

INSTYTUT CYBERNETYKI TECHNICZNEJ
POLITECHNIKI WROCŁAWSKIEJ

Raport serii: PREPRINTY Nr 16/83

JĘZYKOWE ALGORYTMY KOORDYNACJI
I KONTROLI W KOMPUTEROWO
WSPOMAGANYM PROJEKTOWANIU

/rozprawa doktorska/

Leszek Grocholski

Promotor:

prof.zw.dr inż.Jerzy BROMIRSKI

Słowa kluczowe: komputerowo wspoma-
gane projektowanie, proces
projektowania, biblioteka
programów

Wydawnictwo Politechniki Wrocławskiej
Wrocław 1983

SPIS TREŚCI

1. Wstęp	4
1.1. Uzasadnienie potrzeby nadzoru i koordynacji korzystania z oprogramowania użytkowego systemu KWP	4
1.2. Cel i teza pracy	5
1.3. Metodyka i układ pracy	6
1.4. Przegląd literatury	11
2. Specyfika projektowania wspomaganego	14
2.1. Charakter i struktura działań projektowych	14
2.2. Zasady działania systemu KWP	17
2.3. Sposób prowadzenia dialogu z komputerem	19
3. Model systemu KWP, zadania nadzoru i koordynacji	19
3.1. Założenia systemu KWP	19
3.2. Elementy systemu	21
3.3. Procesy wspomagane	29
3.4. Sformułowanie zadań nadzoru i koordynacji	30
3.5. Języki procesów wspomaganych	33
4. Rozwiązanie pierwszego zadania nadzoru i pierwszego zadania koordynacji	34
4.1. System generacyjny SG	34
4.2. Regularność języka $L_1/SG/$	35
4.3. Regularność języka $L_{PW}/KWP/$	38
4.4. Algorytm rozwiązania zadania Z1	39
4.5. Algorytm rozwiązania zadania Z3	43
5. Własności klas języków $L_{1/SG/}$ i $L_{PW/KWP/}$	45
5.1. Definicja RS automatu typu 1	45
5.2. Równoważność klas L_{R1} i $L_{1/SG/}$	45
5.3. Algebraiczne własności klasy L_{R1}	47
5.4. Algebraiczne własności klas języków $L_{1/SG/}$ i $L_{PW/KWP/}$	52
6. Rozwiązanie drugiego zadania nadzoru	53
6.1. Definicja języka $L_2/SG/$	53

6.2. Regularność języka $L_2/SG/$	53
6.3. Regularność języka $LPPW/KWP/$	56
6.4. Algorytm rozwiązania zadania Z_2	57
7. Własności klas języków $L_2/SG/$ i $L_{PPW/KWP/}$	60
7.1. Definicja RS automatu typu 2	60
7.2. Algebraiczne własności klasy L_{R_2}	62
7.3. Algebraiczne własności klas $L_2/SG/$ i $L_{PPW/KWP/}$	65
8. Rozwiązanie drugiego zadania koordynacji	66
8.1. Rozszerzony system generacyjny RSG	66
8.2. Regularność języka $L/RSG/$	66
8.3. Algebraiczne własności klasy L_{RSG}	68
8.4. Równoważność klas L_{RSG} i $L_{PWPP/KWP/}$	69
8.5. Algorytm rozwiązania zadania Z_4	70
9. Zakończenie	74
9.1. Metoda projektowania oprogramowania nadzoru- jącego i koordynującego	75
9.2. Podsumowanie wyników pracy	76
9.3. Kierunki dalszych badań	78
10. Bibliografia	80

Praca stanowi propozycję zastosowania aparatu lingwistyki formalnej do definiowania i badania języków służących do opisu zadań obliczeniowych w systemach komputerowego wspomaganie projektowania /KWP/. W sposób szczególny potraktowano zastosowanie tego aparatu do rozwiązywania zadań nadzoru rozumianych jako zadania porównywania opisów obliczeń z opisami dopuszczalnymi oraz zadań koordynacji polegających na wygenerowaniu /w celu ułatwienia korzystania z zasobów/ opisów dopuszczalnych.

Przy zastosowaniu konwencji formalno - lingwistycznej opis zadania obliczeniowego jest ciągiem symboli, zaś działanie systemu KWP modeluje RS automat. W dysertacji zdefiniowano język procesów wspomaganych /LPW/KWP// i języki pochodne oraz podano algorytmy syntezy akceptujących je RS automatów. Rozwiązania zadań nadzoru i koordynacji prowadzą się do rozwiązania zadań analizy lub syntezy odpowiednich języków do czego wykorzystywane są akceptujące je automaty.

Własności automatów i systemów generacyjnych akceptujących i generujących LPW/KWP/ oraz języki pochodne to również przedmiot rozważań rozprawy. Przedyskutowano - mające praktyczną interpretację - własności dynamiczne tzn. teoriomnogościowe i teoriojęzykowe klasy $\mathcal{L}_{PW/KWP/}$ i klas języków pochodnych.

Meritum dysertacji to opracowanie metody projektowania oprogramowania nadzorującego i koordynującego korzystanie z oprogramowania użytkowego systemu KWP.

1. WSTĘP

1.1. Uzasadnienie potrzeby nadzoru i koordynacji korzystania z programów użytkowych systemu KWP

Podstawowym elementem systemu komputerowego wspomaganie projektowania /w skrócie systemu KWP/ jest biblioteka programów użytkowych. Sposób działania każdego z programów wchodzących w skład biblioteki zdeterminowany jest przez rozkaz inicjujący wykonywanie przez niego obliczeń oraz typ struktury danych użytkownika.

Korzystanie przez projektanta z pojedynczego programu użytkowego, wspomagającego wykonywanie działania projektowego, prowadzi się do sporządzenia opisu zadania obliczeniowego. Opis składa się z rozkazu inicjującego działanie programu oraz danych użytkownika.

Wspomaganie procesów projektowych wymaga użycia wielu programów wspomagających wykonywanie działań składowych procesu.

Opis zadania obliczeniowego, stanowiącego zapis wspomaganie składa się w tym przypadku z ciągu rozkazów oraz ciągu danych.

Systemy KWP są coraz bardziej złożone. Biblioteki programów użytkowych składają się z kilkudziesięciu /systemy średnie [11] , [13] /, czy nawet kilkuset programów /systemy duże [12] , [58] /. Z każdym programem związanych jest kilka lub kilkanaście typów struktur danych.

Złożoność sytemów KWP oraz specyfika wspomaganie, a zwłaszcza:

- jego niejednorodność - współpraca projektanta z komputerem
- występowanie iteracji
- nie zawsze zdeterminowana struktura
- konieczność stosowania zrozumiałego dla komputera opisu zadań
- korzystanie z zawsze zawodnego sprzętu będącego przyczyną występowania przekłamań

powodują, że korzystanie z oprogramowania użytkowego systemu stwarza wiele okazji do popełniania błędów. Dlatego systemy te wyposażone są w specjalne oprogramowanie nadzorujące i koordynujące przebieg procesu wspomaganie. Oprogramowanie takie zapewnia, poprzez rozwiązywanie typowych zadań, wygodne i względnie niezawodne korzystanie z zasobów systemu.

W związku z tym, że decyduje ono o efektywności korzystania z systemu, opracowanie możliwie uniwersalnych, elastycznych i efektywnych metod nadzoru, a także koordynacji, jest jednym z podstawowych problemów metodyki projektowania systemów KWP. Znalazło to potwierdzenie w pracach [1] , [55] Komitetu Technicznego Nr 5 IFIP /zastosowanie komputerów w technice/.

1.2. Cel i teza pracy

Przyjmuje się następujące definicje pojęć nadzoru i koordynacji [45] .

Nadzór - kontrola bieżąca, która polega na porównaniu fragmentów działania z odpowiadającymi im wzorcami.

Koordinacja - "włączenie do działania elementów niezbędnych /powiązanych przyczynowo/ dla osiągnięcia zamierzonego wyniku w odpowiedniej ilości, jakości i czasie" lub "takie scalenie czynów wchodzących w skład działania złożonego by owe czyny sobie pomagały, a co najmniej nie przeszkadzały".

