpóźniejszym momentem czasu zakończenia zdarzenia *y* (zdarzenie *x* rozpocznie się, w momencie zakończenia zdarzenia *y*). W tej hipotetycznej sytuacji zdarzenie *z* będzie zdarzeniem natychmiastowym. Taka sytuacja została przedstawiona na rysunku 35b.

Ponieważ zdarzenie *x* może najwcześniej rozpocząć się w momencie α_{xs}, a zdarzenie *y* może najdłużej trwać przez β^{S}_{ye}, zatem najwcześniejszy moment czasu, w którym może rozpocząć się zdarzenie *y*, by mogło doprowadzić do wystąpienia zdarzenia *z* wynosi:

$$\underline{\alpha_{ys} = \alpha_{xs} - \beta^{S}_{ye}}$$

c. b. d. o.



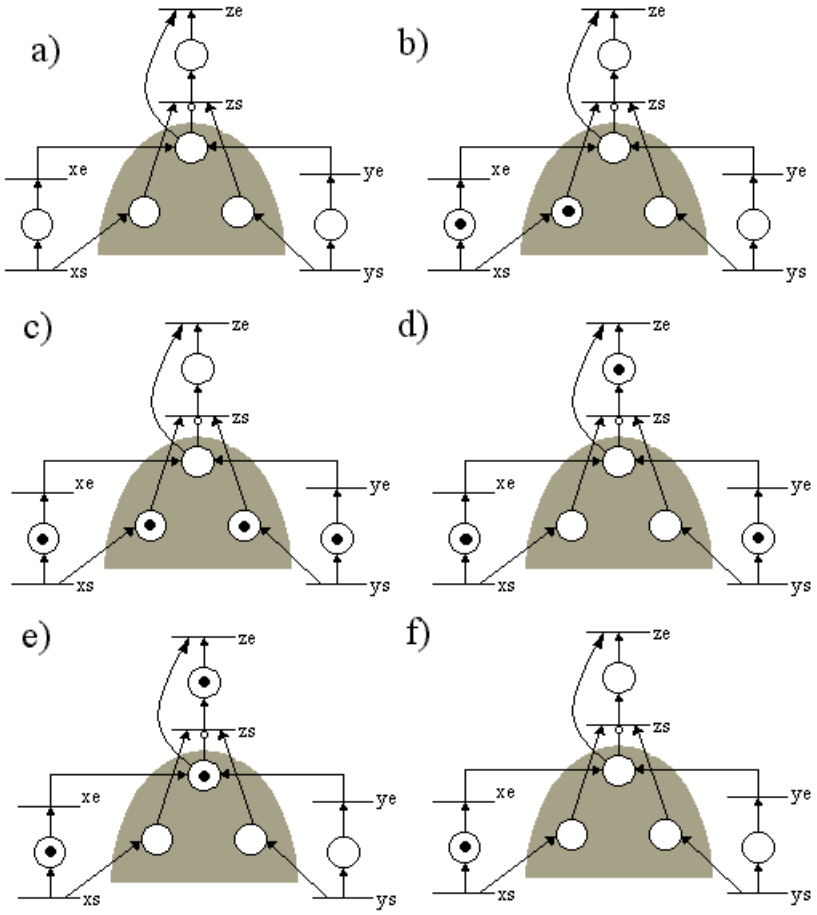
Rys. 35. Położenie zdarzeń *x* i *y* w czasie względem siebie

a) dowolny przypadek,

b) zdarzenie *y* maksymalnie „przesunięte w lewo” względem zdarzenia *x*.

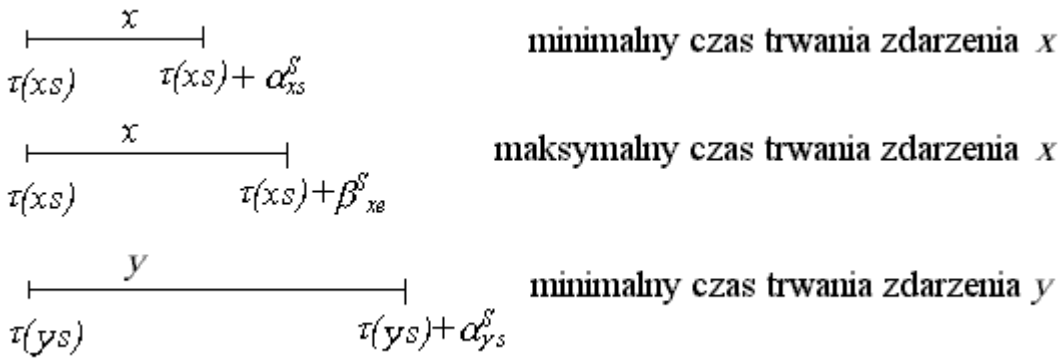
Ad 2. $\tau(xs) \le \tau(ys) \le \tau(ye) \le \tau(xe)$

Kolejność odpalania przejść dla tego przypadku ilustruje rysunek 36.



Rys. 36. Model bramki generalization AND ilustrujący odpalanie przejść w kolejności: a) przed odpaleniem przejść, b) *xs*, c) *ys*, d) *zs*, e) *ye* f) *ze*.

Początek i koniec zdarzenia *y* ma miejsce podczas trwania zdarzenia *x*. By taka sytuacja była możliwa maksymalny czas trwania zdarzenia *x* musi być co najmniej taki, jak minimalny czas trwania zdarzenia *y* (rys.37).



Rys. 37. Przykład parametrów czasowych dla zdarzeń *x* i *y* uniemożliwiających spełnienie relacji $\tau(xs) \le \tau(ys) \le \tau(ye) \le \tau(xe)$

Ten warunek jest spełniony tylko wtedy, gdy: $\alpha^S_{ye} \le \beta^S_{xe}$

Podobne rozumowanie możemy przeprowadzić analizując system nierówności dla omawianego przypadku.

Ponieważ z warunku wynika, że $\tau(xs) \le \tau(ys) \le \tau(ye) \le \tau(xe)$

Z własności sieci Petri'ego:

$$\tau(ys) + \alpha^S_{ye} \le \tau(ye), \ \tau(xs) + \beta^S_{xe} \ge \tau(xe)$$

zatem:

$$\tau(xs) \le \tau(ys) \ \ i \ \ \tau(ys) + \alpha^S_{ye} \le \tau(ye) \le \tau(xe) \ \ i \ \ \tau(xe) \le \tau(xs) + \beta^S_{xe}$$

korzystając z przechodniości nierówności, możemy zapisać:

$$\tau(xs) + \alpha^S_{ye} \le \tau(ye) \le \tau(xs) + \beta^S_{xe}$$

Ponieważ α^S_{ye} oraz β^S_{xe} są liczbami nieujemnymi to powyższa nierówność jest prawdziwa wtedy i tylko wtedy, gdy:

$$\underline{\alpha^S_{ye} \le \beta^S_{xe}}$$

Podobnie jak w przypadku poprzednim, z definicji bramki wiemy, że start i koniec zdarzenia *z* są równe (odpowiednio) startowi i końcowi zdarzenia *y*.

Zatem:

$$\underline{\alpha_{ys} = \alpha_{zs}, \ \beta_{ys} = \beta_{zs}} \text{ oraz } \underline{\alpha_{ye} = \alpha_{ze}, \ \beta_{ye} = \beta_{ze}}$$

Inny tok rozumowania został pokazany poniżej:

Przejście *zs* staje się przygotowane do odpalenia po odpaleniu przejścia *ys* (rysunek 36c). Z własności sieci Petri'ego otrzymamy:

$$\tau(ys)+\alpha^{S}_{zs}\leq\tau(zs)\leq\tau(ys)+\beta^{S}_{zs}$$

Dla analizowanej bramki $\alpha^{S}_{zs}=0$ i $\beta^{S}_{zs}=0$, zatem:

$$\tau(ys)\leq\tau(zs)\leq\tau(ys)$$

$$\tau(ys)=\tau(zs)$$

Ostatecznie:

$$\underline{\alpha_{ys}=\alpha_{zs},\ \beta_{ys}=\beta_{zs}}$$

Przejście *ze* staje się przygotowane do odpalenia po odpaleniu przejścia *ye* (rysunek 36e), zatem:

$$\tau(ye)+\alpha^{S}_{ze}\leq\tau(ze)\leq\tau(ye)+\beta^{S}_{ze}$$

dla analizowanej bramki: $\alpha^{S}_{ze}=0$ i $\beta^{S}_{ze}=0$, zatem:

$$\tau(ye)\leq\tau(ze)\leq\tau(ye)$$

$$\tau(ye)=\tau(ze)$$

Ostatecznie:

$$\underline{\alpha_{ye}=\alpha_{ze},\ \beta_{ye}=\beta_{ze}}$$

Wyznaczmy parametry określające momenty czasu startu i zakończenia zdarzenia *x*, czyli α_{xs}, β_{xs}, α_{xe}, β_{xe}.

Wiedząc, że $\tau(ye)\leq\tau(xe)$ oraz że $\tau(ye)\geq\alpha_{ye}$ i $\tau(xe)\geq\alpha_{xe}$, to korzystając z przechodniości nierówności i mając wyznaczone wcześniej α_{ye}, możemy wyznaczyć α_{xe}:

$$\underline{\alpha_{xe}=\alpha_{ye}}$$

Z własności sieci Petri'ego:

$$\tau(xs)+\beta^{S}_{xe}\leq\tau(xe)\ \wedge\ \tau(xe)\leq\beta_{xe}$$

to:

$$\tau(xs)+\beta^{S}_{xe}\leq\beta_{xe}$$

wiedząc, że $\tau(xs)\leq\beta_{xs}$ możemy wyznaczyć β_{xe}:

$$\underline{\beta_{xe}=\beta_{xs}+\beta^{S}_{xe}}$$

$\tau(ys)\leq\beta_{ys}$ oraz β_{ys} zostało wcześniej wyznaczone, zatem:

$$\tau(xs)\leq\tau(ys)\leq\beta_{ys}$$

Korzystając z przechodniości nierówności najpóźniejszy moment startu zdarzenia *x* wynosi:

$$\underline{\beta_{xs}=\beta_{ys}}$$

Przy wyznaczaniu najwcześniejszego momentu czasu, w którym może rozpocząć się zdarzenie *x* mamy identyczną sytuację, jak dla zdarzenia *y* w przypadku 1. Przeprowadzając analogiczne rozumowanie otrzymamy:

$$\underline{\alpha_{xs}=\alpha_{ys}-\beta^{S}_{xe}}$$

c. b. d. o.

Ad 3. $\tau(xs) \leq \tau(ys) \leq \tau(xe) \leq \tau(ye)$

Analogicznie jak dla przypadku 1.

Ad 4. $\tau(ys) \leq \tau(xs) \leq \tau(xe) \leq \tau(ye)$

Analogicznie jak dla przypadku 2.

5.5. Podsumowanie

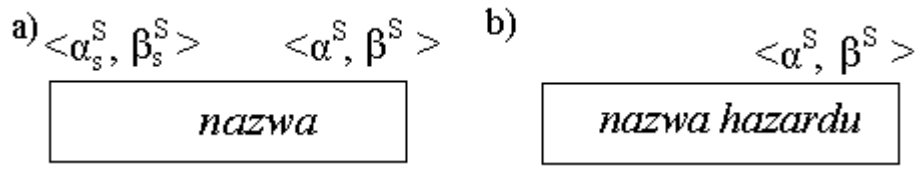
Korzystając ze wzorów (30) - (35) można przeprowadzić analizę drzewa niezdatności z zależnościami czasowymi. W jej wyniku otrzymamy minimalne zbiory przyczyn poszerzone o czas. Dalsze działania projektanta/projektantów czy też konstruktorów systemu polegają na ocenie uzyskanych wyników. Jeśli w danym systemie z określonymi parametrami czasu oraz określonymi przyczynami zdarzeń do sytuacji niebezpiecznej nie dojdzie, to możemy zakończyć proces analizy. W przeciwnym razie konieczne jest podjęcie działań mających na celu wyeliminowanie wskazanych w wyniku analizy sytuacji (koincydencji zdarzeń w czasie) mogących prowadzić do hazardu.

Algorytm analizy zostanie omówiony w kolejnych rozdziałach oraz w załączniku A.

6. Formalny zapis drzewa niezdatności w postaci grafu

6.1. Notacja zdarzeń

Sposób przedstawiania zdarzeń w drzewie niezdatności pokazuje rysunek 38.



Rys. 38. Notacja zdarzeń wraz z parametrami czasu w drzewie niezdatności
a) notacja dla zdarzeń równych hazardowi, b) notacja dla pozostałych zdarzeń

nazwa –nazwa zdarzenia; powinna w sposób precyzyjny i jednoznaczny opisywać dane zdarzenie

α^S, β^S – parametry statyczne związane odpowiednio z najwcześniejszym i najpóźniejszym momentem zakończenia zdarzenia (oznaczenia literowe mają analogiczne znaczenie, jak w symbolice stosowanej w czasowych sieciach Petri'ego [BD91], [BM82]); czas ten liczymy od momentu startu zdarzenia, a wyznaczamy go np. eksperymentalnie (mając zawór, możemy przeprowadzić pomiary czasu jego zamykania, a w sytuacji awaryjnej przyjąć wartość równą ∞, dla zbiornika wodnego możemy zmierzyć czas opróżniania, itp.), na podstawie parametrów fizycznych (np. wyznaczając czasy propagacji sygnału w łączu).

α^S_s, β^S_s – parametry statyczne związane odpowiednio z najwcześniejszym i najpóźniejszym momentem rozpoczęcia zdarzenia licząc chwili, gdy zostały spełnione wszystkie warunki do jego zajścia. Z definicji bram przyczynowych wiemy, że dla zdarzenia wyjściowego $\alpha^S_s=0$ i $\beta^S_s=0$. Parametry te mogą być różne od zera tylko dla drzewa złożonego z jednego zdarzenia – hazardu (takich drzew w praktyce nie analizujemy), ale możliwe jest wystąpienie takiej sytuacji dla bram uogólniających, iż rzeczywiście hazard będzie równy jakiemuś zdarzeniu będącemu liściem – wówczas te parametry również będą brane pod uwagę.

Według tego, co zostało przedstawione wcześniej, notację zdarzeń przedstawioną na rysunku 38 a) stosujemy tylko dla zdarzeń, które są równe hazardowi oraz dla hazardu, jeśli zbudujemy drzewo niezdatności tylko z jednego zdarzenia (hazardu).

Parametry czasowe są wymagane dla zdarzeń będących liśćmi oraz tych, które są zdarzeniami wyjściowymi bramek przyczynowych. Nie możemy podać parametrów czasowych dla zdarzeń wyjściowych w bramach uogólniających, gdyż wynikają one pośrednio ze zdarzeń, które są uogólniane.

Przedział $<\alpha^S, \beta^S>$ nie musi znajdować się koniecznie po prawej stronie zdarzenia, ale powinniśmy tak umieścić go na grafie, by nie było wątpliwości, jakiego zdarzenia dotyczy (może zostać zapisany również w prostokącie z nazwą zdarzenia).

6.2. Bramki – symbole graficzne z uwzględnieniem typu i parametrów czasowych

Sposób notacji bramek przedstawia rysunek 39.

C $<\alpha^S, \beta^S>$	- przyczynowa AND (z ang. *causal AND*); parametry statyczne w nawiasie dotyczą opóźnienia pomiędzy wystąpieniem przyczyn a ich skutkiem (por. z rozdz. 4.3.),
C $<\alpha^S,\beta^S>_1$ $<\alpha^S,\beta^S>_2$	- przyczynowa XOR (z ang. *causal XOR*); parametry statyczne w nawiasie dotyczą opóźnień pomiędzy zdarzeniem na wejściu i jego skutkiem, przy czym: 1 –dla zdarzenia na pierwszym (lewym) wejściu, 2 –dla zdarzenia na drugim (prawym) wejściu. Jeśli w bramie tej wykorzystujemy jedno wejście, to jednego z parametrów nie podajemy.
G	- uogólniająca AND (z ang. *generalization AND*)
G	-uogólniająca XOR (z ang. *generalization XOR)*

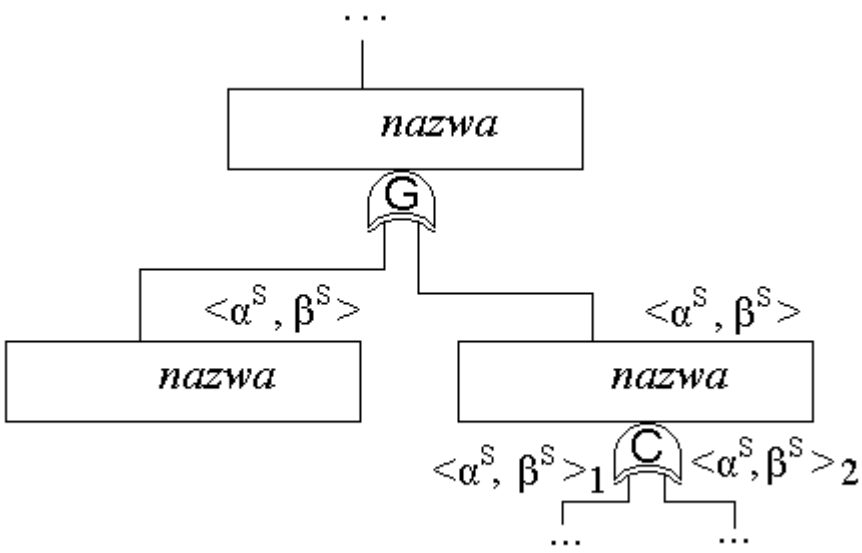
Rys. 39. Notacja bram

Przyjęta symbolika zachowuje zgodność z normami (por. z rys. 3), co do typu bramki i określa jednocześnie jej rodzaj poprzez literę. Dla bramek przyczynowych podajemy dodatkowo parametry czasowe.

W celu uniknięcia niejednoznaczności przedziały czasu dla bramki *causal AND* notujemy przy odpowiednim wyjściu, co zostanie pokazane w rozdziale 6.4.2.

6.3. Przykład notacji

Sposób przedstawiania bramek i zdarzeń w drzewie niezdatności przedstawia rysunek 40.



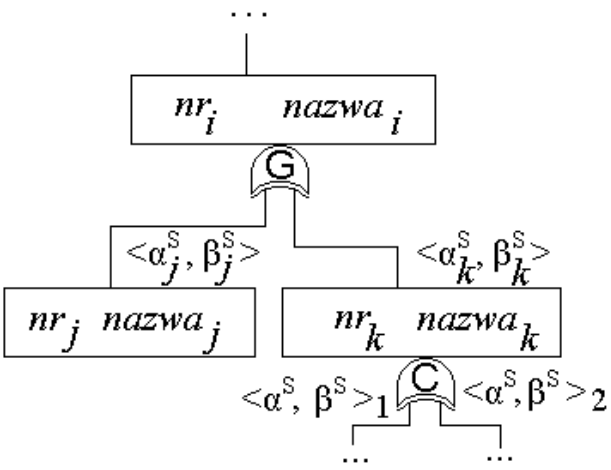
Rys. 40. Fragment drzewa niezdatności zgodny z przyjętą notacją

Zauważmy, że na rysunku 40 bramki przylegają do zdarzenia wyjściowego. Zdarzenia wejściowe łączymy liniami z wejściami bramki. Opóźnienia związane z bramką przyczynową XOR zapisujemy przy odpowiednich wejściach.

Ze względu na wygodę zarówno w zapisie, jak i analizie drzewa niezdatności często wprowadza się numerację zdarzeń i bramek w drzewie niezdatności.

Zatem, oprócz opisu słownego, zdarzenia będą miały również przypisany unikalny numer (nie jest to konieczne, ale ułatwia opis). Ze względu na to, iż bramki są jednowyjściowe przypiszemy bramkom takie same numery jak zdarzeniom wyjściowym.

Taki sposób przedstawiania drzewa niezdatności pokazuje rysunek 41.

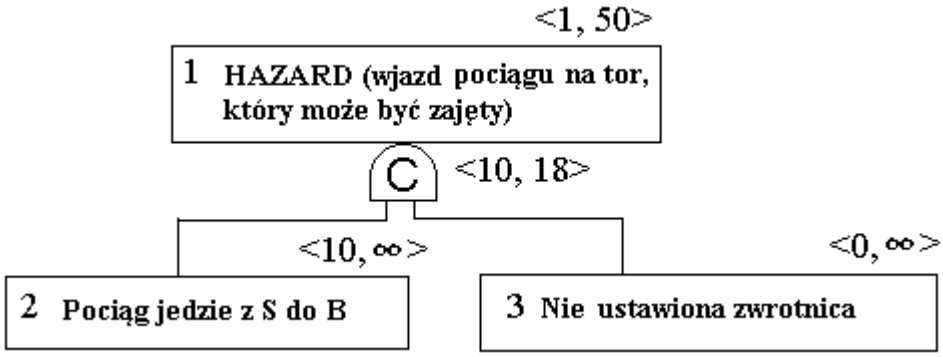


Rys. 41. Fragment drzewa niezdatności z numeracją bramek i zdarzeń

Z godnie z przyjętą zasadą numeracji, bramki na rysunku 41 mają:

- uogólniająca XOR - nr_i,
- przyczynowa XOR - nr_k.

Formalny zapis drzewa niezdatności z rysunku 10 przedstawia rysunek 42 (por. z siecią Petri'ego z rysunku 21). Bramka *causal AND* ma domyślnie przypisany numer 1 (taki, jak zdarzenie wyjściowe).



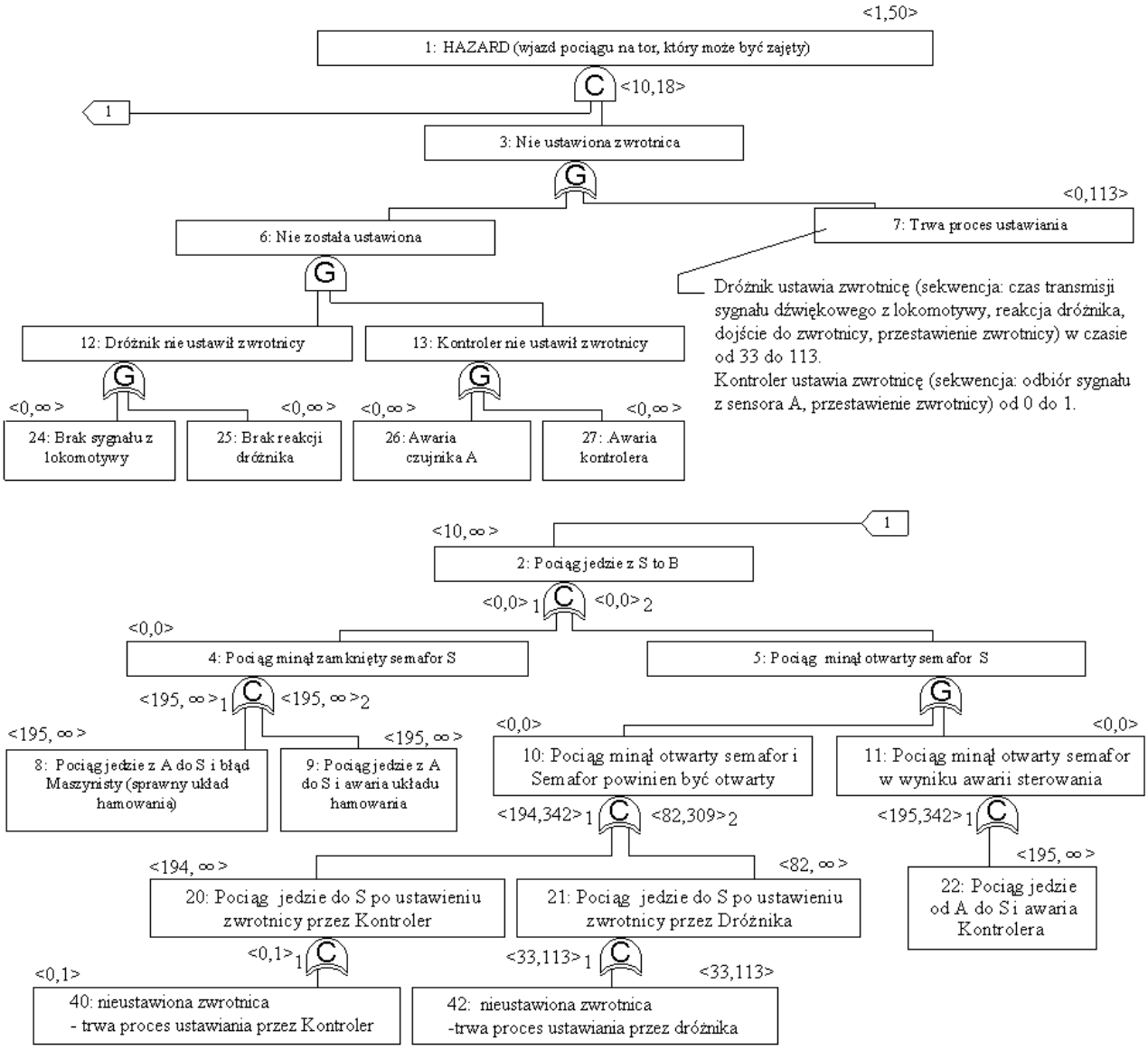
Rys. 42. Formalny zapis drzewa niezdatności z rys. 10
po uwzględnieniu parametrów czasowych

Przedstawione rozwiązanie nie narzuca wymogów co do np. prawej lub lewej strony umieszczania przedziałów czasu przy zdarzeniu, czy też bramie. Jak to zostało jednak wcześniej zaznaczone, powinno to być robione w sposób pozwalający

jednoznacznie przypisać przedział czasu do zdarzenia, czy też bramki. Ta uwaga jest szczególnie istotna w przypadku dużych drzew niezdatności. Oczywiście takiego problemu nie ma, gdy analiza jest wspomagana komputerem, a przez to zapis drzewa, czy to w formie pliku tekstowego, czy w formie graficznej wymusza jednoznaczne przypisanie parametrów czasowych zdarzeniom i bramkom.

6.4. Drzewo niezdatności dla rozjazdu kolejowego

Drzewo błędów dla rozjazdu kolejowego z parametrami opisanymi w sekcji 2.4.2 przedstawia rysunek 43.



Rys. 43. Drzewo błędów z zależnościami czasowymi dla sterowania rozjazdem kolejowym.

7. Algorytm *INES*

7.1. Przyjęta metodologia

Pod uwagę należy wziąć dwie możliwości. Pierwsza, to zbudowanie jednego algorytmu, dzięki któremu otrzymamy zbiór rozwiązań – minimalne zbiory przyczyn ze zdarzeniami z określonymi parametrami czasowymi. Drugie rozwiązanie, to przeprowadzenie analizy w dwóch etapach: reorganizacja struktury drzewa do postaci nazwijmy ją „drzewa przypadków", a następnie przeprowadzenie analizy.

Pierwszy sposób został zaprezentowany w [MS02], choć nie obejmował analizy bramek G_AND. W niniejszej pracy zostanie pokazany algorytm analizy FT z zależnościami czasowymi zgodny z drugim rozwiązaniem. Takie podejście:

- Ułatwi modyfikacje algorytmu, jeśli będzie potrzeba rozbudowania go o np. wielowejściowe bramki uogólniające, wtedy zmianom będzie podlegał tylko pierwszy etap – obliczenia pozostaną niezmienione.
- Nie spowoduje znacznego wzrostu złożoności obliczeniowej, gdyż w pierwszym etapie będzie poddane analizie drzewo binarne, a zatem będzie mogła zostać przeprowadzona w „rozsądnym" czasie.
- Pozwoli na proste wyznaczenie parametrów dla bramek uogólniających AND.
- Umożliwi zbudowanie uproszczonej wersji analizy, tylko momentów czasu dotyczących rozpoczęcia zdarzeń (uproszczenie przypadków dla bramki uogólniającej AND do dwóch).

Mimo iż możliwe jest przeprowadzenie analizy parametrów statycznych już w pierwszym etapie analizy (podczas reorganizacji drzewa niezdatności), to w pracy zostanie pokazany algorytm, który nie stosuje takiego podejścia. Takie rozwiązanie pozwoliłoby wprawdzie na wyeliminowanie pewnych zdarzeń, które do wystąpienia hazardu nie mogłyby prowadzić, ale uniemożliwiłoby np. uzyskanie wyników porównawczych metodą klasyczną (bez zależności czasowych). Ponadto sama struktura drzewa po reorganizacji niesie z sobą również wiele informacji.

7.2. Drzewo niezdatności, drzewo przypadków oraz drzewo wyników analizy

Ze względu na przyjęte rozwiązanie, w procesie analizy występują trzy struktury:

a) drzewo błędów (ang. *FT –fault tree*),

b) drzewo przypadków (ang. *CT – case tree*),

c) drzewo wyników (ang. *RT – result tree*).

Ad a)

Formalny zapis drzewa niezdatności w formie grafu został przedstawiony w rozdziale 6. Analiza komputerowa oraz implementacja omawianego algorytmu wymaga jednak reprezentacji drzewa w postaci symbolicznej – nie przechowujemy w pamięci komputera „rysunku" drzewa, ale jego symboliczny zapis. Przykład takiej notacji zawiera załącznik A.

Ad b)

Drzewo przypadków powstaje po przekształceniu drzewa niezdatności – „eliminacja" bramek uogólniających XOR oraz szacowanie parametrów dla bramek uogólniających AND. Zasadę przekształcenia opisano w rozdziale 7.3.1, a uszczegółowienie algorytmów zawiera załącznik.

Ad c)

Drzewo wyników to zbiór elementów, z których każdy odpowiada jednemu zdarzeniu wraz z wyznaczonymi dla niego parametrami czasu oraz informacją o położeniu w drzewie RT (szczegóły w rozdziale 7.3.2 oraz A.6). Z punktu widzenia analizy - RT zawiera wszystkie minimalne zbiory przyczyn.

Przyjęty zapis wyników pozwala na odtworzenie procesu budowy drzewa, a w szczególnych przypadkach również na śledzenie przebiegu analizy i/lub zatrzymanie oraz wznowienie jej w dowolnym momencie (możliwość „wglądu" w wyniki analizy na dowolnym jej etapie). W przyszłości będzie możliwe uzyskanie reprezentacji graficznej uzyskanych wyników (w narzędziu [Sk05] nie została zaimplementowana taka funkcja) w postaci jak na rysunku 46..

7.3. Algorytm główny (INES)

Algorytm główny obejmuje dwa kroki:

K1: Zbuduj drzewo przypadków (CT).

K2: Wykonaj obliczenia dla drzewa CT (zbuduj RT).

Po uzyskaniu wyników w postaci drzewa RT, jesteśmy w stanie odpowiedzieć na pytanie czy hazard w danym systemie może wystąpić oraz jakie zdarzenie muszą się pojawić i w jakich przedziałach czasu. Ta wiedza pozwala nam na takie postępowanie,

by do hazardu nie dopuścić lub zminimalizować możliwość jego wystąpienia. Przykład analizy zawiera załącznik A.

7.3.1. Budowa drzewa przypadków - algorytm CT

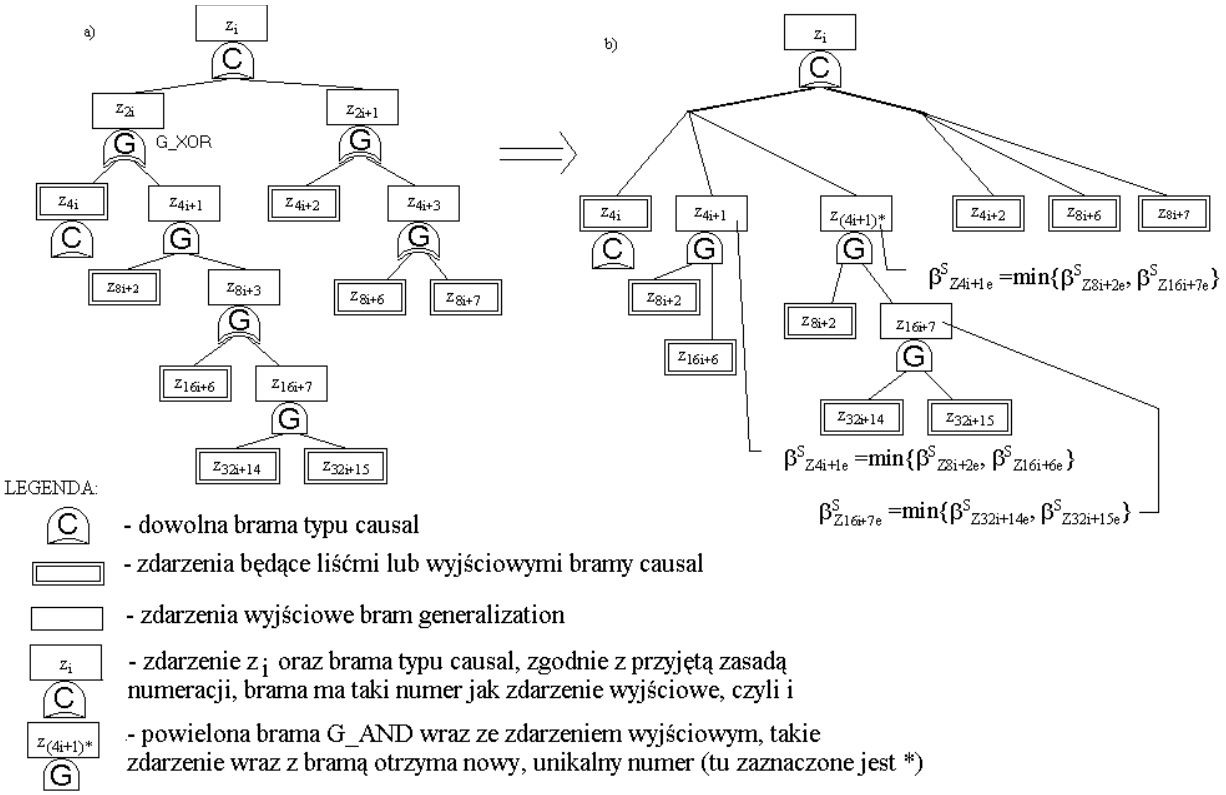
Celem tego etapu jest wyeliminowanie bramek uogólniających XOR oraz oszacowanie parametrów dla bramek uogólniających AND.

W trakcie procesu konstrukcji drzewa przypadków tworzone są również „wirtualne" bramki uogólniające AND. Powstają one przez „powielenie" oryginalnej bramki uogólniającej AND wtedy, gdy do jednego lub obu jej wejść dołączona jest bramka uogólniająca XOR lub struktura kilku połączonych ze sobą bramek - powielenie ma miejsce dla każdej bramki uogólniającej XOR z takiej struktury.

Zasadę budowy drzewa CT pokazuje rysunek 44, a powielenie ma miejsce dla bramki z_{4i+1} (zgodnie z przyjętą zasadą numeracji, bramka ma taki numer, jak zdarzenie na jej wyjściu).

Rysunek 44 b) zawiera bramkę z_{4i+1} oraz $z_{(4i+1)^*}$ - bramka powielona. Bramka z_{4i+1} ma dołączone do swojego prawego wejścia zdarzenie z_{16i+6} (jedno ze zdarzeń wejściowych bramki z_{8i+3}), a bramka $z_{(4i+1)^*}$ - zdarzenie z_{16i+7}.

Jeśli bramka z_{16i+7} byłaby również uogólniająca XOR, to powstałyby dwie powielone bramki: $z_{(4i+1)^*}$ i $z_{(4i+1)^{**}}$ z dołączonymi do prawego wejścia zdarzeniami, odpowiednio: z_{32i+14}, z_{32i+15}.



Rys. 44. Przykład reorganizacji a) drzewa niezdatności w b) drzewo przypadków

Zgodnie ze wzorami (34), w bramach G_XOR rozważamy dwa przypadki. W każdym z nich zdarzenie wyjściowe jest równe wejściowemu. Ponieważ jednak dla bramki przyczynowej potrzebujemy parametrów statycznych to, zgodnie z równaniami (34), sięgamy niejako „w głąb" drzewa. I tak, na przykład: jeśli przyjmiemy, że brama *i* na rysunku 44 jest typu *causal XOR*, to mając dane α^S_{d2} oraz α_{zis}, β_{zis} powinniśmy wyznaczyć α_{z2i+1s}, β_{z2i+1s}. Zdarzenie *z2i+1* jest jednak w bramie G_XOR, zatem zgodnie ze wzorem (34) mamy dwa przypadki: $\alpha_{z2i+1s}=\alpha_{z4i+2s}$, $\beta_{z2i+1s}=\beta_{z4i+2s}$ (zapis ten oznacza, że zdarzenie *z2i+1* jest w tym przypadku równe zdarzeniu *z4i+2*) lub $\alpha_{z2i+1s}=\alpha_{z4i+3s}$, $\beta_{z2i+1s}=\beta_{z4i+3s}$.

Przeprowadzając analogiczne rozumowanie dla zdarzenia *z4i+3*, które jest również zdarzeniem wyjściowym bramki uogólniającej XOR, otrzymamy kolejne dwa przypadki: $\alpha_{z2i+1s}=\alpha_{z4i+3s}=\alpha_{z8i+6s}$, $\beta_{z2i+1s}=\beta_{z4i+3s}=\beta_{z8i+6s}$ lub $\alpha_{z2i+1s}=\alpha_{z4i+3s}=\alpha_{z8i+7s}$, $\beta_{z2i+1s}=\beta_{z4i+3s}=\beta_{z8i+7s}$.