Zadania sprowadzające się do zadań porównania opisu obliczeń z opisami dopuszczalnymi, określonymi przez sposób działania systemu, zaliczane będą do klasy zadań nadzoru, a zadania polegające na wygenerowaniu /w celu ułatwienia korzystania z zasobów/ opisów dopuszczalnych do klasy zadań koordynacji.

Celem pracy jest opracowanie koncepcji i formalnych podstaw syntezy oprogramowania nadzorującego i koordynującego korzystanie z oprogramowania użytkowego systemu KWP.

Teza rozprawy jest następująca:

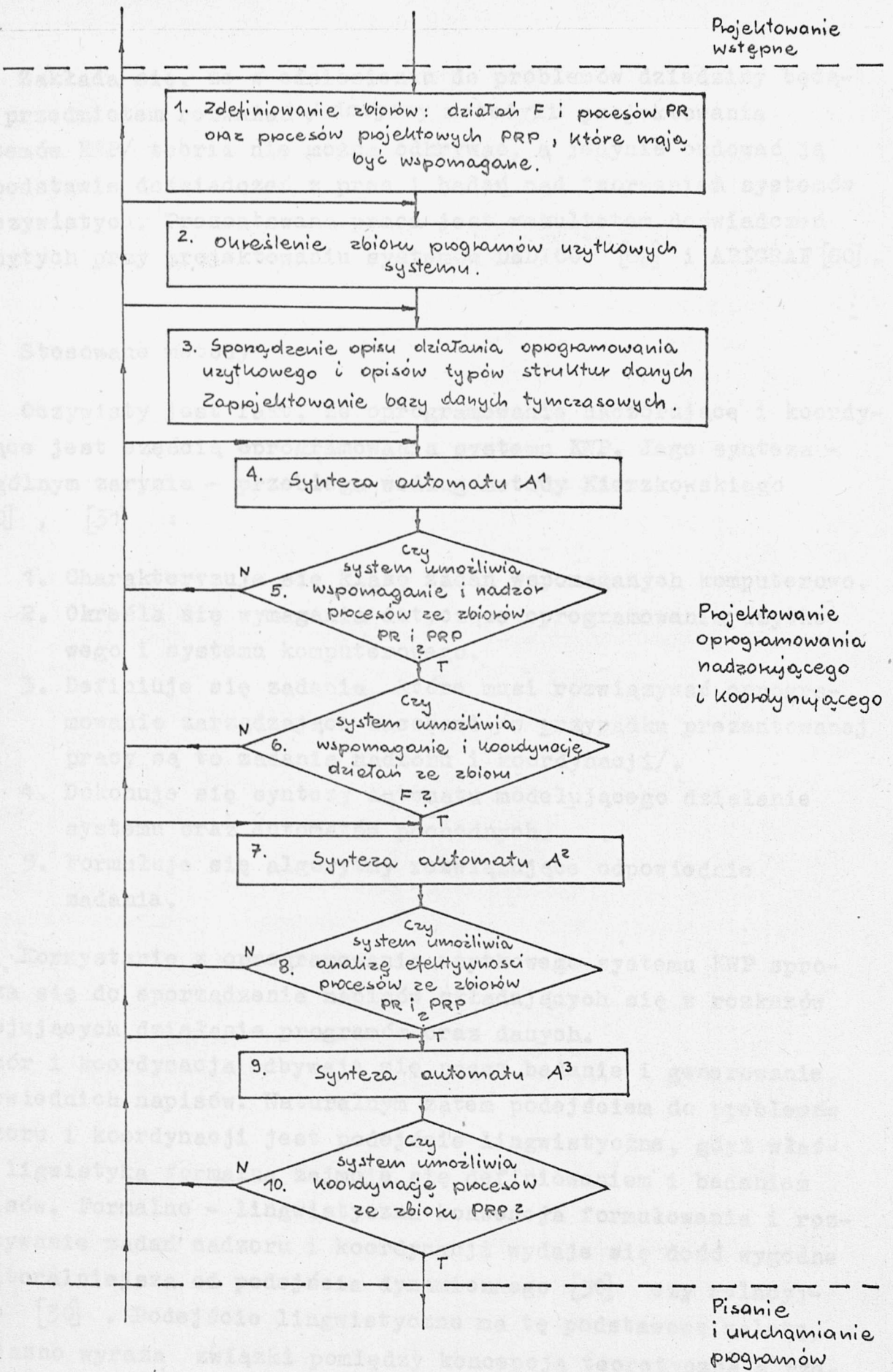
Przedstawione modele biblioteki programów użytkowych, działanie systemu KWP oraz dialogu projektanta z komputerem umożliwiają opracowanie efektywnych algorytmów koordynacji i nadzoru korzystania z oprogramowania użytkowego systemu KWP.

Podstawowym wynikiem przeprowadzonych w pracy rozważań jest metoda projektowania oprogramowania nadzorującego i koordynującego korzystanie z zasobów systemu KWP /rysunek 1.1/. Występujące na rysunku automaty A^1 , A^2 , A^3 są RS automatami modelującymi i analizującymi działanie systemu. W pracy podano algorytmy syntezy tych automatów i zastosowanie do rozwiązywania zadań nadzoru i koordynacji. W zakończeniu omówiono sposób realizacji pętli zwrotnych.

Metoda jest szczególnym przypadkiem /dotyczy oprogramowania nadzorującego i koordynującego/ metody Kierzkowskiego [30], [31] projektowania oprogramowania systemu KWP.

1.3. Metodyka i układ pracy

Przyjmuje się za Kierzkowskim [30], że budowana teoria tj. modele i wynikające z niej metody winny umożliwić rozwiązywanie zadań syntezy w realizowanych systemach KWP oraz opisywać zjawiska w nich zachodzące tak, aby cechy modeli możliwie wiernie odpowiadały właściwością systemów rzeczywistych.



Rys.1.1. Schemat metody projektowania oprogramowania nadzorującego i koordynującego korzystanie z programów użytkowych systemu KWP

Zakłada się, że w odniesieniu do problemów dziedziny będącej przedmiotem rozważań / dotyczy metodyki projektowania systemów KWP/ teorii nie można odkrywać, a jedynie budować ją na podstawie doświadczeń z prac i badań nad tworzeniem systemów rzeczywistych. Prezentowana praca jest rezultatem doświadczeń zdobytych przy projektowaniu systemów DEDICO [62] i APIGRAF [60].

Stosowane metody

Oczywisty jest fakt, że oprogramowanie nadzorujące i koordynujące jest częścią oprogramowania systemu KWP. Jego synteza - w ogólnym zarysie - przebiega według metody Kierzkowskiego [30], [31]:

1. Charakteryzuje się klasę zadań wspomaganych komputerowo.
2. Określa się wymagania dotyczące oprogramowania użytkowego i systemu komputerowego.
3. Definiuje się zadania, które musi rozwiązywać oprogramowanie zarządzające zasobami /w przypadku prezentowanej pracy są to zadania nadzoru i koordynacji/.
4. Dokonuje się syntezy automatu modelującego działanie systemu oraz automatów pochodnych.
5. Formułuje się algorytmy rozwiązujące odpowiednie zadania.

Korzystanie z oprogramowania użytkowego systemu KWP sprowadza się do sporządzenia napisów składających się z rozkazów inicjujących działanie programów oraz danych. Nadzór i koordynacja odbywają się przez badanie i generowanie odpowiednich napisów. Naturalnym zatem podejściem do problemów nadzoru i koordynacji jest podejście lingwistyczne, gdyż właśnie lingwistyka formalna zajmuje się definiowaniem i badaniem napisów. Formalno - lingwistyczna konwencja formułowania i rozwiązywania zadań nadzoru i koordynacji wydaje się dość wygodna i naturalniejsza od podejścia dynamicznego [50] czy relacyjnego [30]. Podejście lingwistyczne ma tę podstawową zaletę, że jasno wyraża związki pomiędzy koncepcją teoretyczną a rzeczywistością komputerową.