Struktura przedstawiona na rysunku 44b nie stanowi już drzewa niezdatności, ale „drzewo przypadków" do analizy. Na przykład, do bramki przyczynowej o numerze

i dowiązanych jest 6 zdarzeń (po trzy do lewego i prawego wejścia). Taki zapis graficzny należy rozumieć:

a) jeśli brama przyczynowa jest typu AND: „należy rozważyć 9 przypadków, korzystając ze wzoru (33) dla każdej pary zdarzeń: $\{z_{4i}, z_{4i+2}\}$, $\{z_{4i}, z_{8i+6}\}$, $\{z_{4i}, z_{8i+7}\}$, $\{z_{4i+1}, z_{4i+2}\}$, $\{z_{4i+1}, z_{8i+6}\}$, $\{z_{4i+1}, z_{8i+7}\}$, $\{z_{(4i+1)*}, z_{4i+2}\}$, $\{z_{(4i+1)*}, z_{8i+6}\}$, $\{z_{(4i+1)*}, z_{8i+7}\}$",

b) jeśli bramka przyczynowa jest typu XOR: „należy rozważyć 6 przypadków, dla każdego zdarzenia (wzór (32)): $z_{4i}, z_{4i+1}, z_{(4i+1)*}, z_{4i+2}, z_{8i+6}, z_{8i+7}$".

Nazwijmy zdarzenie wraz z bramką lub samo zdarzenie (w przypadku liści) węzłem. Zatem węzły wewnętrzne w drzewie FT, to zdarzenia z bramami, a węzły zewnętrzne to zdarzenia.

Poniżej został przedstawiony algorytm w postaci ogólnej. Szczegóły dotyczące numeracji, definicja węzła FT oraz uszczegółowienie algorytm znajdują się w załączniku A.

Poniższą procedurę wywołujemy dla zdarzenia o numerze 1 (hazardu), a wywołanie ma postać: *drzewo(1, LISTA)*.

procedure drzewo(i; var LISTA: tlista)

{***i*** – numer węzła, przekazywany do procedury; ***LISTA*** – lista następników zwracana przez procedurę, w szczególności może być jeden, ***tlista*** – typ zmiennej *LISTA*; w zależności od implementacji może to być wskaźnik, tablica itp.}

LL, LP: tlista*; {LL, LP* – zmienne typu *tlista*; służą do przechowywania odpowiednio, lewych i prawych następników węzła *i*}

r1, r2, nr: typ_całkowitoliczbowy; {Zmienne pomocnicze}

begin

K1: {Jeśli lewy następnik jest węzłem zewnętrznym, to zapisz go na listę *LISTA*, a jeśli wewnętrznym, to wywołaj z jego numerem procedurę *drzewo.}*

if typ(2*i)=LISC then LISTA←2*i {Zapis *LISTA←2*i* oznacza dopisanie do listy liczby *2*i* Ponieważ lista była pusta, więc znajdzie się na niej jeden element.*}*

else if typ(2*i) <> NULL then drzewo (2*i, LL);

{Gdzie:

*2*i* – numer lewego następnika,

*typ(2*i)* – funkcja, która zwraca typ węzła o podanym numerze, w tym przypadku o numerze *2*i*; przy tym, typ węzła może przyjmować wartości: *LISC* – dla węzła będącego liściem (nie zawierającego bramy), *NULL* – jeśli nie ma następnika (żadnego zdarzenia na wejściu bramy - taka sytuacja może mieć miejsce dla bram XOR), *G_XOR*, *G_AND*, *C_XOR* lub *C_AND* – dla węzłów z bramami.}

K2: {Jeśli prawy następnik jest węzłem zewnętrznym, to zapisz go na listę *LISTA*, a jeśli wewnętrznym, to wywołaj z jego numerem procedurę *drzewo*.}

if typ(2*i+1)=LISC then LISTA←2*i+1

else if typ(2*i+1) <> NULL then drzewo (2*i+1, LL);

K3: {Zapisz węzeł, ustalając jego elementy na podstawie list *LL* i *LP* oraz według typu bramki w węźle.}

K3a: ***if typ(i) = G_XOR then LISTA ←LL ∪ LP***

{Jeśli bramka jest uogólniająca XOR, to zapisz na listę *LISTA* sumę elementów z list *LL* i *LP*}

K3b: ***if typ(i) in [C_XOR, C_AND] then***

{Jeśli bramka jest przyczynowa XOR lub przyczynowa AND, to …}

begin

zapisz_wezel_do_drzewa_CT(i, i, LL, LP); {zapisz węzeł.}

{Gdzie:

i – numer węzła w drzewie CT (zawsze będzie taki, jak numer w drzewie FT za wyjątkiem numerów powielonych bram),

i – numer węzła w drzewie FT,

LL oraz *LP.}*

LISTA ←i; {Zapisz numer węzła *i* na listę *LISTA*}

end;

K3c: ***if typ(i) = G_AND then*** {Jeśli bramka jest uogólniająca AND, to …}

begin

nr:=i;

```
for  r1:=1 to rozmiar(LL) do
{dla każdego elementu z listy LL (funkcja rozmiar zwraca
liczbę elementów na liście LL) i …}
   for r2:=1 to rozmiar(LP) do
   {dla każdego elementu z listy LP rób:}
      begin
         if not ((r1=1) and (r2=1)) then
            nr:=replikuj_G_AND;  {Replikuj bramę dla
                  każdej pary elementów (oprócz pierwszej)
                  z list LL i LP. Funkcja replikuj_G_AND
                  zwraca numer bramy po replikacji}
         {By uniknąć wprowadzania zmiennych
         pomocniczych przyjmijmy, że zapis LL[r1] oznacza
         listę jednoelementową zbudowaną z elementu o
         indeksie r1 listy LL. Analogicznie LP[r2]. Wówczas
         kolejna operacja będzie następująca:}
         zapisz_wezel_do_drzewa_CT(nr, i, LL[r1], LP[r2]);
         LISTA←nr;  {Dopisanie do listy LISTA numeru
                  węzła nr.}
      end;
end;
```

Rozpoczynając od zdarzenia numer 1 (hazardu) i postępując zgodnie z przedstawioną procedurą dojdziemy (poprzez wywołania rekurencyjne) do liści, a następnie zbudujemy drzewo CT.

7.3.2. Wyznaczania minimalnych zbiorów przyczyn - algorytm RT

Algorytm wykonywania obliczeń bazuje na algorytmie przeszukiwania przestrzeni stanów (algorytmy „przechodzenia grafu w głąb" oraz „przechodzenia grafu wszerz" zostały opisane na przykład w [BDR96]).

Węzeł drzewa RT posiada: unikalny numer, informację o numerze zdarzenia w drzewie FT, parametry dynamiczne momentu startu oraz momentu zakończenia tego zdarzenia, informację o tworzeniu struktury AND z innymi węzłami oraz numer węzła rodzica. Dokładny opis węzła można znaleźć w załączniku, rozdział A.3. Przykład zapisu węzła zaczerpnięty z załącznika:

```
/3 6 -∞ -10 0 ∞ 1 1 /
```

Znaczenie poszczególnych liczb jest następujące:

3 – numer węzła (trzeci wyznaczony węzeł),

6 – zdarzenie numer 6 w drzewie FT,

$-\infty$ – α_{2s} (najwcześniejszy moment startu zdarzenia),

-10 – β_{2s} (najpóźniejszy moment startu),

0 – α_{2e} (najwcześniejszy moment zakończenia zdarzenia),

∞ – β_{2e} (najpóźniejszy moment zakończenia zdarzenia),

1 – numer struktury AND (wszystkie węzły z numerem 1 tworzą razem jeden przypadek; na rysunku 46 węzły tworzące struktury AND na danym etapie analizy są ujęte w jeden prostokąt),

1 – numer węzła rodzica (oznacza to, że węzeł numer 3 został wyliczony z węzła numer 1, inaczej – jest jego potomkiem). Ponieważ węzeł numer 1 jest związany z bramą przyczynową AND, to węzeł numer 3 stanowi jeden z przypadków wyznaczonych ze wzoru (33).

Obliczenia wykonywane są według poniższego algorytmu.

K1: Wprowadź węzeł startowy na listę *LB*

{Węzeł startowy wyznaczamy dla hazardu ze wzorów: (30), (31). W szczególnej sytuacji, jeśli pierwszą bramą jest brama uogólniająca w kroku tym możemy otrzymać kilka węzłów zapisanych na listę *LB*.}

K2. Dopóki lista *LB* nie jest pusta wykonuj:

K2.1 ***wezel ← LB[1]*** {Wprowadzenie do zmiennej pomocniczej pierwszego elementu z listy *LB*}

K2.2. ***usuń (wezel, LB)*** {Usunięcie pierwszego elementu z listy *LB*. Elementy usunięte z list *LB* zapisywane są na liście *LS*}

K2.3. ***wprowadz (LB, potomki(wezel))*** {Wprowadzenie na listę *LB* węzłów potomnych węzła *wezel*. Węzły potomne są wyznaczane w zależności od typu bramy w węźle *wezel* ze wzorów: (32), (33), (34) lub (35). Jeśli *wezel* związany jest z liściem, to nie ma potomków.}

Historia obliczeń jest przechowywana na liście *LS*. Po zakończeniu analizy lista *LB* jest pusta, a lista *LS* przechowuje wyniki analizy.

Obliczenia możemy przeprowadzić do końca, czyli do uzyskania wszystkich możliwych przypadków lub przerwać w określonym momencie – wówczas na podstawie list *LS* oraz *LB* możemy również określić minimalne zbiory przyczyn dla przeanalizowanego fragmentu drzewa.

W każdym kroku obliczeń do wyznaczenia węzłów potomnych potrzebujemy tylko:

1. parametrów statycznych (dane zawarte w opisie FT),

2. momentów czasu rozpoczęcia i zakończenia zdarzenia dla którego będziemy szukać przyczyn (danych zawartych w węźle).

Możliwe jest zatem zbudowanie algorytmu pozwalającego na równoległe prowadzenie obliczeń. Jeśli, na przykład: dla węzła numer 1 otrzymamy cztery węzły potomne, to możemy każdy z nich wraz z opisem FT „przekazać” do analizy innemu

komputerowi. W przypadku sieci Petri'ego modelującej drzewo niezdatności, taka operacja nie jest możliwe. Jeśli prowadzimy analizę PN modelującą FT, to w każdym jej kroku musimy „widzieć" całą sieć Petri'ego.

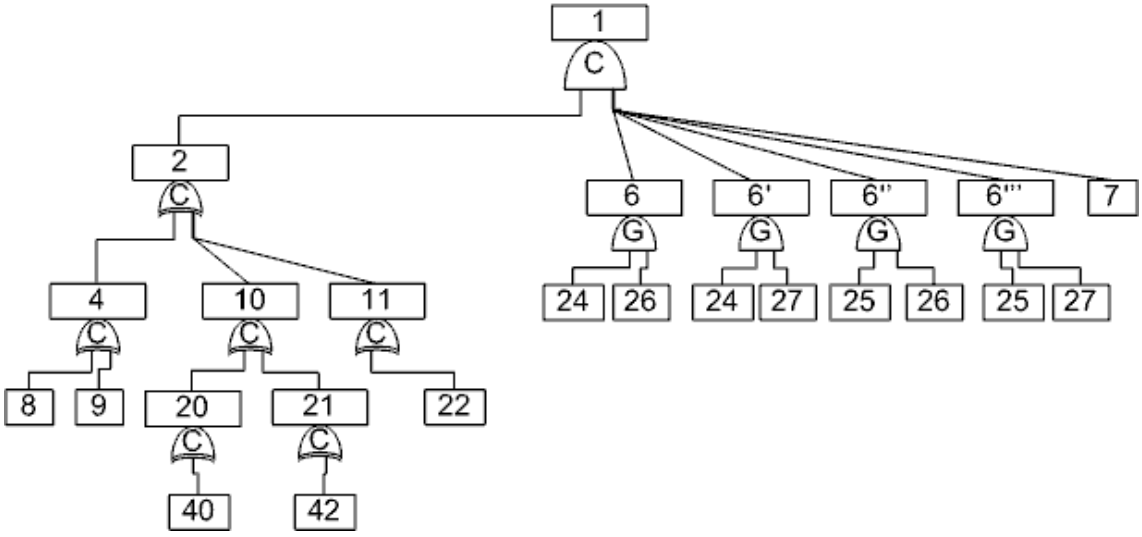
Zagadnienia związane ze złożonością obliczeniową prezentowanej metody zostaną omówione w dalszej części opracowania.

7.4. Przykład - analiza drzewa niezdatności systemu sterowania rozjazdem kolejowym

Analizę przeprowadzamy w dwóch etapach.

Pierwszy z nich będzie polegał na przekształceniu FT w CT. Jak to zostało pokazane w poprzednim rozdziale, eliminujemy bramki uogólniające XOR oraz szacujemy parametry dla bramek uogólniających AND.

Wynik tej operacji w postaci drzewa CT przedstawia rysunek 45.



Rys. 45. Przykład CT dla drzewa niezdatności z rys. 43

Drugi etap, to analiza drzewa przypadków.

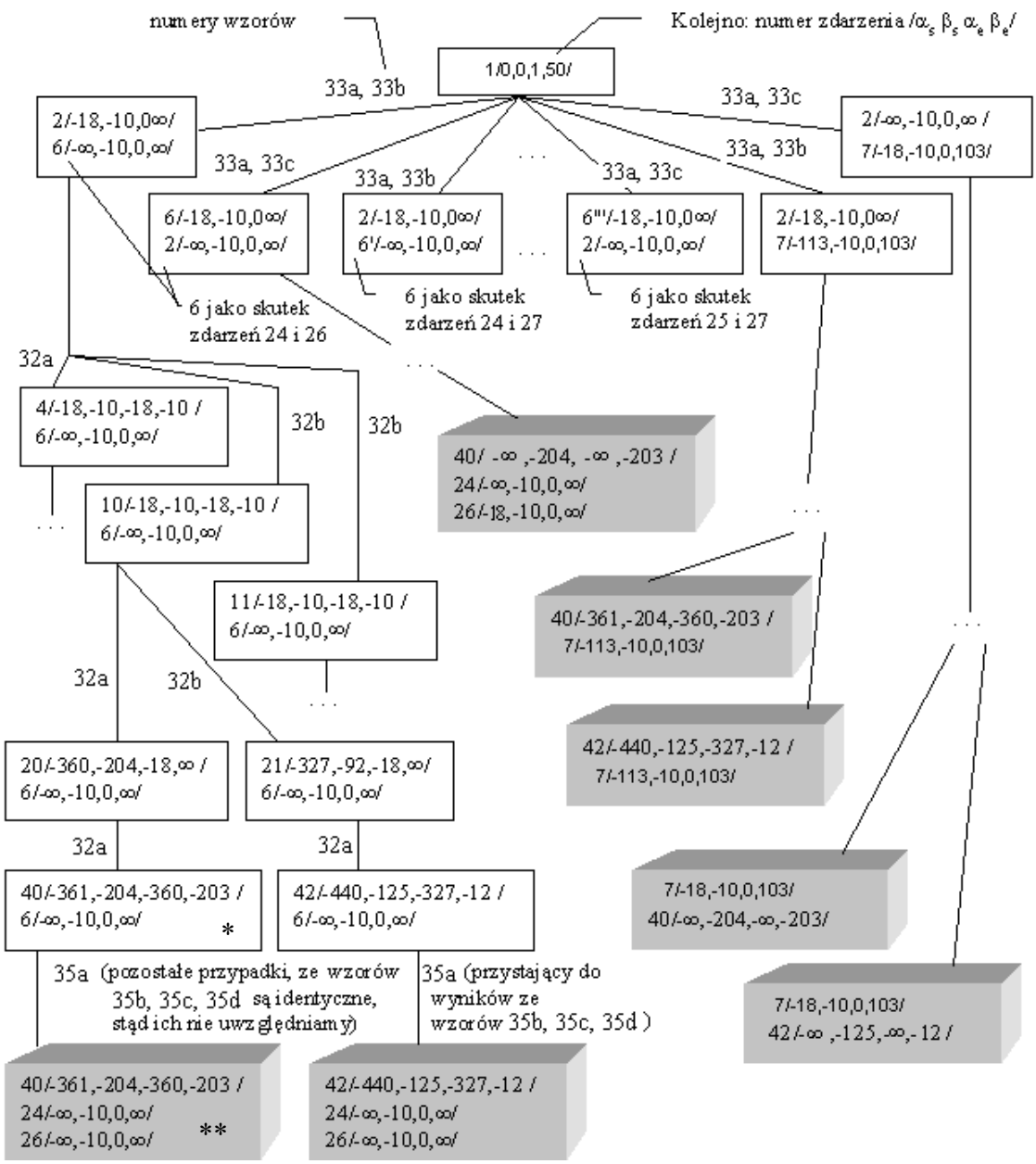
Najpierw wyznaczamy parametry hazardu ze wzorów (30) i (31), a następnie przeprowadzamy obliczenia zgodnie ze wzorami (32) – (35) i uzyskujemy minimalne zbiory przyczyn. Wystąpienie wszystkich zdarzeń z takiego zbioru jest warunkiem

koniecznym do wystąpienia hazardu. W odróżnieniu od klasycznej analizy drzew niezdatności, w prezentowanej metodzie wyniki zawierają również relacje czasowe pomiędzy zdarzeniami.

Fragment uzyskanych wyników dla CT z rys. 45 przedstawia rys. 46. Każdy prostokąt na tym rysunku zawiera minimalny zbiór przyczyn prowadzących do hazardu. W białych prostokątach znajdują się zbiory zawierające elementy, dla których będzie prowadzona dalsza analiza. Po skończeniu obliczeń otrzymamy minimalne zbiory przyczyn złożone ze zdarzeń będących liśćmi (wyróżnione prostokąty).

I tak na przykład prostokąt zaznaczony na rysunku 46 poprzez * stanowi minimalny zbiór przyczyn złożony z dwóch zdarzeń o numerach 40 i 6. Ponieważ zdarzenie numer 6 nie jest liściem, to w procesie analizy zostaną wyznaczone zależności czasowe pomiędzy przyczynami do niego prowadzącymi i otrzymamy zbiór zaznaczony poprzez **.