Formalizmy

Korzysta się z formalizmów: teorii mnogości, teorii automatów i lingwistyki matematycznej.

W książce Bliklego [6] znajdują się definicje wszystkich pojęć lingwistyki i teorii automatów niedefiniowanych w pracy / definicje: operacji teoriomnogościowych i teoriojęzykowych, RS automatu, zupełności, części aktywnej i pasywnej automatu, stanu zakazanego, funkcji M itd./. W celu uproszczenia rozważań zrezygnowano z symboli $\#$, $\#_i$ i $\#_p$.

Algorytmy zamieszczone w pracy to nieco skondensowane nieformalne wersje programów napisanych w Pascalu.

Zakres pracy

W rozdziale drugim omówiono: struktury procesu projektowania, sposoby zastosowań komputerów do wspomagania projektowania oraz cechy systemów KWP, które rozważono w dalszej części pracy.

Trzeci rozdział zawiera charakterystykę elementów systemu KWP: biblioteki programów użytkowych, struktur danych oraz bazy danych. Przedstawiono opis działania programów, sposób korzystania z oprogramowania oraz zasady działania systemu. Sformułowano definicję procesu wspomaganego, poprawnego procesu wspomaganego i poprawnego wspomaganego procesu projektowania oraz związanych z nimi:

- języka procesów wspomaganych - $\angle PW / KWP /$
- języka poprawnych procesów wspomaganych $\angle PPW / KWP /$
- języka poprawnych, wspomaganych procesów projektowania - $\angle PWPP / KWP /$.

W końcowej części rozdziału sformułowano zadanie nadzoru i koordynacji.

Przedmiotem rozważań dalszej części dysertacji są przede wszystkim systemy równoważne pod względem definiowania języka z modelem systemu KWP. Rozważania te prowadzą do syntezy automatów modelujących działanie systemu oraz sformułowania algorytmów rozwiązujących zadania nadzoru i koordynacji.

W rozdziale czwartym zdefiniowano system generacyjny SG i język typu 1 - $\mathcal{L}_1/\text{SG}/$ generowany przez ten system. Zbadano podstawowe własności języka $\mathcal{L}_1/\text{SG}/$ oraz przedstawiono algorytm syntezy automatu A^1 akceptującego ten język. Udowodniono, że klasa języków $\mathcal{L}_1/\text{SG}/$ może być uważana za równoważną klasie $\mathcal{L}_{\text{PW/KWP}}/$. Automat A^1 jest wykorzystywany do rozwiązania pierwszego zadania nadzoru i pierwszego zadania koordynacji. Rozdział kończą opisy algorytmów oraz obliczania maksymalnych czasów ich wykonania.

W rozdziale piątym przedstawiono definicję RS automatu typu 1 oraz przedyskutowano podstawowe własności algebraiczne klasy języków $\mathcal{L}_{R1}$. Klasa ta jest równoważna klasom $\mathcal{L}_1/\text{SG}/$ i $\mathcal{L}_{\text{PW/KWP}}/$. Udowodnione własności pozwalają twierdzić, że algorytmy syntezy automatu A^1 oraz algorytmy rozwiązania pierwszego zadania nadzoru i pierwszego zadania koordynacji mogą być stosowane w systemach KWP ulegających zmianom.

W rozdziale szóstym zdefiniowano język typu 2 - $\mathcal{L}_2/\text{SG}/$ generowany przez system generacyjny, zbadano jego podstawowe własności oraz przedstawiono algorytm syntezy RS automatu A^2 akceptującego $\mathcal{L}_2/\text{SG}/$. Automat A^2 wykorzystany został do rozwiązania drugiego zadania nadzoru.

W rozdziale siódmym zdefiniowano RS automat typu 2 oraz przedstawiono podstawowe własności algebraiczne klasy języków definiowanych przez takie automaty. Klasa ta jest równoważna klasom $\mathcal{L}_2/\text{SG}/$ i $\mathcal{L}_{\text{PPW/KWP}}/$.

W rozdziale ósmym zdefiniowano rozszerzony system generacyjny RSG oraz język $\mathcal{L}/\text{RSG}/$ generowany przez RSG. Pokazano, że klasa $\mathcal{L}_{\text{RSG}}$ jest równoważna klasie $\mathcal{L}_{\text{PWPP/KWP}}/$. Przedstawiono algorytm konstrukcji automatu A^3 akceptującego język $\mathcal{L}/\text{RSG}/$ / $\mathcal{L}_{\text{PWPP/KWP}}/$. Automat ten został wykorzystany do rozwiązania drugiego zadania koordynacji.

W rozdziale dziewiątym-stanowiącym zakończenie - zaproponowano, wynikającą z przedstawionych rozważań, ogólną metodę projektowania i uruchamiania oprogramowania nadzorującego i koordy-

nującego korzystanie z programów użytkowych systemu KWP. Dokonano zestawienia wyników rozprawy oraz wskazano kierunki dalszych badań nad oprogramowaniem organizacyjnym systemów KWP.

1.4. Przegląd literatury

Podana bibliografia obejmuje pozycje źródłowe, z których czerpie stosunkowo młoda metodologia i metodyka projektowania systemów KWP.

W literaturze znajdują się pozycje odnoszące się do:

- teorii formalnych, których formalizmy, pojęcia i metody stosowano w pracy [6] , [7] , [8] , [20]
- pragmatycznej metodologii projektowania [3] , [18] , [19] , [47]
- modeli systemów projektowania [15] , [16] , [30] , [44] , [61]
- metod i algorytmów nadzoru i koordynacji korzystania z oprogramowania systemów wspomagających projektowanie.

Odnosnie metod tworzenia oprogramowania nadzorującego i koordynującego, cytuje się prace opisujące funkcje, które to oprogramowanie powinno realizować oraz proponujące lub analizujące rozwiązania praktyczne. Za reprezentatywne dla tych problemów prace teoretyczne należy uznać: [26] , [27] , [30] , [31] , [35] , [52] , [56] ,

zaś za typowe rozwiązanie praktyczne - fragmenty oprogramowań systemów opisanych w: [10] , [12] , [21] , [23] , [36] , [39] , [43] , [46] , [58] , [59] .

Analiza prac teoretycznych i rozwiązań praktycznych pozwala ująć w postaci następującej listy obecny stan badań nad problemami omawianymi w pracy:

1. Ponieważ korzystanie z oprogramowania użytkowego sprowadza się do sporządzania napisów, opracowywane są specjalne języki służące do ich definiowania i analizy.

2. Wśród tych języków można wyróżnić:

a/ języki o charakterze operacyjnym, których rozkazy inicjują działanie odpowiednich części systemu /Najczęściej spotykanymi językami tego typu są tzw. języki komend /dyrektyw/ [28] , [30] , [31]/,

b/ języki opisu projektowanego obiektu, wśród których można wyszczególnić:

- języki przeznaczone do zapisu modeli matematycznych stosowanych w obliczeniach projektowych [11] , [22] , [39]
- języki zapisu zadań projektowych [30]
- języki ujmowania rozwiązań problemów projektowych [5] , [17]
- języki komunikowania się z bankiem danych [15] , [16] , [30]
- języki opisu technologii [9]
- języki do zapisu modeli symulacyjnych.

3. Wymienione języki są z reguły językami regularnymi i bezkontekstowymi [5] , [17] , [49] , [62], a rzadziej kontekstowymi.

4. Nadzór korzystania z oprogramowania polega na:

- sprawdzeniu czy rozkazowi odpowiada pewna część oprogramowania [35] , [46] , [52] , [56] , [57]
- kontroli możliwości wykonania akcji definiowanej przez użytkownika [10] , [13] , [30] , [32] , [62]
- analizie syntaktycznej, a czasami nawet semantycznej odpowiednich napisów [27] .

5. Zadania nadzoru rozwiązywane są przez moduły programów użytkowych lub przez specjalne analizatory, interpretatory czy kompilatory.

6. Koordynacja sprowadza się najczęściej do podpowiedzenia możliwych do podjęcia akcji i realizowana jest przez moduły programów użytkowych lub specjalne oprogramowanie systemowe.

7. Istnieje stosunkowo niewiele algorytmów rozwiązujących zadania nadzoru i koordynacji oraz metod projektowania odpowiedniego oprogramowania.

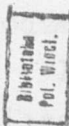
Najczęściej spotykanymi podejściami prowadzącymi do sformułowania algorytmów lub podania metod, są :

- podejście dynamiczne /w sensie teorii sterowania/ [50] posiadające małe znaczenie praktyczne
- podejście relacyjne [30] , [31] , które można sprowadzić do lingwistycznego
- podejście lingwistyczne [61] .

W języku polskim podstawową pozycją poświęconą projektowaniu oprogramowania systemu KWP jest rozprawa habilitacyjna Kierzkowskiego [30] .

Poniżej przedstawiono główne różnice między tą rozprawą, a prezentowaną pracą.

1. Rozprawa dotyczy całego oprogramowania, a praca jedynie oprogramowania nadzorującego i koordynującego.
2. W rozprawie stosuje się podejście relacyjne i lingwistyczne, a w pracy wyłącznie podejście lingwistyczne.
3. W rozprawie do rozwiązywania zadań nadzoru i koordynacji stosuje się 2 automaty - jeden analizujący poprawność ciągu programów, a drugi poprawność danych. W pracy do obu tych celów stosuje się jeden automat.
4. W pracy przedstawiono algorytmy syntezy oraz własności automatów modelujących i analizujących działanie systemu KWP.



2. SPECYFIKA PROJEKTOWANIA WSPOMAGANEGO

Podstawowe znaczenie dla komputerowego wspomagania projektowania mają : charakter i struktura działań projektowych, zasady działania systemu KWP oraz sposób prowadzenia dialogu z komputerem. Powyższe zagadnienia zostaną omówione w tej części pracy.

2.1. Charakter i struktura działań projektowych

Wspomaganie działań projektowych wymaga ich formalnego ujęcia. Poniżej zostanie nakreślony jeden z najczęściej spotykanych sposobów.

Założmy, że dany jest zbiór $P = \{p_1, \dots, p_i, \dots, p_n\}$ parametrów opisujących własności projektowanego obiektu. Wszystkie parametry $p_i \in P$ przyjmują wartości w obszarze ograniczeń

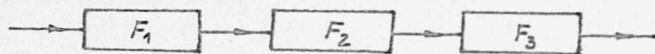
$$p_i = \{x_i : a_i \leq x_i \leq b_i ; x_i, a_i, b_i \in \mathbb{R}\}.$$

Wg. [47] iloczyn kortezjański zbiorów dopuszczalnych parametrów zmiennych tworzy przestrzeń zmiennych rozwiązania. Celem projektowania jest wyznaczenie pewnych położenia wektora p , które wyznaczają zbiór rozwiązań dopuszczalnych. Wyznaczenie położenia wektora p odbywa się stopniowo w wyniku podejmowania wielu działań.

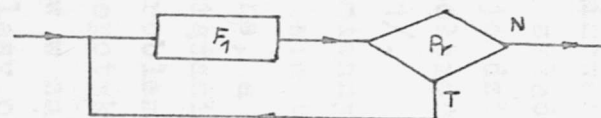
Sposób powiązania ze sobą działań projektowych oraz ich charakter opisuje struktura działań projektowych. Podstawowe cechy struktury determinują tzw. strategie projektowe [47], które określają sposoby i struktury zasadniczych sposobów podchodzenia do rozwiązania problemu projektowego.

Literatura wyróżnia podział strategii ze względu na następujące czynniki: postawę metodologiczną, stopień zaprogramowania i wzorzec poszukiwań. Wydaje się, że podstawowe znaczenie dla sposobu wspomagania komputerowego mają podziały strategii ze względu na 2 ostatnie czynniki.

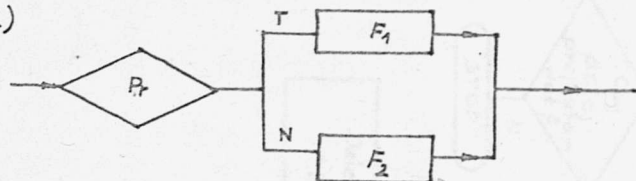
a)



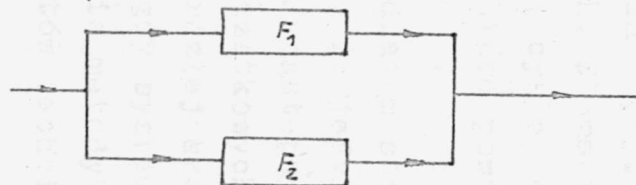
b)



c)



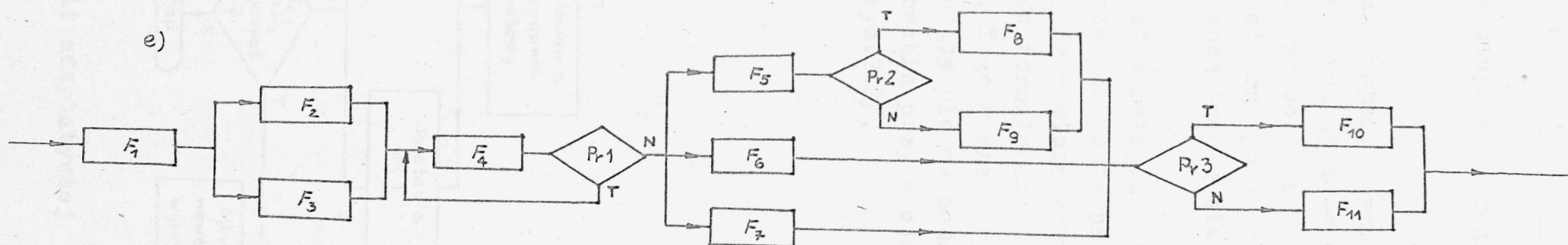
d)



Oznaczenia :

- F_i - działanie- Pr, Pr_i - predykat

e)

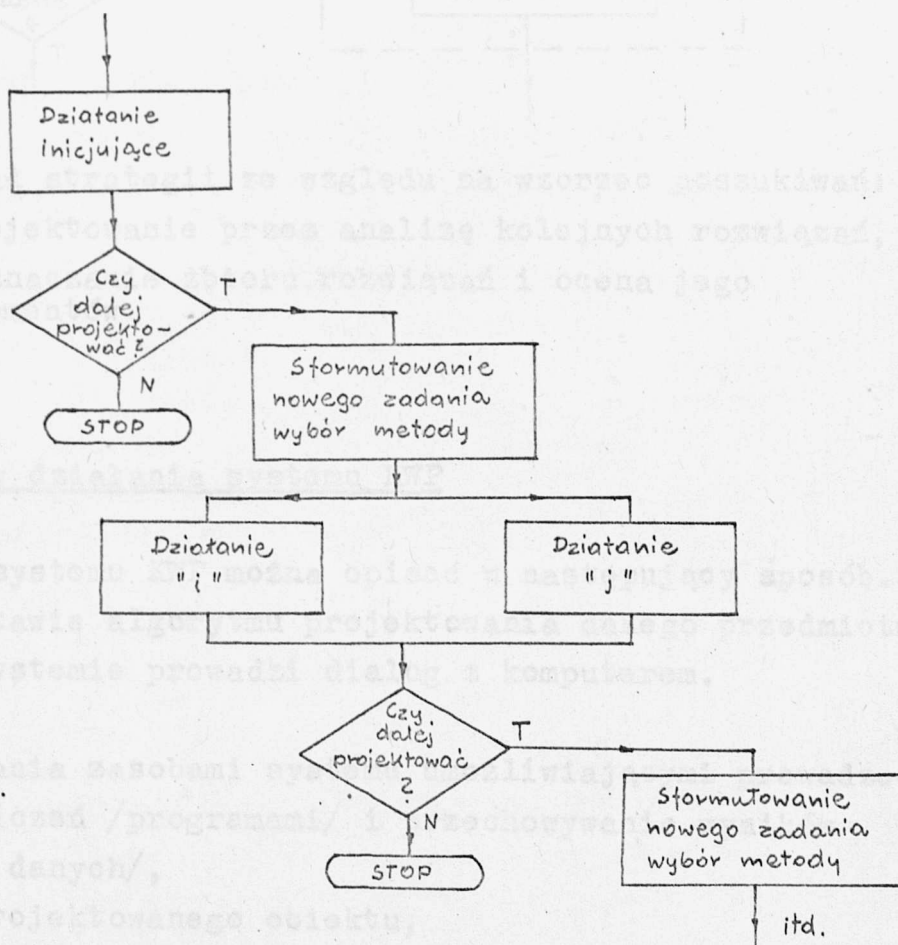


Rys. 2.1. Strategie zdeterminowane a/ liniowa b/ cykliczna
c/ rozgałęziona d/ współbieżna e/ przykładowa sieć
działań procesu projektowania