Rysunek 46 powstał poprzez przetworzenie drzewa RT - zgrupowanie węzłów tworzących struktury AND w prostokątach i narysowanie połączeń pomiędzy nimi.


numery wzorów
Kolejno: numer zdarzenia /α_s β_s α_e β_e/
1/0,0,1,50/
33a, 33b
33a, 33c
2/-18,-10,0∞/
6/-∞,-10,0,∞/
2/-∞,-10,0,∞ /
7/-18,-10,0,103/
6/-18,-10,0∞/
2/-∞,-10,0,∞/
2/-18,-10,0∞/
6'/-∞,-10,0,∞/
6'''/-18,-10,0∞/
2/-∞,-10,0,∞/
2/-18,-10,0∞/
7/-113,-10,0,103/
6 jako skutek zdarzeń 24 i 26
6 jako skutek zdarzeń 24 i 27
6 jako skutek zdarzeń 25 i 27
32a
32b
4/-18,-10,-18,-10 /
6/-∞,-10,0,∞/
40/ -∞ ,-204, -∞ ,-203 /
24/-∞,-10,0,∞/
26/-18,-10,0,∞/
10/-18,-10,-18,-10 /
6/-∞,-10,0,∞/
11/-18,-10,-18,-10 /
6/-∞,-10,0,∞/
40/-361,-204,-360,-203 /
7/-113,-10,0,103/
42/-440,-125,-327,-12 /
7/-113,-10,0,103/
20/-360,-204,-18,∞ /
6/-∞,-10,0,∞/
21/-327,-92,-18,∞/
6/-∞,-10,0,∞/
40/-361,-204,-360,-203 /
6/-∞,-10,0,∞/ *
42/-440,-125,-327,-12 /
6/-∞,-10,0,∞/
7/-18,-10,0,103/
40/-∞,-204,-∞,-203/
35a (pozostałe przypadki, ze wzorów 35b, 35c, 35d są identyczne, stąd ich nie uwzględniamy)
35a (przystający do wyników ze wzorów 35b, 35c, 35d)
7/-18,-10,0,103/
42/-∞ ,-125,-∞,-12 /
40/-361,-204,-360,-203 /
24/-∞,-10,0,∞/
26/-∞,-10,0,∞/ **
42/-440,-125,-327,-12 /
24/-∞,-10,0,∞/
26/-∞,-10,0,∞/

Rys. 46. Fragment uzyskanych wyników w postaci drzewa RT

7.5. Oszacowanie złożoności obliczeniowej

Dokładne wyznaczenie złożoności obliczeniowej algorytmu jest możliwe dla jego określonej implementacji. Ponieważ prezentowany w rozdziale 7 algorytm oraz jego uszczegółowienie w załączniku A jest na niższym poziomie szczegółowości (wyższym poziomie abstrakcji) niż implementacja, wyznaczenie złożoności obliczeniowej zostało nazwane „szacowaniem".

Złożoność obliczeniowa:

a) tworzenie drzewa CT dla drzewa niezdatności o m bramach może mieć w najgorszym przypadku złożoność wykładniczą (można wskazać taki przykład), ale można przyjąć założenie, iż w praktyce jest $o(m^2)$,

b) tworzenia drzewa wyników (RT) dla CT o n węzłach w najgorszym przypadku jest $\circ(2^{2n})$.

Szacowanie złożoności wraz z uzasadnieniem oraz omówienie najgorszych przypadków znajduje się w załączniku rozdziały A.5.4, A.6.4.

W praktyce, taka struktura drzewa niezdatności, jak dla pesymistycznego przypadku (drzewo tylko z bramami uogólniającymi AND oraz XOR w specyficznym układzie) nie występuje (dla takiego drzewa nie byłaby możliwa również analiza klasyczna ze względu na zbyt dużą liczbę minimalnych zbiorów przyczyn).

Ponadto, często przy wyznaczaniu kolejnych węzłów drzewa RT możemy korzystać z uzyskanych wcześniej wyników. Jeśli parametry dla wyliczonego następnika są identyczne, jak dla przypadku już wcześniej analizowanego, to możemy skorzystać z wcześniej przeprowadzonych obliczeń. Takie przypadki zostały zaznaczone w wynikach uzyskanych dla przykładu rozjazdu kolejowego - załącznik, rozdział A.6.5.. Również dla przykładów: system palnika gazowego [MS00], system sterowania rogatkami na przejeździe kolejowym [MS03] - zawsze część obliczeń ulegała uproszczeniu.

Identyczne węzły możemy uzyskać także przy wyznaczaniu następników i tak dla bramy:

- przyczynowej AND – gdy wyliczone dla zdarzeń *x* i *y* ze wzoru (33b) i (33c) wartości α_{xs} β_{xs}, α_{xe} β_{xe}, α_{ys} β_{ys}, α_{ye} β_{ye}, są sobie równe,
- uogólniającej AND – gdy między wyliczonymi wartościami, ze wzorów: (35a), (35b), (35c) lub (35d) zachodzi równość. Najczęściej redukcji może ulec jeden lub dwa węzły.

Jeśli trafilibyśmy jednak na najgorszy (pesymistyczny) przypadek to możemy, postępując zgodnie z zasadą „dziel i zwyciężaj" rozdzielić go na „podprzypadki" i przekazać do analizy kilku komputerom.

Rozważania dotyczące tego zagadnienia znajdują się w rozdziale A.6.4.

7.6. Podsumowanie

Przyjmijmy, że kontroler jest niezawodny. Pozostaną nam wówczas do rozważenia poniższe przypadki:

[*Z8* lub *z9* lub *z40* lub *z42*] i [*z7* lub (*z24* i *z26*) lub (*z25* i *z26*)]

1) Minimalne zbiory przyczyn zawierające zdarzenia: (*z7* i *z40*) lub (*z7* i *z42*) są związane z normalnym funkcjonowaniem rozjazdu.

   Niech *τ(zsi)* oznacza moment startu zdarzenia *i*, dla ustawiania zwrotnicy przez kontroler (zdarzenia *z7* i *z40*), to otrzymamy:

   $\tau(zs40) \leq -125 \text{ and } \tau(zs7) \geq -113$

   $\tau(zs40) \leq -204 \text{ and } \tau(zs7) \geq -18$

   oraz, dla zdarzeń *z7* i *z42*:

   $\tau(zs42) \leq -204 \text{ and } \tau(zs7) \geq -113$

   $\tau(zs42) \leq -125 \text{ and } \tau(zs7) \geq -18$

   Zatem, jeśli automatyczne sterowanie jest sprawne lub zadanie ustawienia zwrotnicy wykonuje dróżnik - do hazardu nie dojdzie.

2) Rozważmy teraz sytuację związaną ze zdarzeniem *z26*. W wyniku analizy otrzymamy następujące zależności czasowe dla zdarzenia *z26*:

   $z26\ /-\infty,-10,0,\infty/$,

   $z26/-18,-10,0,\infty/$

By doszło do hazardu zdarzenie *z26* musi trwać, co najmniej 10 jednostek czasu.

By takiej sytuacji zapobiec możemy wprowadzić, na przykład testowanie sensora przez kontroler, co pewien czas <10s. W przypadku stwierdzenia awarii kontroler postąpi tak, jak gdyby otrzymał sygnał z sensora (testowanie nie może być oczywiście przeprowadzane podczas przejazdu pociągu przez sekcję BO).

Mając wiedzę o tym, jakie zdarzenia i w jakich relacjach czasowych mogą powodować hazard możliwe jest także zbudowanie zabezpieczenia w postaci podsystemu. Zabezpieczenie może wykorzystywać na przykład dodatkową zwrotnicę (umieszczoną za zwrotnicą podstawową) kierującą pociąg na tor awaryjny. Jeśli zostanie wykryta koincydencja zdarzeń mogąca prowadzić do hazardu, to pociąg zostanie skierowany na tor awaryjny. Wjazd na ten tor będzie powodował automatyczne włączenie hamowania w pociągu.

8. Obszar zastosowań opracowanej metody

Opracowana metoda analizy drzew niezdatności z zależnościami czasowymi obejmuje cztery podstawowe typy bram:

- uogólniającą XOR,
- uogólniającą AND,
- przyczynową XOR,
- przyczynową AND.

W związku z tym, w obecnej postaci może być stosowana do drzew niezdatności zbudowanych z użyciem właśnie tych, dwuwejściowych modeli bram. Przy czym, możliwe jest zastępowanie bramek wielowejściowych odpowiadającymi im strukturami złożonymi z dwuwejściowych bram.

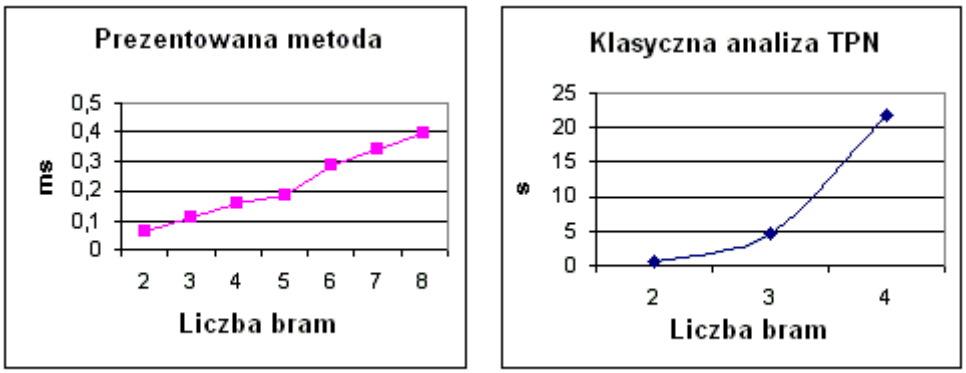
Zastosowanie wielowejściowych bram jest możliwe bez ingerencji w algorytmy związane z konstrukcją drzewa CT oraz RT, a jedynie poprzez odpowiednią definicję algorytmu „wczytywania" drzewa niezdatności z pliku. Zagadnienie to może stanowić obszar dalszych prac nad narzędziem do analizy drzew niezdatności z zależnościami czasowymi.

Złożoność czasowa przeprowadzania obliczeń w prezentowanej metodzie jest znacznie mniejsza niż w klasycznej analizie czasowych sieci Petri'ego modelujących drzewa niezdatności. Przeprowadzone porównanie dla implementacji algorytmu w wersji przedstawionej w [MS02] pozwoliło uzyskać poniższe wyniki:

Tabela 1

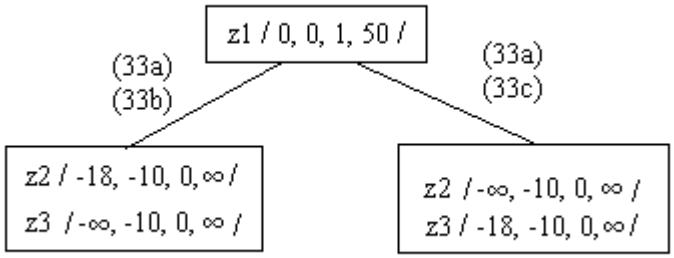
Liczba bramek przyczynowych		Średni czas obliczeń [s]	
AND	XOR	Prezentowana metoda	Klasyczna analiza TPN
1	1	59,15 • 10-6	0,500
2	1	108,00 • 10-6	4,844
3	1	157,67 • 10-6	21,782
4	1	186,74 • 10-6	-
4	2	286,27 • 10-6	-
4	3	341,42 • 10-6	-
4	4	398,43 • 10^{-6}	-

Szczegóły dotyczące algorytmu oraz uzyskanych wyników na komputerze klasy PC z procesorem AMD K5 100 MHz i pamięcią operacyjną wielkości 48MB zawiera publikacja [MS02]. Na rysunku 47 pokazano graficzną reprezentację wyników z tabeli 1.



Rys. 47. Zestawienie średnich czasów obliczeń dla prezentowanej metody oraz analizy TPN modelującej drzewa niezdatności w zależności od ilości bram

Wyniki czasowe przeprowadzonej symulacji są korzystne głównie ze względu na to, że analiza TPN przeprowadzana w sposób klasyczny wymaga wyznaczenia wszystkich możliwych klas stanów, natomiast w opracowanej metodzie tylko tych, które mogą prowadzić do wystąpienia hazardu. Rysunek 48 pokazuje wyniki uzyskane metodą INES dla fragmentu systemu rozjazdu kolejowego pokazanego na rysunku 10 z parametrami czasowymi jak dla TPN z rysunku 22.



Rys. 48. Wyniki analizy dla FT z rys. 10 (parametry jak dla TPN z rys. 22)

Dla porównania, wyniki uzyskane dla czasowej sieci Petri'ego (model przedstawia rysunek 22, a wyniki w postaci klas – rysunek 23) wymagały analizy 11 klas. Im więcej bram tym szybciej rośnie liczba klas. Właśnie ze względu na ich dużą liczbę wyniki w tabeli 1 kończą się na TPN modelującej cztery bramy. Z tego też względu TPN modelująca FT może być wykorzystana jedynie dla fragmentów dużych drzew.

Do przeprowadzania dynamicznej analizy dowolnego węzła drzewa RT wystarczy nam znajomość tylko parametrów dynamicznych zapisanych w tym węźle oraz znajomość FT, zatem możliwe jest również przeprowadzanie niezależnych obliczeń w systemach rozproszonych. Niewątpliwie będzie to jeden z obszarów dalszych prac związanych z algorytmami analizy i prezentacji wyników.

Rozważmy teraz jak kształtują się czasy analizy dla bardziej rozbudowanych drzew niezdatności. Rezultaty otrzymane z wykorzystaniem narzędzia [Sk05] na Laptopie ACTINA (procesor: Intel Celeron 2,4 GHz, pamięć operacyjna: DDR-SDRAM 256 MB, chipsem: SiS 650) pokazuje tabela 2.

Wyniki stanowią średni czas z 50 pomiarów. Ze względu na bardzo małe wielkości czasu, jeden pomiar obejmował 1000 powtórzeń wczytania FT z pliku i wyznaczenia CT lub 100 - krotne wyznaczenie RT – w drugim przypadku.

Tabela 2

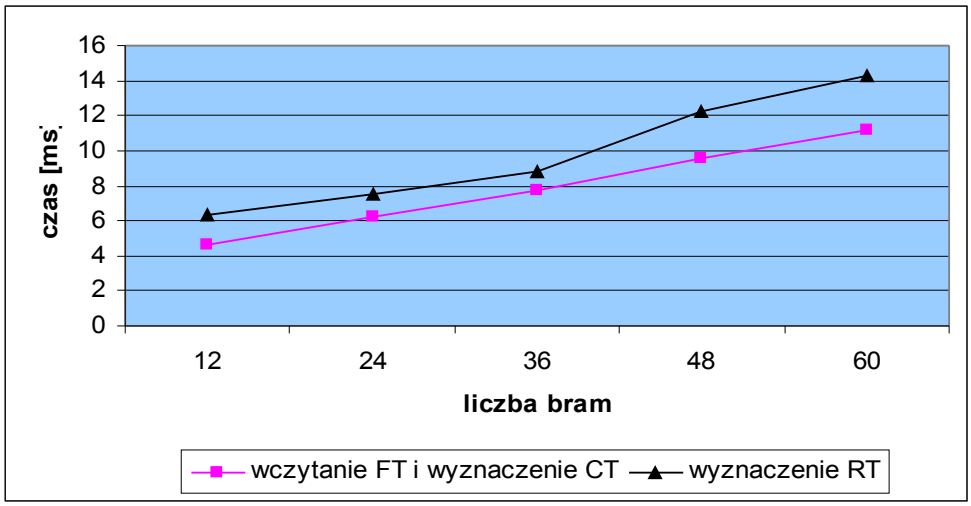
Liczba bram	Wczytanie FT z pliku oraz wyznaczenie CT [s]	Wykonanie obliczeń - algorytm RT [s]	Rodzaj FT poddanego analizie
12	$4{,}57 \bullet 10^{-3}$	$6{,}32 \bullet 10^{-3}$	FT z rysunku 43.
24	$6{,}21 \bullet 10^{-3}$	$7{,}51 \bullet 10^{-3}$	FT z rysunku 43 z dołączonym takim samym FT w miejsce zdarzenia 7 i zmienioną bramą 7 (poprzednio numer 1 w dołączanej strukturze) na G_XOR.
36	$7{,}77 \bullet 10^{-3}$	$8{,}79 \bullet 10^{-3}$	FT jak dla przypadku poprzedniego z dołączoną strukturą FT (taką jak na rys 43) w miejsce zdarzenia 8. Również brama 8 została zmienina na G_XOR.
48	$9{,}57 \bullet 10^{-3}$	$12{,}21 \bullet 10^{-3}$	FT jak dla przypadku poprzedniego z dołączoną strukturą FT (taką jak na rys 43) na wolne wejście bramy 11.
60	$11{,}16 \bullet 10^{-3}$	$14{,}28 \bullet 10^{-3}$	FT jak dla przypadku poprzedniego z dołączoną

			strukturą FT (taką jak na rys 43) w miejsce zdarzenia 9. Brama 9 została zmieniona na G_XOR

Analizie poddane zostały FT utworzone w oparciu o rysunek 43 i nie można ich uogólniać jako wyniki reprezentatywne dla wszystkich FT. Złożoność analizy zależy między innymi od takich czynników jak rodzaj i sposób rozmieszczenia bram (kolejność następowania po sobie) w FT oraz parametrów czasowych.

Na przykład: dla FT złożonego z czterech bram (korzeń – zdarzenie z bramą przyczynową AND, do prawego wejścia dołączony liść, do lewego wejścia dołączona struktura pokazana w załączniku na rysunku A2 - najgorszy przypadek dla 3 bram) uzyskujemy czasy: $2{,}23 \bullet 10^{-3}$ s (wczytanie FT i wyznaczenie CT), $4{,}25 \bullet 10^{-3}$ (dla RT).

Ze względy na ograniczenie elementów na liście *LISTA* w narzędziu prototypowym [Sk05], nie można było przeprowadzić analizy dla przykładu z rysunku A3. Będzie to możliwe po zakończeniu prac nad pełną wersją programu.



Rys. 49. Średni czas wykonywania obliczeń w zależności od ilości bram

Innym czynnikiem wpływającym na pomiary (ze względu na niewielkie wartości czasu) było środowisko *Windows XP Home Edition*, w którym uruchamiany był program przeprowadzający analizę. Wyniki przedstawione w tabeli 2 zostały

wygenerowane dla programu uruchomionego w trybie okienkowym, seria pomiarów dla trybu pełnoekranowego (w przypadku posiadanego laptopa wymagająca dodatkowego monitora zewnętrznego, gdyż oprogramowania *Borland Pascal 7.0* w trybie pełnoekranowym powoduje migotanie wyświetlacza LCD) była korzystniejsza ze względu na wartości czasu, ale wzrost pozostał proporcjonalny.