Podział strategii ze względu na stopień zaprogramowania działań

Najczęściej spotykanymi procesami wspomaganyymi są procesy zdeterminowane, dla których znane są rodzaj i kolejność działań składowych. Elementami takich procesów są liniowe sekwencje działań, cykle, działania alternatywne oraz działania współbieżne, które powiązane ze sobą tworzą sieć działań /rys.2.1/.

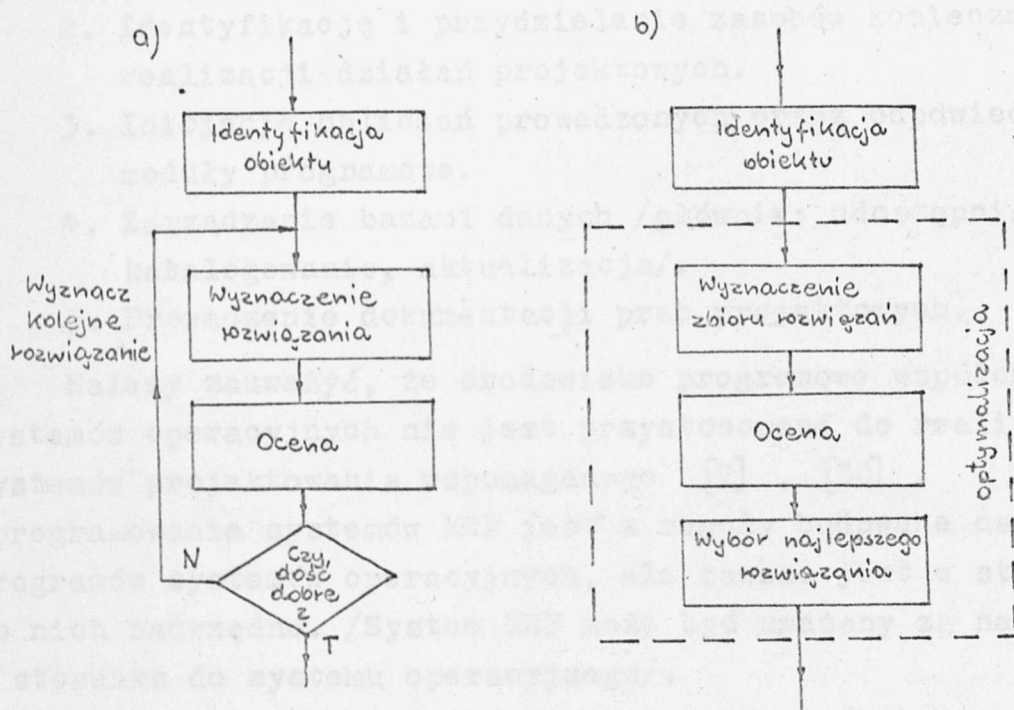
Odmiernym rodzajem strategii jest strategia adaptatywna. Zakłada się tutaj, że jedynie pierwsze działania projektowe są określone, a wybór następnych występuje w miarę rozwoju procesu. Po rozwiązaniu początkowych problemów formułowane i rozwiązywane są problemy bardziej szczegółowe. Tego rodzaju sytuacje są często spotykane gdy system projektujący nie ma doświadczeń i wzorów w zakresie metodyki postępowania przy projektowaniu danej klasy obiektów technicznych /rys.2.2/.



Rys.2.2. Ilustracja strategii adaptatywnej

Podział strategii ze względu na wzorzec poszukiwań

Istotę tego podziału wyjaśnia rys.2.3. Przedstawia on również typowe rodzaje działań występujących w procesie projektowania oraz podstawowe rodzaje podejmowanych decyzji.



Rys.2.3. Podział strategii ze względu na wzorzec poszukiwań:
a/ projektowanie przez analizę kolejnych rozwiązań,
b/ wyznaczanie zbioru rozwiązań i ocena jego elementów

2.2. Zasady działania systemu KWP

Działanie systemu KWP można opisać w następujący sposób.
Projekt na podstawie algorytmu projektowania danego przedmiotu oraz wiedzy o systemie prowadzi dialog z komputerem.
Dialog dotyczy:

- zarządzania zasobami systemu umożliwiającymi prowadzenie obliczeń /programami/ i przechowywanie wyników /bazami danych/,
- opisu projektowanego obiektu,
- sterowania przebiegiem pojedynczych obliczeń.

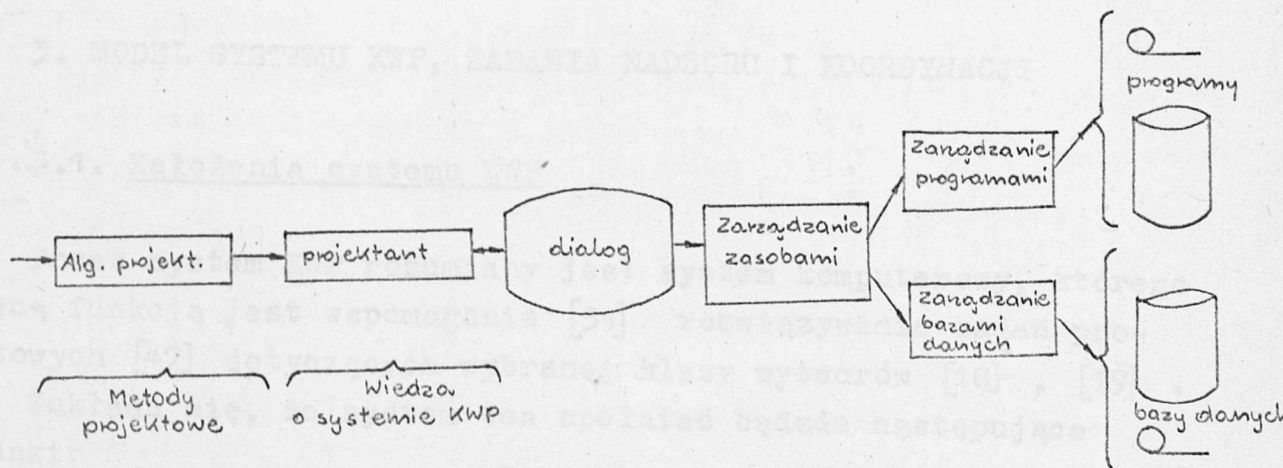
Oprogramowanie zarządzające na podstawie informacji uzyskanych od projektanta wykonuje wiele akcji mających na celu kontrolę dialogu i udostępnianie zasobów. Akcje te obejmują między innymi:

1. Kontrolę syntaktyczną, a także w pewnych przypadkach semantyczną dialogu.
2. Identyfikację i przydzielanie zasobów koniecznych do realizacji działań projektowych.
3. Inicjację obliczeń prowadzonych przez odpowiednie moduły programowe.
4. Zarządzanie bazami danych /głównie: udostępnianie, katalogowanie, aktualizacja/.
5. Prowadzenie dokumentacji prac projektowych.

Należy zauważyć, że środowisko programowe współczesnych systemów operacyjnych nie jest przystosowane do realizacji systemów projektowania wspomaganego [7] , [30] .

Oprogramowanie systemów KWP jest z reguły budowane na bazie programów systemów operacyjnych, ale zawsze jest w stosunku do nich nadrzędne. /System KWP może być uważany za nadsystem w stosunku do systemu operacyjnego/.

Na rysunku 2.4. przedstawiono schemat funkcjonowania systemu KWP [31] .



Rys.2.4. Schemat funkcjonowania systemu KWP

2.3. Sposób prowadzenia dialogu z komputerem [31]

W systemie KWP możliwe są 3 tryby dialogu z komputerem. Najprostszy sposób polega na realizacji obliczeń wg. określonego ściśle algorytmu i bez interwencji użytkownika w tok obliczeń. Zarządzanie przez użytkownika zasobami takiego systemu polega na określaniu sekwencji programów realizujących dany algorytm projektowy oraz sekwencji danych wymaganych przez te programy. Taki sposób nazywany jest często trybem planowanym. Drugi sposób realizacji obliczeń można nazwać trybem alternatywnym. Tryb ten umożliwia wstępne zestawienie - z możliwych modułów programowych - programu hipotetycznego. Projektant dynamicznie określa kolejny moduł obliczeniowy oraz zestaw parametrów wymaganych przez ten moduł. W trakcie obliczeń nie ma jednak możliwości ingerencji.

Trzeci sposób to tzw. tryb adaptatywny. Umożliwia on realizację ewolucyjnego procesu projektowania. Każdy kolejny etap /działanie i odpowiadający mu program, sterowanie obliczeń wykonywanych przez program/ wybierany jest po wykonaniu etapu poprzedniego. Do projektanta należy również ocena i podejmowanie decyzji dotyczących wyników etapu poprzedniego.

3. MODEL SYSTEMU KWP, ZADANIA NADZORU I KOORDYNACJI

3.1. Założenia systemu KWP

Przez system KWP rozumiany jest system komputerowy, którego główną funkcją jest wspomaganie [31] rozwiązywania zadań projektowych [47] dotyczących wybranej klasy wytworów [18] , [19] .

Zakłada się, że system ten spełniać będzie następujące warunki:

- Wspomaganie rozwiązywania zadań odbywa się etapowo poprzez realizację kolejnych działań projektowych. Większość działań zrutynizowanych wykonywanych jest przez programy użytkowe.

- Projektant inicjuje działanie programów, może sterować przebiegiem obliczeń przez nie wykonywanych oraz podejmować decyzje dotyczące dalszego przebiegu procesu.
- Celem projektowania jest rozwiązywanie zadania projektowego a następnie uzyskanie zestawienia dotyczącego otrzymanego rozwiązania. Zestawienie to ma formę dokumentacji projektowej i jest sporządzane przez specjalne programy systemu.
- Projektant nie jest zobowiązany do znajomości całego systemu. Wystarcza mu jedynie znajomość rozkazów inicjujących działania programów oraz struktur danych, które musi dostarczyć tym programom. Dane wprowadzane przez użytkownika są formalnie ujętymi parametrami dotyczącymi utworu [18] , [19] oraz strukturami sterującymi przebiegiem obliczeń.
- Opis wspomaganego przez system procesu rozwiązywania zadania składa się z ciągów rozkazów inicjujących działania programów oraz struktur danych użytkownika.
- Działanie programu zdeterminowane jest przez rozkaz inicjujący wykonywanie obliczeń oraz typ struktury danych.
- Składowanie, aktualizacja, przesyłanie struktur danych między programami odbywa się bez, lub przy minimalnej interwencji i wiedzy użytkownika.
- System jest otwarty - istnieje możliwość dodawania i usuwania programów.
- System nie utrudnia procesu projektowania, a wręcz przeciwnie - umożliwia używanie nazw działań i parametrów przyjętych w tradycyjnym projektowaniu, posiada mechanizmy kontrolujące przebieg procesu oraz ułatwia korzystanie z zasobów /podpowiada kolejne akcje, wyjaśnia działanie systemu/.
- Możliwe jest przetwarzanie informacji zarówno w trybie wsadowym jak i konwersacyjnym. Dialog z komputerem może przebiegać w sposób zaplanowany, alternatywny i adaptacyjny.

- System posiada cechy, które zgodnie z klasyfikacją Sielickiego [48] przedstawiono na rysunku 3.1.

3.2. Elementy systemu

Poniżej zostaną scharakteryzowane te elementy systemu KWP, które są ważne dla prezentowanych w pracy rozważań. Pominięte będą, jako mało istotne, takie elementy jak system cyfrowy i system operacyjny. W celu wyraźnego zaznaczenia i odróżnienia od podobnych pojęć związanych z systemami generowania ciągów, wszystkie symbole pojęć związanych z systemem KWP będą symbolami z daszkiem np. $\hat{f}_{in}$, $\hat{d}_x$, $\hat{t}_n$.

Dla użytkowników systemu KWP - profesjonalnych projektantów - najważniejsze znaczenie mają te elementy, które decydują o walorach eksploatacyjnych:

1. Programy wspomagające wykonywanie działań projektowych /tzw. programy użytkowe/.
2. Struktury danych wymaganych i generowanych przez programy ze szczególnym uwzględnieniem struktur wprowadzanych przez użytkowników.

Struktury danych

Niech $\mathcal{N}$ będzie skończonym zbiorem nazw a $\mathcal{V}$ skończonym zbiorem wartości.

Daną nazywa się parę /naz, var/ gdzie: naz $\in \mathcal{N}$, var $\in \mathcal{V}$. Przyjęto, że zbiór $\mathcal{N} \times \mathcal{V}$ rozbito na rozłączne podzbiory zwane typami danych. Dwie dane są tego samego typu ^(nazwy i) gdy ich wartości należą do tego samego podzbioru iloczynu kartezjańskiego $\mathcal{N} \times \mathcal{V}$.

Struktura danych to ciąg danych.

Przyjęto, że zbiór struktur danych rozbito na rozłączne podzbiory zwane typami struktur danych. Każdy typ struktury danych można zdefiniować podając zbiór warunków jakie muszą spełniać struktury, które do niego należą. Zbiór taki składa się z trzech części [51] :

1. Udział człowieka

- tryb zdeteminowany
- tryb alternatywny
- tryb adaptatywny

2. Operowanie danymi

- niezależne dla każdego programu
- przez pole informacyjne
- z udziałem baz danych

3. Repertuar działań

- obliczenia < proste
- obliczenia < wielowariantowe
- obliczenia i wytwarzanie dokumentacji
- obliczenia i sterowanie wytwarzaniem

4. Obszar tematyczny

- jeden wytwór
- klasa wytworów
- metoda inżynierska

5. Sprzęt

- kalkulatory
- proste komputery wsadowe
- systemy z podziałem czasu
- systemy z podziałem czasu
- wyposażone w urządzenia graficzne

6. Oprogramowanie

- programy użytkowe, języki algorytmiczne
- j.w. + system operacyjny
- j.w. + uniwersalne pakiety obliczeń projektowych i języki problemowo-zorientowane
- j.w. + języki dialogowe i języki komend
- złożone zintegrowane systemy programowe automatycznego projektowania

Rys.3.1. Drzewo cech systemów KWP z zaznaczonym zakresem tematycznym rozwiązywania rozważanego w pracy

- znaczeń elementów występujących w definicji,
- reguł gramatycznych /składniowych/,
- reguł semantycznych /w sensie matematycznym/.

Zarówno reguły gramatyczne jak i semantyczne można opisać przy pomocy pojęć teorii mnogości i rachunku predykatów [20], [53].

Zbiór warunków definiujących typ struktury danych tsd_x oznaczany będzie war_{tsd_x} .

Dwie struktury danych są tego samego typu, jeżeli spełniają ten sam zbiór warunków.