Wyniki dla algorytmu RT uzyskane wcześniej w trybie pełnoekranowym oraz z wyłączoną częścią programów uruchamiany standardowo na testowym komputerze były następujące: $4{,}15 \bullet 10^{-3}$ (dla 12 bram); $5{,}23 \bullet 10^{-3}$ (dla 24 bram); $7{,}44 \bullet 10^{-3}$ (dla 36 bram); $9{,}45 \bullet 10^{-3}$ (dla 48 bram).

9. Podsumowanie

W pracy została zaprezentowana nowa metoda analizy drzew niezdatności z zależnościami czasowymi.

W rozdziale 6 przedstawiony został formalny sposób notacji zależności czasowych dla zdarzeń i bramek w drzewach niezdatności, który pozwala na jednoznaczność oraz na precyzję zarówno opisu, jak i późniejszego procesu analizy.

Podstawę do przeprowadzenia analizy stanowią opracowane i opisane w rozdziale 5 systemy równań i nierówności. Natomiast rozdział 7 zawiera opracowany algorytm umożliwiający przeprowadzenie procesu analizy. Uszczegółowienie algorytmu zawiera załącznik A. Opracowana metoda została, między innymi, zweryfikowana na przykładzie systemu palnika gazowego [MS02], przejazdu kolejowego [MS03] oraz na przykładzie systemu rozjazdu kolejowego [Sk02].

Złożoność obliczeniowa prezentowanej metody jest znacznie niższa od złożoności analizy przy pomocy TPNs modelujących drzewa niezdatności. Porównanie czasu obliczeń obu metod zostało pokazane w rozdziale 8.

W opracowanej metodzie analizujemy dwa rodzaje warunków:

1) statyczne – wynikające z właściwości projektowanego systemu i otoczenia, których analiza jest prosta, a ich niespełnienie gwarantuje, że hazard nie wystąpi,
2) dynamiczne – liczone względem umownego momentu „0”; jeśli warunki dynamiczne zostaną spełnione, oznacza to, że hazard w analizowanym systemie może wystąpić.

W załączniku A został zaprezentowany proponowany sposób notacji zdarzeń w drzewach niezdatności, drzewie CT oraz RT, a także algorytmy umożliwiające przeprowadzenie analizy drzewa niezdatności.

Działanie algorytmu zostało zweryfikowane poprzez implementację na komputerze klasy PC w języku Borlad Pascal 7.0 i wykorzystane do uzyskania wyników dla przykładu zawartego w raporcie ([SK05] - testowa wersja narzędzia).

W związku z powyższym możemy stwierdzić, iż istnieje formalna metody analizy drzew niezdatności z zależnościami czasowymi na podstawie wstecznej analizy

czasowych sieci Petri'ego modelujących drzewa niezdatności – metoda została przedstawiona w niniejszym opracowaniu.

Metoda INES pozwala na analizę bardziej rozbudowanych FT niż w przypadku klasycznej analizy czasowych sieci Petri'ego modelujących drzewa niezdatności, choć może wymagać, w pesymistycznym przypadku, algorytmów równoległych do wykonywania obliczeń. Zapewnienie jednoznaczności i precyzji opisu zostało pokazane w rozdziale 6.

Ograniczenia metody to: możliwość korzystania z czterech typów dwuwejściowych bram oraz jednoprzyczynowość zdarzeń.

Analiza INES pozwala dodatkowo na:

- analizę bezpieczeństwa przy poprawnej pracy systemu, a w szczególności analizę osiągalności hazardu dla określonych parametrów czasowych,
- wykazanie, że określone zbiory przyczyn nie doprowadzą do hazardu (ze względu na relacje czasowe),
- wspomaganie projektowania podsystemów lub urządzeń wykrywających i zapobiegających wystąpieniu hazardu.

10. Bibliografia

[BD91] B. Berthomieu, M. Diaz, *Modelling and Verification of Time Dependent Systems Using Time Petri Nets*, IEEE Transaction of Software Engineering, vol. 17, no. 3, March 1991, pp. 259-273

[BDR96] L. Banachowski, K. Diks, W. Rytter, A*lgorytmy i struktury danych*, WNT, Warszawa 1996

[BFS75] Barlow R.E, Fussel J.B., Singpurwalla N.D., *Reliability and Fault Tree Analysis*, (Eds.), SIAM, Philadelphia, 1975

[BM82] B. Berthomieu, M. Menasche, *A State Enumeration Approach for Analyzing Time Petri Nets*, 3. European Workshop on Applications and Theory of Petri Nets, Varenna (Italy), September 1982

[CD97] S. Chakraborty, D.L. Dill, *Approximate Algorithms for Time Separation of Events*, in: Proc. ICCAD, 1997

[Cichocki97] T. Cichocki, *FMEA -zastosowanie metody,* Informatyka 10/97, październik 1997, str. 42-45

[CR83] J.E. Coolahan, N. Roussopoulos, *Timing Requirements for Time-Driven Systems Using Augmented Petri Nets*, IEEE Transaction of Software Engineering, vol. SE-9, no. 5, september 1983, pp.603-616

[Douglass90] B.P. Douglass, *Doing Hard Real Time, Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns*, Addison-Wesley, 1999

[DS85] red. I. Dziubiński, T. Świątkowski, *Poradnik matematyczny Część 1*, PWN, Warszawa 1985

[GMNRV77] S. Garriba, P. Mussio, F. Naldi, G. Reina, G. Volta, *Efficient Construction of Minimal Cut Sets from Fault Trees,* IEEE Transactions on Reliability, Vol. R-26, No 2, 1977, 88-94

[GMW95] J. Górski, J. Magott, A. Wardziński, *Modelling Fault Tree Using Petri Nets,* 14th International Conference on Computer Safety, Reliability and Security, SAFECOMP'95 Belgirate (Italy), October 11-13, 1995, pp.90-100

[Górski94] J. Górski, *Extending Safety Analysis Techniques With Formal Semantic,* In Technology and Assessment of Safety Critical Systems, (F.J. Redmill, Ed.), Springer-Verlag 1994

[Górski00] J. Górski red., *Inżynieria oprogramowania,* MIKOM, Warszawa 2000, str. 185-225

[GW94] J.Górski, A. Wardziński, *Analiza bezpieczeństwa komputerowego systemu sterowania*, Informatyka nr 8, 1994, str. 7-10

[GW95] J. Górski, A. Wardziński, *Formalising Fault Trees,* Safety Critcal System Symposium, Brihgton (UK), luty 1995

[HAZOP] Draft Interim Defence Standard: A Guidelline for HAZOP Studies on Systems which include a Programmable Electronic System, Ministry of Defence, 1995

[HK95] W. Hennings, N. Kuznetsov, *FAMOCUTN & CUTQN: Programs for Fast Analysis of Large Fault Trees with Replicated & Negated Gates*, IEEE Transactions on reliability, vol. 44, 1995, str. 368-376

[HM94] Z. Huzar, J. Magott, *Systemy czasu rzeczywistego -metody specyfikacj*, Informatyka nr 8, 1994, str. 14-17

[IEC 61882] *Hazard and operability studies (HAZOP studies) - Application guide*, International Electrotechnical Commision, IEC Standard, Publication 61882, 2001

[IEC1025] *Fault Tree Analysis (FTA)*, International Electrotechnical Commission, International Standard 1025, 1990

[IEC1508] *Functional Safety: Safety-Related Systems, Part 1: General Requirements*, IEC, 1995 Odpowiednik: PN-EN 61508-1, 2001

[IEC812] *Analysis technique for system reliability -Procedure for Failure Mode and Effect Analysis,* International Electrotechnical Commision, IEC Standard, Publication 812, 1985

[Intel] http://www.intel.com/products/processor/pentium4HTXE/index.htm

[Jarmuż96] P. Jarmuż, *Analiza bezpieczeństwa systemu czasu rzeczywistego*, praca dyplomowa, Francusko-Polska Wyższa Szkoła Nowych Technik Informatyczno -Komunikacyjnych, Poznań, 1996

[Jones90] C. B. Jones, *Systematic Software Development using VDM,* Prentice Hall Int., 1990

[Laprie89] J. C. Laprie, *Dependability: A Unifing Concept for Reliable Computing and Fault Tolerance*, BSP Professiona Books, Oxford, 1989

[Le95] Lees F P (1995). *Loss Prevention in the Process Industries.* 2nd Edition. Butterworth- Heinemann Ltd, Oxford, UK, (3 Volumes).

[Leveson87] N. Leveson, *Software Safety of embedded systems*, Communications of the ACM, vol. 34, no. 2, 1991, pp. 34-46

[LJ87] N. Limnious, J. P. Jeanette, *Event Trees and their Treatment on PC Computers,* Reliability Engineering , vol. 18, no. 3, 1987

[LJ97] L. Love, C. Johnson, *Using Diagrams to Support the Analysis of System 'Failure' and Operator 'Error'*, http://www.dcs.gla.ac.uk/~johnson/papers/aft.html

[LS87] N.Leveson, J. Stolzy, *Safety Analysis Using Petri Nets*, IEEE Transaction of Software Engineering, vol. SE-13, no. 3, march 1987

[LT93] N. G. Leveson, C. S. Turner, *An Investigation of the Therac-25 Accidents,* IEEE Computer, 1993, pp. 18-41

[MD92] K. L. Mc Millan, D. L. Dill, *Algorithms for Interface Timing Verification*, in*:* Proc. ICCD, October 1992

[MM96] Edited by C. Heitmeyer, D. Mandrioli, *Formal Methods for Real-Time Computing,* John Wiley & Sons Ltd, 1996, pp. 135-165

[Mosix] http://openmosix.sourceforge.net

[MS98] J.Magott, P.Skrobanek, *Metody analizy czasowych sieci Petri'ego modelujących drzewa niezdatności*, Systemy Czasu Rzeczywistego'98, Szklarska Poręba, 1998, str. 84-93

[MS00] J. Magott, P. Skrobanek, *A Method of Analysis of Fault Tree with Time Dependencies*, Proc. SAFECOMP 2000, Rotterdam, The Netherlands, LNCS, Vol. 1943, Springer-Verlag, 2000, pp. 176-186.

[MS02] J.Magott, P.Skrobanek, *Method of time Petri net analysis for analysis of fault trees with time dependencies*, IEE Proc.-Comput. Digit. Tech., Vol.149, No.6, November 2002

[MS03] J. Magott, P. Skrobanek, *Safety Analysis of a Railroad Crossing Using Fault Trees with Time Dependencie*s, in: Proc. Int. Workshop Computational Intelligence in Modelling, Control, and Automation, Wiedeń, 12-14, February, 2003

[MT95] Malhotra M., Trivedi K.S., *Dependablity Modeling Using Petri Nets*, IEEE Transactions on Reliability, Vol. 44, 1995 Sept., 428-440

[Ni71] D.S.Nielsen, *The Cause Cosequence Diagram Method as a Basis for Quantitative Accident Analysis*, Danish Atomic Energy Commision, RISO-M-1374, may 1971

[NUREG81] *Fault Tree Handbook* NUREG-0492, US Nuclear Regulatory Commission, 1981

[PN-88] *PN -88/T-01016/01 -/16 Przetwarzanie informacji i komputery. Terminologia*

[PN -EN 61508-3] *Bezpieczeństwo funkcjonalne - Systemy wiążące się z bezpieczeństwem - Część 3: Wymagania dotyczące oprogramowania*, Polski Komitet Normalizacyjny, 2002

[PN -EN 61508-7] *Bezpieczeństwo funkcjonalne - Systemy wiążące się z bezpieczeństwem - Część 7: Bibliografia stosowanych technik*, Polski Komitet Normalizacyjny, 2002

[PN-IEC1025] *Analiza drzew niezdatności (FTA)*, Polski Komitet Normalizacyjny, grudzień 1994

[PP88] L. B. Page, J.E. Perry, *An Algorithm for Fault-Tree Probabilities Using the Factoring Theorem, Microelectronics and Reliability,* Vol. 28, No 2, 1988, pp. 273-286

[Relex] *Relex® Reliability Software (v7.3)* Copyright © Relex Software Corporation

[Sacha97] K. Sacha, *Bezpieczeństwo oprogramowania w normie IEC1508*, In proc., Systemy Czasu Rzeczywistego'97, Szklarska Poręba, wrzesień, 1997, str. 175-182

[Sandia] http://www.sandia.gov/media/teraflop.htm

[Sandra] SiSoftware Sandra 2005 (program dostępny np.: http://www.octools.com/files/san2005.SR1-1050-W64-OTZ.exe)

[Shimeall91] T.J.Shimeall, *Software safety analysis in heterogeneous Multiprocessor Control Systems,* in. proc. Annual Reliability and Maintainability Symposium, 1991, str.290 - 294

[Sk00] P.Skrobanek, *Metoda analizy drzew błędów (FTS - Fault Trees) z zależnościami czasowymi,* Systemy Czasu Rzeczywistego'00, Kraków, 2000, str. 167-179

[Sk02] P.Skrobanek, *Analiza drzew błędów (FTS - Fault Trees) z zależnościami czasowymi,* Systemy Czasu Rzeczywistego'02, Ustroń, 2002

[Sk05] P. Skrobanek, Prototyp narzędzia do analizy FT z zależnościami czasowymi, 2005

[Sk96] P. Skrobanek, *Metoda badania zależności czasowych występujących w drzewach błędów*, p*raca* dyplomowa, Instytut Cybernetyki Technicznej, Politechnika Wrocławska, 1996

[Sk97] P. Skrobanek, *Wykorzystanie drzew błędów z zależnościami czasowymi do wprowadzania zabezpieczeń do systemów*, Systemy Czasu Rzeczywistego'97, Szklarska Poręba, 1997, str. 183-192

[SPG91] A. Silberschatz, J. Peterson, P. Galvin *Podstawy systemów operacyjnych*, the Polish edition WNT, Warszawa 1993, str.196

[Sysło97] M. Sysło, *Algorytmy*, WSiP, Warszawa 1997

[TVP] *Wiadomości 19*[30], TVP 1, 14 marca 2005

[Ur01] Urbanek A., *Leksykon Teleinformatyka*, IDG Poland S. A., Warszawa 2001

[Vesely81] W.E.Vesely et el., *Fault Tree Handbook,* NUREG 0492, US Nucleary Regulatory Commission, 1981

[VGM92] P. Vanbekbergen, G. Goossens, H. D. Man, *Specification and Analysis of Timing Constraints in Signal Transition Graphs*, in: Proc. European DAC, March 1992

[W96] A. Wardziński, *Analiza drzew błędów systemów komputerowych związanych z bezpieczeństwem, Rozprawa doktorska ETI 6/96*, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska, 1996

[Wirth80] N. Wirth, *Algorytmy + struktury danych = programy*, WNT, Warszawa 1980

[Ż95] Z. Żurakowski, *Systemy Komputerowe w Zastosowaniach Związanych z Bezpieczeństwem*, Informatyka, nr 3, 1995

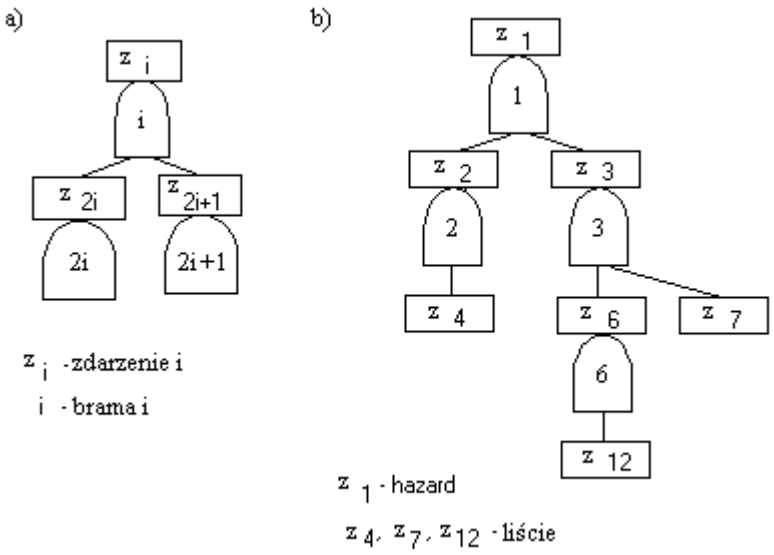
ZAŁĄCZNIK A

A.1. Opis drzewa niezdatności

Przyjmijmy następującą numerację zdarzeń i bramek w drzewie niezdatności przedstawioną na rysunku A1. Zgodnie z nią:

- numer bramki oraz zdarzenia na jej wyjściu jest taki sam,
- hazard, który w takim drzewie stanowi korzeń ma numer 1,
- dla dowolnego zdarzenia o numerze i, numery następników to $2 \cdot i$ oraz $2 \cdot i + 1$.

Sama metoda nie narzuca ograniczeń na numerację zdarzeń i bramek w drzewie. Przyjęte rozwiązanie ma na celu uproszczenie budowy algorytmu oraz jego implementacji.



Rys. A1. Numeracja zdarzeń i bramek w drzewie niezdatności a) dowolne zdarzenie z_i oraz brama i, b) przykład drzewa

Nazwijmy zdarzenie wyjściowe wraz z bramką węzłem. Wówczas możemy powiedzieć, że drzewa niezdatności opisujemy poprzez strukturę złożoną z węzłów:

w _FT= record of

nr_zdarzenia	//	W naszym przypadku - numer zdarzenia wyjściowego oraz numer bramki.
as, bs, ae, be	//	Wartości odpowiadające odpowiednio parametrom statycznym startu i zakończenia zdarzenia *i*: α^S_{si}, β^S_{si}, α^S_{ei}, β^S_{ei}. Dla bramek uogólniających parametry te nie będą określane.
ds, de, d2s, d2e	//	Parametry określające opóźnienie pomiędzy przyczyną (przyczynami) oraz skutkiem. Dla bramki C_AND będą zapisywane tylko wartości *ds* oraz *de*. Dla bramki C_XOR wszystkie cztery. Dla bramek uogólniających (G_XOR, G_AND) tych parametrów nie definiujemy.
typ	//	Możliwe wartości: C_AND, C_XOR, G_AND, G_XOR, LISC. Przy czym: LISC oznacza zdarzenie z_i które jest liściem (nie występuje w żadnej bramie).

end;

Powyższa struktura węzła dotyczy rozwiązania związanego z przyjętą numeracją bramek oraz zdarzeń. Na podstawie numeru zdarzenia jesteśmy w stanie jednoznacznie określić jego położenie w drzewie (poprzednik oraz następniki). W innym przypadku należałoby dodać, co najmniej dwa pola do oznaczenia węzłów potomnych.

Powyższy opis jest również zgodny z prezentowanym algorytmem w [MS02]. Drzewo niezdatności jest zbiorem węzłów zapisanych w powyższej postaci.