Jeżeli sd_i , sd_j są strukturami danych, to wynikiem ich złożenia jest struktura $sd_k = sd_i sd_j$.

W zbiorze struktur danych SD wyróżnia się strukturę pustą $\square$, którą można interpretować jako pusty ciąg danych. Typ struktury pustej oznaczany będzie $t_\square$. Zakłada się, że systemy $\{sd^*, \cup, \square\}$, $\{2^{sd}, \cup, \square\}$, $\{tsd^*, \cup, t_\square\}$, $\{2^{tsd}, \cup, t_\square\}$, gdzie TSD jest zbiorem typów struktur danych, są półgrupami z jednością.

Podane definicje są modyfikacjami definicji znanych w teorii struktur danych np. [14], [29], [53]. Niżej zostały odnotowane powody uzasadniające te modyfikacje:

1. Nie zakłada się unikalności nazw danych wchodzących w skład struktur, gdyż założenie takie nie wniosło by nic nowego do pracy i nie odpowiadało by sposobowi w jaki traktowane są struktury danych w systemach KWP.
2. W pracy podchodzi się do struktur danych głównie od strony wejścia/wyjścia programu, przechowywania ich w bazach danych oraz wprowadzania/ wyprowadzania przez urządzenia peryferyjne.
3. Modelem matematycznym struktury rekordowej, typowej dla operacji wejścia/wyjścia i opisu zawartości bazy danych, najczęściej jest zbiór. Stosowane w pracy pojęcie ciągu wydaje się bardziej uniwersalne oraz odpowiadające praktyce.

Zbiór struktur danych systemu KWP oznaczany będzie $\hat{SD} = \{ \hat{sd}_1, \hat{sd}_2, \dots \}$. Zbiór ten jest zbiorem ciągów danych stanowiących abstrakcyjne przedstawienie informacji niezbędnych do wspomaganie procesu oraz relacji i warunków zachodzących między

nimi. Struktury danych umożliwiają przechowywanie, dostęp i aktualizację informacji o utworze, ze szczególnym uwzględnieniem wzajemnej wymiany danych wejściowych i wyjściowych między programami.

W zbiorze typów struktur danych $\hat{TSD}$ systemu KWP można wyróżnić następujące podzbiory:

- TSD_S - podzbiór typów struktur odpowiadających katalogom i normom. Struktury takich typów przechowywane są w bazie danych stałych systemu /oznaczanej BDS/.
- TSD_M - podzbiór zawierający typy struktur danych uzyskiwanych w wyniku działania programów wspomagających projektowanie i przechowywanych w bazie danych tymczasowych systemu /oznaczanej BDT/.
- TSD_U - podzbiór typów struktur danych pochodzących od projektanta i wprowadzanych z urządzeń wejścia systemu. Struktury te są: "zbiorami kartowymi" zawierającymi opis utworu, parametrami i ograniczeniami wprowadzanymi w trakcie dialogu oraz strukturami sterującymi przebiegiem obliczeń.
- TSD_D - podzbiór typów struktur danych stanowiących dokumentację projektową i wyprowadzanych przez urządzenia wyjścia systemu. Zakłada się, że podzbiór TSD_D zawiera typ struktury pustej t_0 .

W pracy - jako nieistotne dla dalszych rozważań - pominięte zostaną struktury, których typy należą do podzbioru TSD_S .

Przez zbiór typów struktur danych systemu KWP $\hat{TSD} = \{t_{sd_1}, t_{sd_2}, \dots\}$ uważany będzie zbiór składający się z elementów podzbiorów TSD_M, TSD_U, TSD_D tzn., że:

$$\hat{TSD} = TSD_M \cup TSD_U \cup TSD_D.$$

Należy zauważyć, że podzbiory TSD_M, TSD_U, TSD_D nie muszą być parami rozłączne.

Przyjęto, że w danym momencie w bazie danych tymczasowych BDT mogą znajdować się wyłącznie struktury danych różnych typów. Przyjęcie takiego założenia pozwala sformułować następującą definicję stanu systemu KWP:

Stanem systemu KWP nazywany będzie zbiór typów struktur danych znajdujących się w BDT.

Stan systemu określa możliwe akcje użytkownika.

Zbiór stanów systemu KWP oznaczany będzie $\hat{S} = \{\hat{s}_0, \hat{s}_1, \dots\}$. Zbiór $\hat{S}$ spełnia oczywiście warunek: $\forall \hat{s}_x \in \hat{S} / \hat{s}_x \in 2^{\text{TSD}_M} /$. Stan $\hat{s}_0 = \emptyset$ nazywany będzie stanem początkowym systemu.

Opis działania oprogramowania użytkowego

Przyjęto, że programy wchodzące w skład oprogramowania użytkowego są programami wielowariantowymi tzn., że w zależności od sposobu ich użycia mogą dostarczać wyniki różnego typu.

Przyjęto również, że sposób przeprowadzania przez program obliczeń zdeterminowany jest przez rozkaz inicjujący wykonywanie przez niego obliczeń /zwany dalej krótko - rozkazem/ oraz typ struktury danych dostarczanych przez użytkownika /struktury, której typ należy do podzbioru $\hat{\text{TSD}}_U$ / wprowadzanych przez urządzenia wejścia komputera.

Ponieważ wielowariantowe obliczenia wykonywane przez wielowariantowe programy mogą wspomagać wykonywanie kilku działań założono, że pojedynczemu programowi może odpowiadać kilka rozkazów

Przyjęto, że dla danego programu $\hat{p}_x$ rozkaz oraz typ struktury danych dostarczanych przez użytkownika determinują:

- typy struktur danych wejściowych programu znajdujących się w BDT systemu
- typy struktur danych wyjściowych programu wyprowadzanych do BDT
- typy struktur danych przechowywanych w BDT ulegające zniszczeniu po zakończeniu wykonywania przez $\hat{p}_x$ obliczeń
- typy struktur danych, których obecność w BDT uniemożliwia użycie $\hat{p}_x$
- typ dokumentacji projektowej sporządzanej przez program $\hat{p}_x$.

W dalszej części pracy rozważa się jedynie te programy, dla których przyjęte założenie jest prawdziwe, pomija się natomiast programy, dla których niemożliwe jest określenie na podstawie sposobu użycia, typów wyników oraz zmian zawartości bazy danych.

Formalnie działanie programów wchodzących w skład oprogramowania użytkowego opisuje dziesiątka $\hat{O}DO_{KWP}$ /opis działania oprogramowania użytkowego systemu KWP / . $\hat{O}DO_{KWP} =$

$$= \{ \hat{P}, \hat{C}, \hat{TSD}, \hat{Dz}_{KWP}, \hat{f}_{dek}, \hat{f}_{in}, \hat{f}_{out}, \hat{f}_{kon}, \hat{f}_{dem}, \hat{f}_{dok} \}, \quad \text{gdzie:}$$

$\hat{P}$ - zbiór programów użytkowych $-\hat{P} = \{ \hat{p}_1, \hat{p}_2, \dots \},$

$\hat{C}$ - zbiór rozkazów $-\hat{C} = \{ \hat{c}_1, \hat{c}_2, \dots \},$

$\hat{TSD}$ - zbiór typów struktur danych $-\hat{TSD} = \{ \hat{tsd}_1, \hat{tsd}_2, \dots \},$

$$\hat{TSD} = \hat{TSD}_M \cup \hat{TSD}_U \cup \hat{TSD}_D,$$

$\hat{Dz}_{KWP} \subseteq \hat{C} \times \hat{TSD}_U$ - dziedzina $\hat{O}DO_{KWP}$,

$\hat{f}_{dek} : \hat{C} \rightarrow \hat{P}$ - funkcja przyporządkowująca rozkazom odpowiadające im programy,

$\hat{f}_{in} : \hat{Dz}_{KWP} \rightarrow 2^{\hat{TSD}_M}$ - funkcja przyporządkowująca parom $\langle \text{rozkaz, typ struktury danych użytkownika} \rangle$ typy struktur danych wejściowych przechowywanych w BDT,