A.2. Opis drzewa przypadków

Algorytm przekształcenia drzewa niezdatności w drzewo przypadków zostanie pokazany w rozdziale A5. Ponieważ po przekształceniu jeden węzeł może mieć więcej niż dwa następniki, to w opisie węzła należy umieścić wskaźniki do węzłów potomnych. Ponadto należy uwzględnić to, że bramki *uogólniające AND* mogą ulec replikacji. Ponieważ w procesie tym powielona brama otrzymuje nowy numer, należy zatem dodać pole, w którym zapamiętamy numer bramki, którą powielono. Zgodnie z powyższym, węzeł w drzewie CT (ang. *CT – Case Tree*) może mieć poniższą strukturę:

w_CT= record of

nr_zdarzenia		
nr_stary	//	Numeru bramki G_AND z której została powielona
	//	dana brama G_AND.
as, bs, ae, be	//	W przypadku bramki typu G_AND, *be* jest
	//	szacowane na podstawie maksymalnych czasów
	//	współwystępowania zdarzeń wejściowych (patrz
	//	rysunek 31).
ds, de, d2s, d2e		
Typ	//	Po reorganizacji w drzewie nie występują bramki
	//	G_XOR. Pozostałe wartości, jak poprzednio.
LL	//	Lista numerów węzłów potomnych związanych
	//	z lewym wejściem.
LP	//	Lista numerów węzłów potomnych związanych
	//	z prawym wejściem

end;

Listy *LL* oraz *LP* przechowują numery węzłów potomnych związanych odpowiednio z lewym oraz z prawym wejściem bramki. W praktyce nie ma konieczności przepisywania parametrów statycznych dla zdarzeń z FT, gdyż mając dane *nr_zdarzenia* i *nr_stary* możemy te wartości odczytać z FT.

A.3. Opis drzewa wyników

Do prezentacji wyników wykorzystywane są poniższewęzły:

w_RT= record of //ang. *RT –result tree*

nr_wezla		
nr_zdarzenia_bramki	//	Numeru zdarzenia wyjściowego, jak i numer
	//	bramki. Ze względu na przyjęty sposób numeracji,
	//	numery te są identyczne.
as, bs, ae, be	//	Dynamicznie wyznaczone wartości startu
	//	i zakończenia zdarzenia o numerze
	//	*nr_zdarzenia_bramki*
w_and	//	Pole to służy do zaznaczenia węzłów występujących
	//	w strukturach AND. Wszystkie węzły tworzące
	//	jedną strukturę AND mają ten sam numer. Dla
	//	węzłów związanych strukturą XOR pole to

	//	przyjmuje wartość 0, co zostanie pokazane.
ojciec	//	Numer węzła „rodzica”.

end;

LS, LB :lista elementów typu w_RT;

Listy te, zostaną wykorzystane do przechowywania węzłów, które były już analizowane (*LS*) oraz tych, które będą analizowane (*LB*). Na danym etapie analizy suma elementów z list $LS \cup LB$ określa minimalne zbióry przyczyn.

A.4. Algorytm główny

Algorytm główny obejmuje:

```
K1: wczytaj_FT('ft.txt'); {wczytanie drzewa FT z pliku}

K2: utworz_CT;            {utworzenie drzewa CT}

K3: wyznacz_RT;           {wykonanie obliczeń - wyznaczenie
                           minimalnych zbiorów przyczyn}
```

Po uzyskaniu wyników w postaci drzewa RT jesteśmy w stanie odpowiedzieć na pytanie, czy hazard w danym systemie może wystąpić oraz jakie zdarzenia muszą się pojawić i w jakich przedziałach czasu. Ta wiedza pozwala nam na takie postępowanie, by do hazardu nie dopuścić lub zminimalizować możliwość jego wystąpienia.

A.5. Algorytm CT

A.5.1. POSTAĆ OGÓLNA

procedure CT(i; var LISTA)

{***i*** – numer węzła przekazywany do procedury, dla którego będą wyznaczane następniki; ***LISTA*** – lista następników węzła *i*, zwracana przez procedurę, w szczególnym przypadku może być jeden}

LL, LP*;*

{LL, LP – zmienne, tego samego typu co LISTA; służą do przechowywania, odpowiednio, lewych i prawych następników węzła *i*}

begin

K1: jeśli lewy następnik jest liściem, to umieść go na liście LISTA,
w przeciwnym razie wywołaj procedurę ***CT*** dla lewego następnika:
CT (2*i, LL)

K2: jeśli prawy następnik jest liściem, to umieść go na liście *LISTA*,
w przeciwnym razie wywołaj procedurę ***drzewo*** dla prawego następnika:
CT (2*i+1, LP),

K3: zapisz węzeł, ustalając jego elementy na podstawie list *LL* i *LP* oraz na podstawie typu i numeru bramki.

end;

Postępując w ten sposób, dojdziemy do liści, a następnie idąc „od dołu" zbudujemy drzewo CT. Pomijanie bramek uogólniających typu XOR, a także powielanie bramek uogólniających typu AND ma miejsce w kroku K3, co zostanie pokazane w uszczegółowieniu algorytmu.

A.5.2. USZCZEGÓŁOWIENIE

Niech:

max_nr –największy numer nadany zdarzeniu w drzewie niezdatności.

```
procedure CT(i:word;var LISTA:tlista_CT);
{i – numer węzła, LISTA – lista lewych i prawych następników węzła.
Prawe następniki kodowane są dla odróżnienia znakiem „-"}

var LL, LP :tlista_CT; {listy LL i LP przechowują, odpowiednio, listę
                        lewych i prawych następników węzła}
    LL_p,LP_p :tlista_CT; {listy pomocnicze}
    k,l    :integer;      {zmienne pomocnicze}
    poz_LL, poz_LP, a, b :word;


1.   Begin
        {K1: wyczyszczenie list LL i LP}
2.   wyczysc_LL_i_LP();
        {K2: wyznaczenie lewych następników}
3.        if typ(2*i)=LISC then LL←LL+(2*i); {jeśli lewy następnik to
```

```
                                                    liść - dopisujemy go na
                                                    koniec listy LL}
4.          if typ(2*i) in [1,2,3,4] then CT(2*i,LL); {jeśli lewy
                    następnik jest zdarzeniem wyjściowym bramki (1,2,3,4
                    – typy bram) wówczas, rekurencyjnie, wywołujemy
                    procedurę CT dla lewego następnika; w wyniku
                    otrzymamy listę następników dla lewego wejścia: LL}
              {Jeżeli do lewego wejścia nie dołączono zdarzenia, co może
              mieć miejsce w przypadku bram XOR, to nie wykonujemy żadnej
              akcji}
      {K3: wyznaczenie prawych następników – analogicznie, jak
           w poprzednim przypadku}
5.    if typ(2*i+1)=LISC then LP←LP+(2*i+1);
6.          if typ(2*i+1) in [1,2,3,4] then CT(2*i+1,LP);

          {K4: zapisanie węzła, po wyznaczeniu następników}
7.          case typ(i) of
8.             G_XOR: CT_pomin_GXOR;  {w przypadku bramki uogólniającej
                      XOR, na listę LISTA, dopisywane są wszystkie
                      elementy z LL i LP; nie jest zapisywany węzeł w CT
                      –wszystkie następniki są „przekazywane” poziom
                      wyżej w drzewie}
9.             G_AND: CT_zapisz_GAND; {jeśli liczba następników lewego
                      lub prawego wejścia jest większa niż 1, to dla
                      każdej pary następników nie licząc pierwszej, ma
                      miejsce replikacja bramki; dodatkowo, w każdym
                      przypadku szacowane są parametry statyczne dla
                      zdarzenia wyjściowego bramki, zgodnie z rys. 44;
                      na listę LISTA zapisywany jest tylko numer bramki
                      i/lub numer bramki po replikacji}
10.     C_XOR,C_AND: CT_zapisz_C; {w przypadku bramki przyczynowej,
                      zapisujemy węzeł, a na listę LISTA zapisujemy
                      numer bramki i}
11.          end;
12.   end;
```

A.5.3. USZCZEGÓŁOWIENIE PROCEDUR

```
procedure CT_zapisz_C; {zapis dowolnej bramki typu causal}
   begin
     zapisz_CT(i,i,LL,LP); {zapisanie nowego węzła do struktury CT}
     LISTA←i;          {zapisanie na listę numeru węzła; Numer ten
                            zostanie przekazany do procedury CT
```

```
                              z której nastąpiło wywołanie, zatem numer
                              węzła i znajdzie się na liście, jako
                              następnik w węźle z którego nastąpiło
                              wywołanie}
   end;
```

```
procedure CT_pomin_GXOR;

   begin                      {Dla bramki uogólniającej XOR nie
                              zapisujemy węzła w drzewie CT.
                                 Następniki, zarówno lewego, jak
                              i prawego wejścia są zapisywane na listę
                              LISTA. Zostaną one przekazane do
                              procedury, z której nastąpiło wywołanie.
                              Zostaną „potraktowane” jako następniki
                              (zależnie od wywołania, prawego lub lewego
                              wejścia) węzła związanego z procedurą
                              wywołującą.
                                 Jeśli zdarzyłaby się sytuacja taka,
                              iż procedura, z której nastąpiło
                              wywołanie, związana jest również z bramką
                              G_XOR, to zostaną one przekazane do
                              kolejnego poziomu „wyżej” w drzewie CT}
     LISTA←LL;                {Dołączenie do listy LISTA lewych
                              następników.}
     LISTA←LP;                {Dołączenie do listy LISTA prawych
                               następników bramki uogólniającej XOR}
  end;
```

```
procedure CT_zapisz_GAND;

{    W procedurze tej ma miejsce replikacja bram oraz szacowanie
parametrów statycznych. Czas trwania zdarzenia wyjściowego bramy
z jest uzależniony od współwystępowania zdarzeń wejściowych x i y.
Najkrócej może trwać 0 (dla zdarzenia natychmiastowego), a najdłużej
tyle, ile wynosi mniejszy z maksymalnych czasów trwania zdarzeń x
i y. Zatem: asze=0, bsze=min{bsxe,bsye}. Czasy te szacowane są
w procedurze CT_GAND_bsze}
```

```
 var r1,r2, :integer;      {Zmienne pomocnicze}
       nr_i :longint;
begin
  for r1:=1 to rozmiar(LL) do
    for r2:=1 to rozmiar(LP) do
      {pierwszy element – bez replikacji}
      if (r1=1) and (r2=1) then
           begin
                CT_GAND_bsze
                              {Wyznaczenie asze oraz bsze dla pary
                              elementów z list LL i LP oraz
                              zapisanie do struktury przechowującej
                              informację o FT, patrz rozdział A.2}
                nr_i:=i;
            end
       else {replikacja bramki}
           begin
             max_z:=max_z+1;
               CT_GAND_bsze_new; {Utworzenie nowej bramki
                              o numerze max_z = replikacja
                              analizowanej bramki, a następnie
                              wyznaczenie asze oraz bsze dla pary
                              elementów (LL[r1], LP[r2])
                              i zapisanie do struktury
                              przechowującej informację o FT.}
               nr_i:=max_z;
           end;
   zapisz_CT(nr_i,i,LL[r1],LP[r2]); {zapis węzła do CT}
   LISTA ←nr_i; {Dopisanie na listę LISTA kolejnego następnika}
 end;
```

A.5.4. CT – złożoność obliczeniowa

Analiza drzew niezdatności przyjętą metodą w dużym stopniu zależy od konstrukcji drzewa oraz przyjętych parametrów statycznych. Jeśli ze względu na te parametry do hazardu nie może dojść, to analiza może zakończyć się nawet zaraz po rozpoczęciu analizy drzewa przypadków, ale zawsze przeprowadzona zostanie konstrukcja drzewa CT.

W tym podrozdziale zostanie oszacowana złożoność obliczeniowa algorytmu tworzenia drzewa CT.

OSZACOWANIE LICZBY ZDARZEŃ

Niech drzewo błędów zawiera m - bram. Ponieważ bramy mają co najwyżej dwa zdarzenia na wejściu (bramy typu XOR mogą mieć jedno), to w naszym drzewie mamy co najwyżej *m+1* liści [Wirth80]. Zatem liczba zdarzeń w drzewie niezdatności wynosi co najwyżej $m + (m+1) = 2m+1$.

OSZACOWANIE ZŁOŻONOŚCI DLA PROCEDURY GŁÓWNEJ

Operacje dominujące w zaprezentowanym algorytmie, to porównanie i podstawienie. Przeanalizujmy kolejno poszczególne instrukcje algorytmu CT.

Instrukcja 2.

Czyszczenie list polega na wypełnieniu ich umowną wartością (zatem ilość podstawień jest taka, jak wielkość list) lub jedynie na zainicjowaniu listy (w przypadku list tworzonych dynamicznie). Zatem, zarówno w pierwszym, jak i drugim rozwiązaniu (zależnie od implementacji list) mamy stałą liczbę operacji. Oznaczmy ją jako C_1.

Komentarza wymaga wywołanie rekurencyjne procedury CT.

Zwróćmy uwagę, że rozumowanie dla instrukcji 1 dotyczy pojedynczego wywołania procedury CT. W praktyce będą one realizowane tyle razy, ile będzie wywołań rekurencyjnych procedury CT. Ponieważ wywołania te mają miejsce dla węzłów FT nie będących liśćmi, ilość takich operacji będzie nie większa niż liczba bram, czyli *m*. W związku z powyższym, maksymalną ilość operacji możemy oszacować na: $C_1 * m$

Instrukcja. 3.

Mamy jedno porównanie i co najwyżej jedno podstawienie. Zauważmy również, że instrukcja ta wykonuje się tylko dla połowy węzłów w FT (tylko dla lewych następników każdego węzła), a zatem liczba wykonań tej instrukcji nie przekracza: *m+1*. Ponadto, co najwyżej połowa wywołań będzie związana ze spełnieniem warunku, a zatem nie więcej niż $\lceil (m+1)/2 \rceil$ podstawień. Gdzie $\lceil\ \rceil$ – zaokrąglenie w górę do liczby całkowitej.

Zatem, liczba porównań + liczba podstawień jest nie większą niż:

$m+1+\lceil(m+1)/2\rceil$

Funkcja typ(i)) pobiera ze struktury opisującej FT (np. tablica) informację o typie bramy (np. liczbę kodującą typ). W zależności od implementacji może wymagać od jednej do kilku operacji. Oznaczając stałą liczbę operacji przez C_2 otrzymamy:

$(m+1+\lceil(m+1)/2\rceil)*C_2$

Instrukcja 4.

Analogicznie do rozumowania przeprowadzonego dla instrukcji 3, liczba porównań będzie nie większa niż $m+1$.

Instrukcja 5.

$(m+1+\lceil(m+1)/2\rceil)*C_3$ - dla prawych następników, analogicznie do instrukcji numer 3.

Instrukcja 6.

$m+1$ - dla prawych następników, analogicznie do instrukcji 4.

Instrukcja 7.

Tylko jeden z jej elementów (instrukcji 8. 9. lub 10.) zostanie wykonany raz w każdym wywołaniu procedury CT. Ponieważ procedura CT zostanie wywołana, jak ustaliliśmy wcześniej, nie więcej niż m razy, zatem liczba wykonań tej instrukcji będzie nie większa niż m. W celu dalszego szacowania złożoności rozważymy teraz złożoność instrukcji 8, 9, 10 – na podstawie uszczegółowienia procedur: *CT_pomin_GXOR*, *CT_zapisz_GAND*, *CT_zapisz_C*.

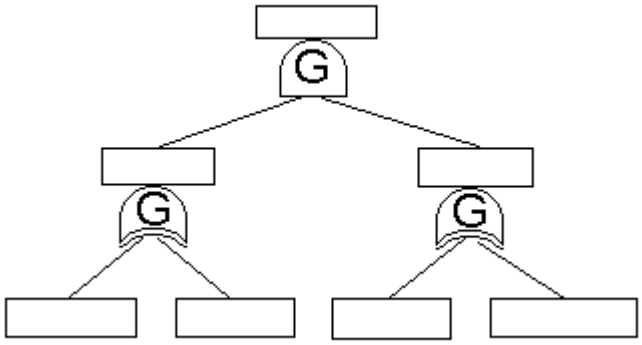
Instrukcja 8. (*CT_pomin_GXOR*)

Ilość operacji jest zależna od implementacji struktury list. Jeśli implementacja związana będzie listami realizowanymi dynamicznie, wówczas dołączenie wymagało będzie jedynie, np. modyfikacji wskaźnika na kolejny element. Możemy przyjąć, że będzie to kilka operacji. Oznaczmy liczbę potrzebnych operacji jako C_4.

Jeśli całe drzewo byłoby zbudowane z tego typu bram, to liczba operacji będzie: $m*C_4$

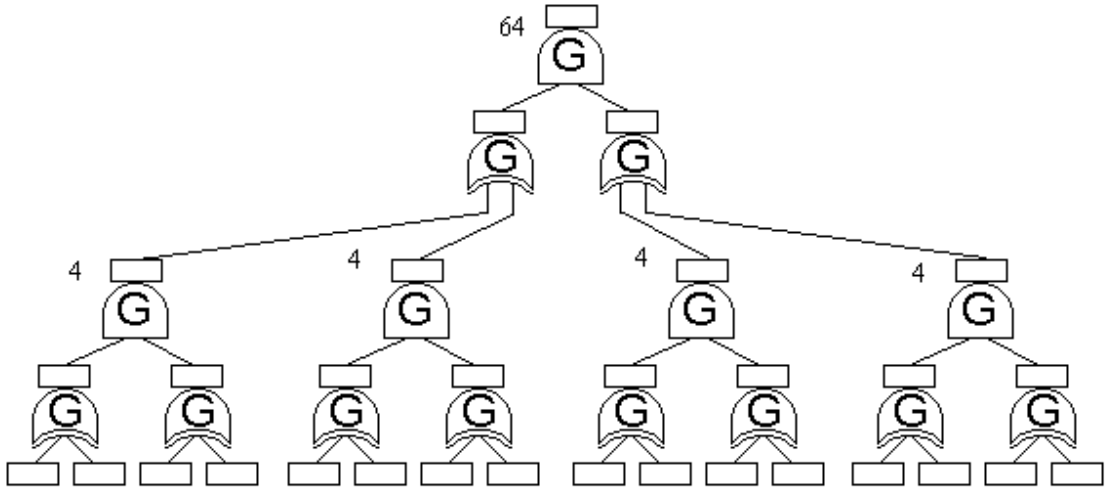
Instrukcja 9. (*CT_zapisz_GAND*)

Najgorszy przypadek dla trzech bram ma miejsce dla struktur pokazanych na rysunku A2 (na rysunku nie zaznaczono opisu zdarzeń, gdyż nie jest to istotne dla przeprowadzanych rozważań). Wówczas lista *LL* i *LP* dla bramy AND będą zawierały po dwa elementy, a zatem trzeba będzie przeanalizować 2 x 2 przypadki, a brama uogólniająca zostanie powielona trzy razy.



Rys. A2. Najgorszy przypadek dla trzech bram w FT

Skonstruujemy teraz FT złożone z pięciu takich struktur jak na rysunku A2 – wynik pokazuje rysunek A3.



Rys. A3. Najgorszy przypadek dla 15 bram w FT

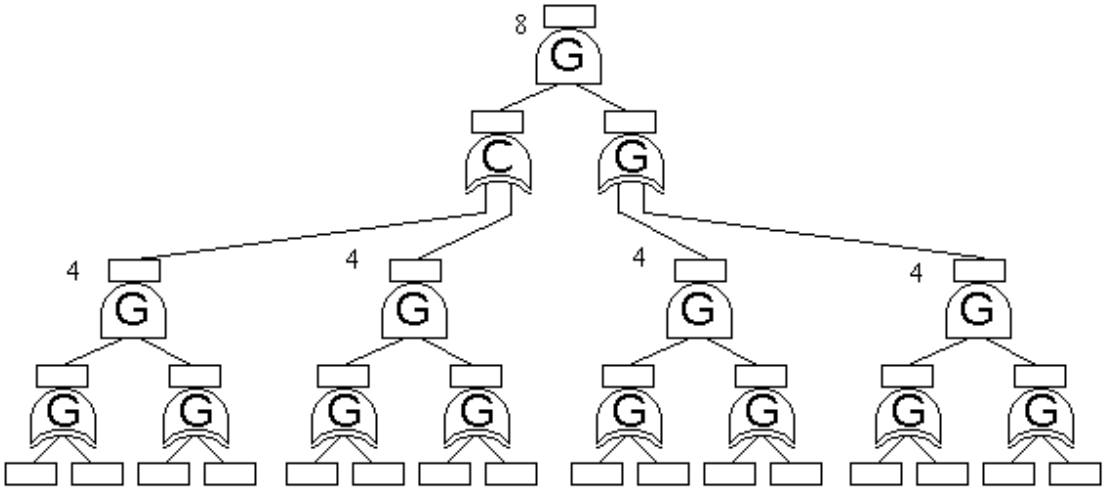
Cyfry przy bramach AND mówią o liczbie elementów na liście *LISTA* (zwracanych poziom wyżej). Jeśli postąpimy podobnie dla struktury z rysunku A3, jak dla FT z rysunku A2, to otrzymamy na wyjściu bramy 128 x 128 przypadków. Wyniki dla kolejnych kroków takiego postępowania oraz oszacowanie liczby iteracji (odpowiada ona liczbie elementów na liście *LISTA* szczytowej bramy AND) związanych z elementami na listach *LL* i *LP* przedstawia poniższa tabela:

Liczba bram	Ilość elementów na liście *LISTA* dla szczytowej bramy AND	UWAGI
3	4	FT z rys. A2.
15	64	FT z rys. A3
63	16384	
255	1073741824	
1023	$4{,}61169*10^{18}$	

Ilość elementów na wyjściu bramy szczytowej określa wzór: (2*'ilość w kroku poprzednim')^2. Możemy zatem mówić o złożoności wykładniczej.

Powyższy przykład jest czysto teoretyczny. W praktyce FT zawierają różne typy bram (przykłady różnych FT można znaleźć między innymi w [W96], [GMW95], [MS03], [Leveson87]). Poza tym, taka konstrukcja FT jak dla pesymistycznego przypadku nie pozwala również na analizę klasyczną (zbyt duża liczba minimalnych zbiorów przyczyn).

Ponieważ na wyjściu bram przyczynowych zawsze jest jedno zdarzenie, to ich obecność w drzewie niezdatności znacznie redukuje liczbę przypadków – na rysunku A4 przedstawiono przykład takiej sytuacji.



Rys. A4. Najgorszy przypadek dla 15 bram po zamianie jednej bramy na przyczynową

W związku z powyższym, możemy przyjąć założenie, iż pesymistyczna złożoność nie występuje w FT modelujących rzeczywiste systemy. W innych przypadkach szacowana złożoność jest $o(m^2)$ – nawet dla przypadku, gdy pierwsza brama jest uogólniająca AND, a pozostałe uogólniające XOR.

Zagadnienia złożoności oraz algorytmy jej szacowania na podstawie struktury FT będą stanowiły obszar dalszych badań nad metodą INES.

Instrukcja 10. (*CT_zapisz_C*)

W procedurze tej następuje zapisanie węzła do struktury CT, co zajmuje stałą liczbę kilku, kilkunastu (w zależności od implementacji struktury CT) operacji podstawienia oraz jedno podstawienie elementu na listę. Liczbę tych operacji zapiszmy jako C_6.

Ponieważ procedura ta może być wywołana maksymalnie *m* razy (jeśli wszystkie bramy będą typu *causal*), zatem ilość operacji związana z tą instrukcją będzie nie większa niż:

$C_6 * m$

Zbierając otrzymane wyniki:

Instrukcja		Oszacowana liczba operacji dla m bram
2		$m*C_1$
3		$(m+1+\lceil(m+1)/2\rceil)*C_2$
4		$m+1$
5		$(m+1+\lceil(m+1)/2\rceil)*C_3$
6		$m+1$
7	8	$m'*C_4$
	9	$(m'')^2$ – *dla FT modelujących rzeczywiste systemy*
	10	$m'''* C_6$

Przy czym: $m'+m''+m'''=m$.

Ponieważ instrukcje 2 do 7 są wykonywane sekwencyjnie, zatem szacowana złożoność obliczeniowa w najgorszym przypadku jest $o(m^2)$. Liczba bram w drzewach niezdatności może być rzędu kilkuset (dane nieoficjalne - jak to zostało powiedziane we wstępie, FT dla określonego systemu stanowi tajemnicę firmy i nie jest udostępniane dla osób postronnych. Praktycznie nie można uzyskać przykładowych FT dużych systemów).

Częstotliwość taktowania dzisiejszych procesorów jest rzędu GHz i wynosi, np. 3,73 GHz (dla procesora Pentium® 4 *Extreme Edition*, dane wg. [Intel]). Według [Sandra] szybkość wykonywania operacji stałopozycyjnych w takim procesorze (Pentium® 4 3,8 GHz) wynosi około 13231 MIPS. Rozważmy teoretyczny czas wykonywania operacji:

Tabela A1

m (liczba elementów)	500	1 000	5 000
m^2	250 000	1 000 000	25 000 000
Liczba operacji do wykonania (przyjmijmy 1000 *m^2)	250 000 000	1 000 000 000	25 000 000 000
Czas wykonania [s]	0,02	0,07	1,88

Zwróćmy uwagę, że złożoność wszystkich operacji w instrukcjach innych niż 9, są *o(m)*. W praktyce jednak drzewa niezdatności składają się z różnych typów bram, a zatem i proces konstrukcji drzewa CT będzie przebiegał szybciej. Ponadto, dla liczby bram rzędu 1000, nawet, jeśli są to bramy uogólniające AND, też jesteśmy w stanie w rozsądnym czasie przeprowadzić konstrukcję drzewa CT.

Rozważmy jeszcze sytuację teoretyczną omówioną dla instrukcji dziewiątej. Czas wykonania operacji związanych z tym przypadkiem jest rozsądny dla 255 bram (patrz tabela A2), przy założeniu 100 operacji przypadających na jedną iterację. Zatem nawet dla dużych FT bylibyśmy w stanie przeprowadzić konstrukcję CT w rozsądnym czasie.

Tabela A2

liczba bramy	Ilość elementów na liście *LISTA* dla ostatniej bramy	Ilość iteracji związana z wszystkimi bramami AND w FT	100*Ilość iteracji	Czas [s]
3	4	4	400	$3{,}02*10^{-8}$
15	64	68	6800	$5{,}14*10^{-7}$
63	16384	16452	1645200	$1{,}24*10^{-4}$
255	1073741824	1073758276	$1{,}07376*10^{11}$	8,12
1023	$4{,}61169*10^{18}$	$4{,}61169*10^{18}$	$4{,}61169*10^{20}$	34855158488 (ponad 1100 lat)

A.5.5. CT – przykład dla systemu sterowania rozjazdem kolejowym

Zapis poszczególnych węzłów w drzewie CT:

1 2 -6 -43 -44 -45 -7 0 2 4 -10 -11 0 4 8 -9 0 6 24 -26 0 10 20 -21 0 11 22 0 20 40 0 21 42 0 43(6) 24 -27 0 44(6) 25 -26 0 45(6) 25 -27 0

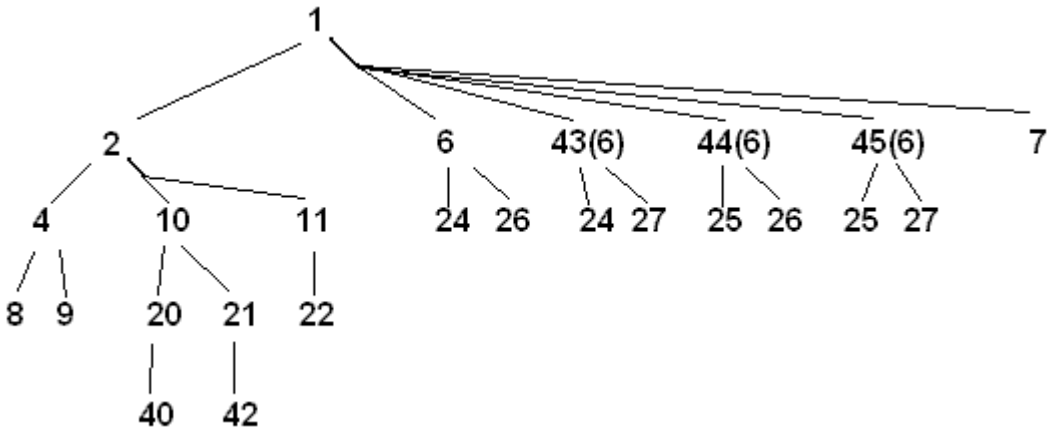
1 2 -6 -43 -44 -45 -7, gdzie: 1 – numer bramki, 2 – następnik związane z lewym wejściem, -43, -44, -45, -7 – następniki związane z prawym wejściem bramki 1

Wartość „0” użyto jako znacznik końca węzła.

Bramki 43, 44, 45 powstały w wyniku replikacji bramki nr 6. Jak to zostało wskazane w rozdziale A.2, węzły drzewa CT nie mszą zawierać informacji o parametrach statycznych związanych zarówno z bramką, jak i ze zdarzeniem na jej

wyjściu, gdyż w obliczeniach wykorzystywane są parametry zapisane w strukturze drzewa FT.

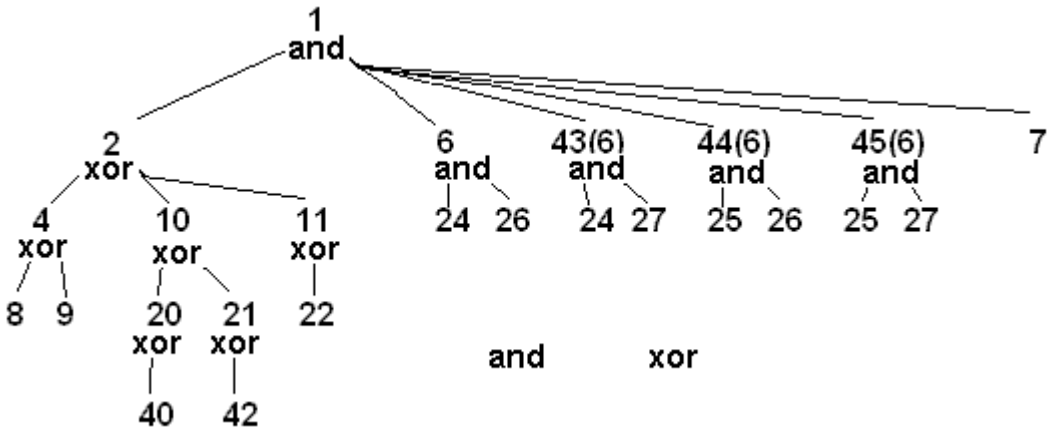
Drzewo CT w postaci graficznej znajduje się na rysunku A5.



Rys. A5. Drzewo CT dla systemu rozjazdu kolejowego

Na rysunku A5 zostały zaznaczone tylko numery węzłów. Jak to zostało opisane wcześniej, każdy numer odpowiada numerowi zdarzenia oraz numerowi bramki (oprócz węzłów zewnętrznych – liści, które bramek nie zawierają).

Ponieważ wiemy, które bramki są typu AND, a które XOR, w prosty sposób możemy na bazie drzewa CT uzyskać minimalne zbiory przyczyn, lecz bez zależności czasowych. Uzyskane wyniki będą odpowiadały klasycznej analizie drzew niezdatności. Drzewo CT z zaznaczonymi typami bramek przedstawia rysunek A6.



Rys. A6. Drzewo CT z zaznaczonymi typami bram

Przykładowe minimalne zbiory przyczyn:

a) 8, 24, 26
b) 8, 24, 27
c) 42, 7

W procesie tworzenia minimalnych zbiorów przyczyn z wykorzystaniem drzewa CT należy zwrócić uwagę, żeby różnych przypadków nie potraktować łącznie. To znaczy, jeśli do jakiegoś wejścia bramki mamy dołączone różne przypadki, należy traktować je oddzielnie.

Przykładowo, brama nr 10 ma dołączone do prawego wejścia dwa zdarzenia 10 i 11, a brama numer 1 – pięć zdarzeń (co jest związane z pięcioma przypadkami do analizy). Stworzenie minimalnego zbioru przyczyn: [10, 11, 6, 43(6), 44(6), 45(6), 7] byłoby błędem. W tej sytuacji uzyskamy kilka minimalnych zbiorów przyczyn: [10,6], [10,43(6)], [10,44(6)], [10,45(6)], [10,7], [11,6], [11,43(6)], [11,44(6)], [11,45(6)], [11,7]. Docelowo zaś: [40,24,26], [42,24,26], [40, 24, 27], …

Uwagi, dotyczące tworzenia minimalnych zbiorów przyczyn z wykorzystaniem CT, mają znaczenie tylko ze względu na możliwość wykorzystania wyników do prezentacji minimalnych zbiorów przyczyn w ujęciu klasycznym (bez relacji czasowych). Nie ma to jednak znaczenia w przypadku analizy czasowej, której algorytm zostanie omówiony w kolejnym podrozdziale.

A.6. Algorytm RT

Analizę rozpoczynamy mając dany pierwszy węzeł (lub pierwsze węzły w przypadku replikacji bramek uogólniających AND) umieszczony na liście *LB*. W kolejnych krokach, węzły, które są analizowane trafiają na listę *LS*.

Lista *LS* przechowuje wyniki analizy. W dowolnym momencie procesu analizy, na bazie listy *LS* możemy zbudować fragment drzewa RT, a po jej zakończeniu – całe drzewo. Wyniki analizy dla przykładu systemu rozjazdu kolejowego znajdują się w rozdziale A.7., a graficzna reprezentacja jej fragmentu na rysunku 46.

A.6.1. POSTAĆ OGÓLNA

Algorytm RT obejmuje dwa kroki:

{K1: wyznaczenie pierwszego węzła i zapisanie na listę *LB*}

```
wyznacz_pierwszy_wezel; {Procedura wyznacza parametry dla pierwszego
                         węzła, czyli dla zdarzenia będącego
                         hazardem ze wzorów (30), (31). Przy czym,
                         jeśli pierwszą bramką jest bramka
                         uogólniająca AND i ulegnie replikacji, to
                         na listę należy wprowadzić również węzły
                         związane z powielonymi bramami}
```

{K2: przeprowadzenie obliczeń, rozpoczynając od pierwszego węzła}

```
inode(LS,LB);           {Wykonujemy obliczenia. Lista LB zawiera
                         węzeł startowy}
```

Obliczenia wykonywane są zgodnie z algorytmem opisanym w rozdziale 7.3.2. W dalszej części tego rozdziału znajduje się uszczegółowienie algorytmu opisanego właśnie we wspomnianej sekcji.

A.6.2. USZCZEGÓŁOWIENIE

```
procedure inode(LS, LB :tlist_LS_LB);

var
    ...              {część deklaracyjna zawiera zmienne pomocnicze,
                      ponieważ nie jest to istotne dla algorytmu
                      – zostanie ona pominięta. Sekcja ta, zawiera
                      również definicja wszystkich uszczegółowionych
                      procedur.}

begin
  while LB[0]>1 do   {Lista LB zawiera węzły, które nie zostały
                      poddane analizie. Jak to zostało zapisane
                      w algorytmie, w sekcji 7.3.2, analizę prowadzimy
                      dopóki LB nie jest pusta.
                          W prototypie narzędzia [Sk05] przyjęto
                      zasadę: pierwszy element listy zapisujemy na
                      pozycji 1; pozycja o numerze 0 przechowuje adres
                      pierwszej wolnej pozycji na liście. Jeśli
                      LB[0]=1, oznacza to, iż lista jest pusta.}
    begin
       n:=LB[1];     {n -pierwszy element listy}
       LB[0]:=LB[0]-1; {modyfikacja adresu wolnej pozycji}
```

```
LS[LS[0]]:=n; {Zapisanie węzła n na listę archiwum. Podobnie,
               jak w przypadku listy LB, LS[0] zawiera adres
               wolnej pozycji na liście LS}
LS[0]:=LS[0]+1; {Modyfikacja adresu wolnej pozycji}
for plb:=2 to LB[0] do LB[plb-1]:=LB[plb]; {przepisanie listy
               LB, jeśli na liście LB nie będzie elementów, to
               instrukcja się nie wykona}
```

Kolejna instrukcja wymaga komentarza. Przeprowadzając obliczenia metodą „z góry na dół", w każdym kroku obliczeń wykorzystujemy dane dotyczące jednego węzła. W praktyce, podczas analizy różnych przypadków zdarza się, iż liście *LB* zostaje umieszczony węzeł, który ma identyczne parametry (numer bramki oraz parametry czasowe) jak węzeł umieszczony wcześniej. Wówczas, przeprowadzając obliczenia uzyskamy takie same wyniki jak dla wcześniejszego węzła.

W celu uniknięcia niepotrzebnej analizy, która objęłaby całe poddrzewo wyników w RT, algorytm zaznacza takie węzły kolejnym numerem, lecz ze znakiem „-". Zatem wszystkie węzły ze znakiem „-" traktowane są jako te, dla których została już przeprowadzona analiza (możemy skorzystać z wyników przechowywanych w pamięci).

```
if n>0 then {jeśli mniejsze, to przypadek taki jest taki sam,
             jak wcześniej analizowany}
  begin

    RT.odczyt(n,nr_bramki,azs,bzs,aze,bze); {Odczytanie
                                             wyznaczonych
                                             parametrów zdarzenia
                                             w RT}

    ustal_poczatek_danych(nr_bramki,pozFT_CT,pozFTp);
                                             {Ustalenie początku
                                             danych
                                             w strukturach
                                             opisujących
                                             parametry drzewa
                                             niezdatności
                                             (pozFTp) oraz dane
                                             węzła w CT
                                             (pozFT_CT)}
```

```
          i:=pozFT_CT+1;          {Pozycja o adresie pozFT_CT+1
                                   wskazuje pierwszy lewy następnik
                                   bramki. Jeśli brama nie ma
                                   następnika, to na pozycji i znajduje
                                   się wartość 0.}

           case typ(nr_bramki) of {Wyznaczenie i zapisanie na listę LB
                                   następników analizowanego węzła RT.
                                   Obliczenia wykonujemy ze wzorów
                                   odpowiednich dla danej bramki.}
             LISC : begin end;     {Jeśli rozważane zdarzenie jest
                                   liściem, nie wykonujemy żadnych
                                   obliczeń}
             C_XOR: RT_CXOR; {dla bramki przyczynowej XOR}
             C_AND: RT_CAND; {dla bramki przyczynowej AND}
             G_AND: RT_GAND; {dla bramki uogólniającej AND}
           end;
         end;
      end;
  end;
```

A.6.3. USZCZEGÓŁOWIENIE PROCEDUR

```
procedure RT_CXOR;
begin
  pobierz_param_CXOR(pozFTp,asd1,bsd1,asd2,bsd2); {odczytanie
                                                   parametrów bramki
                                                   przyczynowej XOR}
```

W poniższej iteracji wyznaczamy kolejne węzły drzewa RT. Zgodnie z przyjętą strukturą drzewa CT następniki związane z lewym wejściem są zapisane ze znakiem „+”, natomiast związane z prawym – ze znakiem „-”. Zmienna *i*, jak to zostało wyjaśnione w opisie procedury *inode*, wskazuje na następniki analizowanego węzła.

```
  while FT_CT[i]>0 do
    begin
      odczytaj_bsxe(FT_CT[i]); {Odczytanie parametru β^S_xe; FT_CT[i]
                                -zawiera numer zdarzenia x}
```

```
    RT_CXOR_oblicz_L(asd1,bsd1,bsxe);
                              {Korzystając z parametrów: asd1,bsd1,
                               bsxe wyznaczamy parametry dynamiczne.
                               Obliczenia wykonujemy zgodnie ze
                               wzorem (32a). Przy czym, zgodnie ze
                               wzorem (32a), nowy przypadek otrzymamy
                               tylko wówczas, jeśli spełniona
                               zostanie zależność: asd1≤bsxe.
                                Wyznaczony przypadek zostanie
                               zapisany jako węzeł RT.}

    i:=i+1;
  end;

 while FT_CT[i]<0 do          {Przeprowadzamy analizę dla następników
                               związanych z prawym wejściem}
  begin
    odczytaj_bsye(-FT_CT[i]); {Odczytanie parametru β^S_ye; -FT_CT[i]
                               -numer zdarzenia y}
    RT_CXOR_oblicz_P(asd2,bsd2,bsye);
                              {Korzystając z parametrów: asd2,bsd2,
                               bsye wyznaczamy parametry dynamiczne.
                               Obliczenia wykonujemy zgodnie ze
                               wzorem (32b). Przy czym, zgodnie ze
                               wzorem (32b), nowy przypadek otrzymamy
                               tylko wówczas, jeśli spełniona
                               zostanie zależność: asd2≤bsye.
                                Wyznaczony przypadek zostanie
                               zapisany jako węzeł RT.}
    i:=i+1;
  end;
 end;


procedure RT_CAND;
begin
  pobierz_param_CAND(pozFTp,asd,bsd,asd2,bsd2); {odczytanie parametrów
                                                  bramki przyczynowej
                                                  AND}
```

Ponieważ do wykonania obliczeń dla bramki przyczynowej AND są potrzebne dwa zdarzenia wejściowe, należy przeanalizować jeden lub kilka przypadków.

Jeden przypadek ma miejsce wówczas, gdy do lewego i do prawego wejścia dołączone są tylko pojedyncze zdarzenia. Jeśli do któregoś wejścia bramki mamy dołączone więcej niż jedno zdarzenie (przypadek do analizy), to musimy rozważyć przypadki: „każde zdarzenie z lewego wejścia z każdym zdarzeniem z prawego wejścia". Dla tej bramki zawsze mamy, co najmniej, po jednym zdarzeniu „na każdym" wejściu.

```
k:=pozFT_CT+2;
while FT_CT[k]>0 do k:=k+1; {Ustawienie k na pierwszy element na
                             prawym wejściu –następniki dołączone do
                             prawego wejścia są zapisane w drzewie CT
                             ze znakiem „-"}

while FT_CT[i]>0 do
  begin
    j:=k;
    while FT_CT[j]<0 do
      begin                        {Odczytanie parametrów statycznych
                                    zdarzeń wejściowych: x i y}
        odczytaj_bsxe(FT_CT[i]);
        odczytaj_bsye(-FT_CT[i]);

            RT_CAND_oblicz(asd,bsd,bsxe,bsye); {Wykonanie obliczeń,
                                    zgodnie ze wzorem (33). W wyniku
                                    obliczeń, jeśli spełnione są
                                    zależności określone wzorem (33a),
                                    otrzymamy dwa przypadki. Pierwszy,
                                    ze wzoru (33b), drugi –(33c). Przy
                                    czym, przed zapisaniem drugiego
                                    przypadku następuje sprawdzenie,
                                    czy nie jest on taki sam jak
                                    pierwszy.}

        j:=j+1;
      end;
    i:=i+1;
  end;
end;
```

```
procedure RT_GAND;
  begin
    odczytaj_bsxe(FT_CT[i]);  {Odczytanie parametrów statycznych
                               zdarzenia x}
    odczytaj_bsye(-FT_CT[i]); {Odczytanie parametrów statycznych
                               zdarzenia y}
   RT_GAND_oblicz(bsxe,bsye); {Wykonanie obliczeń, zgodnie ze wzorem
                               (35). Przy czym, każdy wyznaczony
                               kolejno przypadek jest porównywany
                               z poprzednimi. W praktyce często
                               zdarza się tak, iż wyznaczony
                               przypadek jest takie sam, jak jeden
                               z poprzednich. Taka sytuacja ma często
                               miejsce, jeśli parametry: bsxe i bsye
                               mają wartość równą ∞}

  end;
```

A.6.4. RT – złożoność obliczeniowa

Założenie: niech liczba węzłów w drzewie CT, które są związane z bramami, wynosi *n*. Upraszczając, będziemy mówić, że liczba bram wynosi *n*.

Liczba *n* dla drzewa CT nie jest równa liczbie bram w drzewie niezdatności, oznaczonej wcześniej jako *m* (w wyniku replikacji może ich być więcej).

Zasadniczą częścią algorytmu RT jest procedura *inode.* Wykonywana jest ona dla każdego węzła i zawiera pewną stałą liczbą operacji. Ponieważ drzewo RT tworzone jest dynamicznie (podczas analizy), najgorszy przypadek będzie miał miejsce wtedy, gdy struktura drzewa CT będzie taka, że dla *n* - bram uzyskamy maksymalną liczbę przypadków do przeanalizowania.

Mając wzory (32), (33) oraz (35) – jednym z nich posługujemy się przy wyznaczeniu następników danego wierzchołka RT (patrz rozdział A.6.2 i A.6.3), zauważmy, że generują one maksymalnie, odpowiednio 2, 2 lub 4 (brama uogólniającej AND) przypadki, dla których prowadzić będziemy dalszą analizę.

Zatem, ze względu na złożoność obliczeniową najgorszy przypadek dla n bram w drzewie CT jest wówczas, gdy są to bramy uogólniające AND.

Możemy teraz oszacować liczbę przeprowadzanych obliczeń dla pesymistycznego przypadku.

TEZA: Liczba przypadków L podczas analizy drzewa CT zawierającego n bram uogólniających AND wynosi maksymalnie $L(n)=2^{2n}$.

DOWÓD:

1° Sprawdzamy prawdziwość wzoru dla $n=1$.

Dla $n=1$ mamy $L(1)=2^2=4$

Zgodnie ze wzorem (35) dla jednej bramy uogólniającej AND są cztery przypadki.

Zatem dla $n=1$ wzór jest prawdziwy.

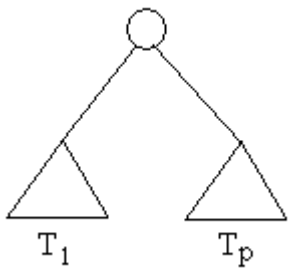
2°

Założenie indukcyjne: wzór jest prawdziwy dla $k\geq 1$ bram, zatem mamy $L(k)=2^{2k}$ przypadków.

Dodajmy jedną bramę (jeden węzeł) do drzewa CT, otrzymamy:

- T_l ma a bram, $a\leq k$
- T_p *ma* b bram, $b\leq k$
- $a+b=k$

gdzie T_l i T_p – odpowiednio, lewe i prawe poddrzewo (patrz rysunek A7) po dołożeniu nowego węzła.



Rys. A7. Drzewo CT po dołożeniu nowego węzła ($k+1$)

Dla dołożonego węzła (ponieważ zawiera bramę uogólniającą AND) mamy, ze wzoru (35) cztery przypadki. Liczba przypadków do przeanalizowania wynosi obecnie:

$$4 * L(k)$$

zatem:

$$4*2^{2k}=\underline{\underline{2^{2(k+1)}=L(k+1)}}$$

Z 1°, 2° i zasady indukcji wynika prawdziwość powyższego wzoru dla każdej liczby naturalnej $n \geq 1$.

Zatem pesymistyczna złożoność liczby przypadków jest $\circ(2^n)$.

Ponieważ każdy przypadek wymaga czasu procesora, to dla większej liczby bram (kilkudziesięciu) nie można będzie przeprowadzić obliczeń na jednym „procesorze" w rozsądnym czasie. Przy złożoności $\circ(2^n)$ i szybkości wykonywania instrukcji 13231 MIPS-ów, już dla $n=30$ czas obliczeń jest rzędu 1 dnia (ponad 22 godziny).

W praktyce takich przypadków jak drzewo CT złożone z samych bram uogólniających AND nie spotkamy, ale jeśli tak by się stało to:

1. Korzystając z „programowania[3] dynamicznego", poprzez sprawdzanie identyczności wyliczonych węzłów RT z poprzednimi znaczna część analizy może nie być potrzebna.

2. W najgorszym razie, korzystając z zasady „dziel i zwyciężaj", możemy przeprowadzić obliczenia równolegle na wielu procesorach lub wielu komputerach tworzących klaster.

W pierwszym przypadku, moglibyśmy do tego celu wykorzystać np. projekt superkomputera Sandia/Inlet (docelowo z 9072 procesorami Pentium Pro), który, jak podają twórcy w [Sandia], podczas testowania wykonał w ciągu 1 godziny i 20 minut 6,4 kwadryliona zmiennopozycyjnych operacji. Oczywiście, samo wykorzystanie nawet tak dużej mocy obliczeniowej nie wystarczyłoby do rozwiązania naszego problemu.

[3] Słowo „programowanie" w ujęciu metod optymalizacji odnosi się do sposobu zapisywania rozwiązań w tablicy i wykorzystywania ich w celu uproszczenia obliczeń nie ma wiele wspólnego z programowaniem rozumianym jako ogół czynności mających na celu opracowanie programu, szczegóły [Sysło97]

Jednak, jeśli nasz problem złożony z, załóżmy, *n=30* bram podzielimy na trzy problemy po *10* bram, to wówczas obliczenia przebiegną bardzo szybko.

W drugim - możemy do tego celu wykorzystać nawet komputery z „pracowni informatycznej", jeśli posiadają system operacyjny linux w wersji kompatybilnej, na przykład z oprogramowaniem OpenMosix. Szczegóły instalacji, użytkowania oraz źródła znajdziemy w [Mosix]. Oczywiście w jednym i drugim przypadku konstrukcja algorytmu musi umożliwiać równoległe przeprowadzanie obliczeń.

Algorytm podziału złożonego drzewa na prostsze przypadki może polegać na przykład na policzeniu czterech przypadków dla pierwszej bramy, a następnie uruchomieniu dla lewego i prawego poddrzewa CT (patrz rys. A4) osobnego procesu na innym komputerze/procesorze. Proces taki, jeśli będzie taka potrzeba, można powtórzyć na przykład kilkakrotnie – w zależności od *n*.

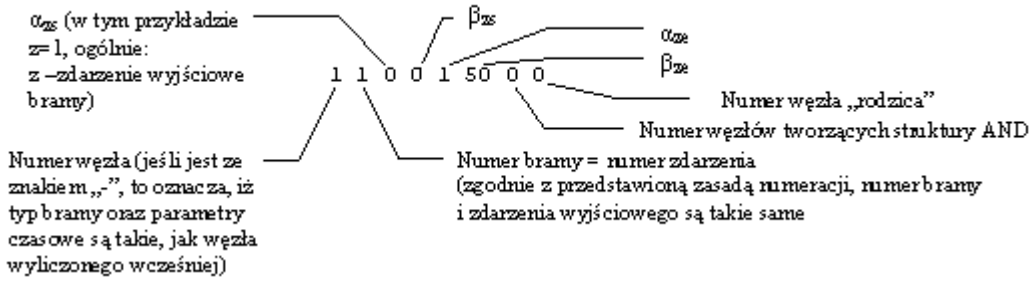
Wyrafinowany algorytm szacowania czasu analizy i podziału problemu na przypadki będzie stanowił dalszy etap badań nad prezentowaną metodą.

A.6.5. RT dla systemu rozjazdu kolejowego

Lista LS dla systemu rozjazdu kolejowego wygenerowana narzędziem [Sk05] ma następującą postać (dla przejrzystości, kolejne elementy rozdzielone są „/"):

```
    1 1 0 0 1 50 0 0 /2 2 -18 -10 0 ∞ 1 1 /3 6 -∞ -10 0 ∞ 1 1 /4 2 -∞
-10 0 ∞ 2 1 /5 6 -18 -10 0 ∞ 2 1 /-2 2 -18 -10 0 ∞ 3 1 /7 43 -∞ -10 0
∞ 3 1 /-4 2 -∞ -10 0 ∞ 4 1 /9 43 -18 -10 0 ∞ 4 1 /-2 2 -18 -10 0 ∞ 5 1
/11 44 -∞ -10 0 ∞ 5 1 /-4 2 -∞ -10 0 ∞ 6 1 /13 44 -18 -10 0 ∞ 6 1 /-2
2 -18 -10 0 ∞ 7 1 /15 45 -∞ -10 0 ∞ 7 1 /-4 2 -∞ -10 0 ∞ 8 1 /17 45
-18 -10 0 ∞ 8 1 /-2 2 -18 -10 0 ∞ 9 1 /19 7 -113 -10 0 103 9 1 /-4 2
-∞ -10 0 ∞ 10 1 /21 7 -18 -10 0 103 10 1 /22 4 -18 -10 -18 -10 0 2 /23
10 -18 -10 -18 -10 0 2 /24 11 -18 -10 -18 -10 0 2 /25 24 -∞ -10 0 ∞ 11
3 /26 26 -∞ -10 0 ∞ 11 3 /27 4 -∞ -10 -∞ -10 0 4 /28 10 -∞ -10 -∞ -10
0 4 /29 11 -∞ -10 -∞ -10 0 4 /30 24 -18 -10 0 ∞ 12 5 /-26 26 -∞ -10 0
∞ 12 5 /-25 24 -∞ -10 0 ∞ 13 5 /33 26 -18 -10 0 ∞ 13 5 /-25 24 -∞ -10
0 ∞ 14 7 /35 27 -∞ -10 0 ∞ 14 7 /-30 24 -18 -10 0 ∞ 15 9 /-35 27 -∞
-10 0 ∞ 15 9 /-25 24 -∞ -10 0 ∞ 16 9 /39 27 -18 -10 0 ∞ 16 9 /40 25 -∞
-10 0 ∞ 17 11 /-26 26 -∞ -10 0 ∞ 17 11 /42 25 -18 -10 0 ∞ 18 13 /-26
26 -∞ -10 0 ∞ 18 13 /-40 25 -∞ -10 0 ∞ 19 13 /-33 26 -18 -10 0 ∞ 19 13
/-40 25 -∞ -10 0 ∞ 20 15 /-35 27 -∞ -10 0 ∞ 20 15 /-42 25 -18 -10 0 ∞
21 17 /-35 27 -∞ -10 0 ∞ 21 17 /-40 25 -∞ -10 0 ∞ 22 17 /-39 27 -18
-10 0 ∞ 22 17 /52 8 -∞ -205 -18 ∞ 0 22 /53 9 -∞ -205 -18 ∞ 0 22 /54 20
-360 -204 -18 ∞ 0 23 /55 21 -327 -92 -18 ∞ 0 23 /56 22 -360 -205 -18 ∞
0 24 /57 8 -∞ -205 -∞ ∞ 0 27 /58 9 -∞ -205 -∞ ∞ 0 27 /59 20 -∞ -204 -∞
∞ 0 28 /60 21 -∞ -92 -∞ ∞ 0 28 /61 22 -∞ -205 -∞ ∞ 0 29 /62 40 -361
-204 -360 -203 0 54 /63 42 -440 -125 -327 -12 0 55 /64 40 -∞ -204 -∞
-203 0 59 /65 42 -∞ -125 -∞ -12 0 60 /
```

Opis pierwszego węzła listy przedstawionej powyżej znajduje się na rysunku A8 – zgodnie z notacją węzła przedstawioną w rozdziale A.3.



Rys. A8. Opis węzła RT

Taka reprezentacja, w celu ułatwienia interpretacji, może zostać poddana dalszej „obróbce". Fragment graficznej reprezentacji powstałej na bazie tej listy przedstawia rysunek 46. Narzędzie [Sk05] nie zostało jeszcze poszerzone o moduł dokonujący automatycznej transformacji z postaci tekstowej w postać graficzną.

A.7. Spis ilustracji

mgr inż. Paweł Skrobanek

Instytut Informatyki, Automatyki i Robotyki
Politechniki Wrocławskiej
ul. Janiszewskiego 11/17
50-372 Wrocław

Niniejszy raport otrzymują:

1. OINT	1 egz.
2. Biblioteka Główna PWR	1 egz.
3. Z-ca Dyrektora Instytutu	1 egz.
4. Promotor	1 egz.
5. Recenzenci	2 egz.
6. Autor	1 egz.
	Razem: 7 egz.

Raport wpłynął do Redakcji I-6
w sierpniu 2005 r.