$\hat{f}_{out} : \hat{Dz}_{KWP} \rightarrow 2^{\hat{TSD}_M}$ - funkcja przyporządkowująca parom $\langle \text{rozkaz, typ struktury danych użytkownika} \rangle$ typy struktur danych wyjściowych przechowywanych w BDT,

$\hat{f}_{dem} : \hat{Dz}_{KWP} \rightarrow 2^{\hat{TSD}_M}$ - funkcja przyporządkowująca parom $\langle \text{rozkaz, typ struktury danych użytkownika} \rangle$ typy struktur danych przechowywanych w BDT, które zostają zniszczone w wyniku wykonania obliczeń przez nie zdeteminowanych,

$\hat{f}_{kon} : \hat{Dz}_{KWP} \rightarrow 2^{\hat{TSD}_M}$ - funkcja przyporządkowująca parom $\langle \text{rozkaz, typ struktury danych użytkownika} \rangle$ typy struktur danych, których obecność w BDT uniemożliwia wykonanie obliczeń przez nie zdeteminowanych,

$\hat{f}_{dok} : \hat{Dz}_{KWP} \rightarrow TSD_D$ - funkcja przyporządkowująca parom $\langle \text{rozkaz, typ struktury danych użytkownika} \rangle$ dokumentację projektową dostarczaną przez program $\hat{f}_{dek}/\text{rozkaz}/$

Należy zauważyć, że struktury danych użytkownika oraz dokumentacja są spójnymi lub co najwyżej poprzedzielanymi komunikatami systemu ciągami danych, co powoduje, że w ODO_{KWP} operuje się pojedynczymi typami struktur wyprowadzanych przez użytkownika oraz dokumentacji. Struktury takiego typu mogą oczywiście składać się z wielu struktur składowych. Np. struktura danych wprowadzana przez użytkownika może składać się ze struktur sterujących przebiegiem obliczeń oraz opisujących utwór.

Struktury danych przechowywane w bazie danych tymczasowych mogą znajdować się w różnych miejscach i być oddzielone innymi strukturami danych co powoduje, że w ODO_{KWP} funkcje $\hat{f}_{in}$, $\hat{f}_{out}$, $\hat{f}_{kon}$, $\hat{f}_{dem}$ jako swoje wartości przyjmują podzbiory typów struktur danych.

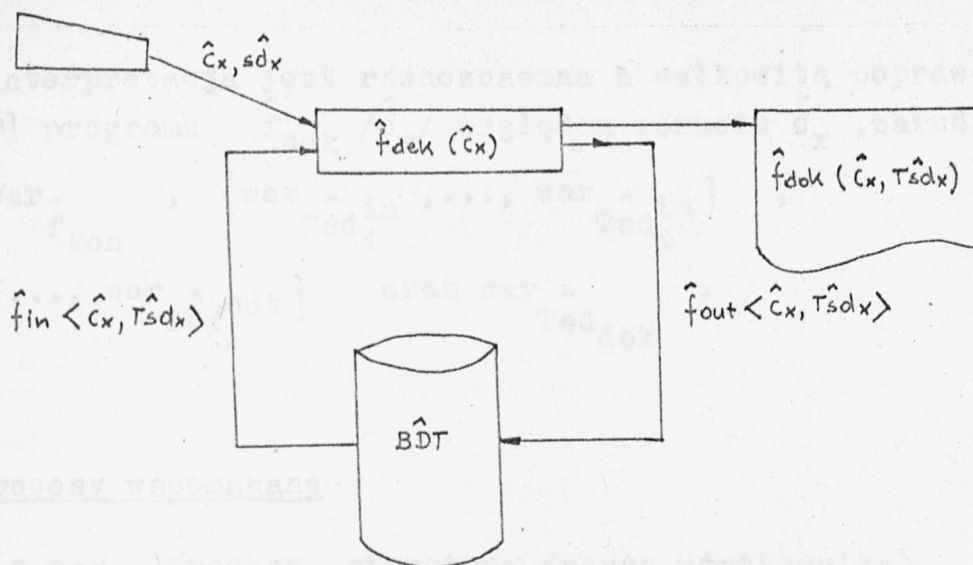
Przez Tsd_x oznaczany będzie typ struktury $\hat{s}d_x$.

Niech $\langle \hat{c}_x, Tsd_x \rangle$ należy do dziedziny ODO_{KWP} .

Korzystając z opisu działania oprogramowania ODO_{KWP} można w następujący sposób scharakteryzować obliczenia zdeterminowane przez parę $\langle \hat{c}_x, Tsd_x \rangle$:

- obliczenia wykonywane są przez program $\hat{f}_{dek}/\hat{c}_x/$
- obliczenia można wykonywać jeżeli w BDT nie znajduje się żadna struktura, której typ należy do $\hat{f}_{kon} \langle \hat{c}_x, Tsd_x \rangle$ oraz znajdują się wszystkie struktury danych o typach określonych przez $\hat{f}_{in} \langle \hat{c}_x, Tsd_x \rangle$
- obliczenia kończą się wprowadzeniem do BDT struktur o typach określonych przez $\hat{f}_{out} \langle \hat{c}_x, Tsd_x \rangle$ oraz wyprowadzeniem przez urządzenia wyjścia dokumentacji /struktur typu $\hat{f}_{dok} \langle \hat{c}_x, Tsd_x \rangle /$
- wprowadzaniu do BDT wyników towarzyszy niszczenie struktur typu $\hat{f}_{dem} \langle \hat{c}_x, Tsd_x \rangle$.

Na rysunku 3.3 przedstawiono schemat obliczeń zdeterminowanych przez parę $\langle \hat{c}_x, Tsd_x \rangle$.



Rys.3.3. Schemat obliczeń zdeterminowanych przez parę $\langle \hat{c}_x, \hat{T}_{sd_x} \rangle$

Scharakteryzowana w początkowej części podrozdziału możliwość opisu typu struktury danych - przy pomocy zbioru warunków - pozwala na przedstawioną poniżej interpretację obliczeń zdeterminowanych przez parę $\langle \hat{c}_x, \hat{T}_{sd_x} \rangle$.

Jeżeli:

- struktura danych użytkownika spełnia warunek $\text{war}_{\hat{T}_{sd_x}}$
- w $\hat{B}DT$ nie znajduje się żadna struktura o typie należącym do $\hat{f}_{kon} \langle \hat{c}_x, \hat{T}_{sd_x} \rangle$ /warunek ten oznaczany będzie $\text{war}_{\hat{kon}}$ /
- w $\hat{B}DT$ znajdują się struktury $\{\hat{s}_{d_1}^{in}, \dots, \hat{s}_{d_i}^{in}\}$ takie, że $\hat{f}_{in} \langle \hat{c}_x, \hat{T}_{sd_x} \rangle = \{\hat{T}_{sd_1}^{in}, \dots, \hat{T}_{sd_i}^{in}\}$ /spełniające warunki $\{\text{war}_{\hat{T}_{sd_1}^{in}}, \dots, \text{war}_{\hat{T}_{sd_i}^{in}}\}$ /,

to program $\hat{f}_{dek} / \hat{c}_x /$ zakończy obliczenia i dostarczy wyniki w postaci struktur $\{\hat{s}_{d_1}^{out}, \dots, \hat{s}_{d_j}^{out}\}$ takich, że

$\hat{f}_{out} \langle \hat{c}_x, \hat{T}_{sd_x} \rangle = \{\hat{T}_{sd_1}^{out}, \dots, \hat{T}_{sd_j}^{out}\}$ /spełniających warunki $\{\text{war}_{\hat{T}_{sd_1}^{out}}, \dots, \text{war}_{\hat{T}_{sd_j}^{out}}\}$ / oraz struktury $\hat{s}_{d_{dok}}$ takiej, że $\hat{f}_{dok} \langle \hat{c}_x, \hat{T}_{sd_x} \rangle = \hat{T}_{sd_{dok}}$ /spełniającej warunek $\text{war}_{\hat{T}_{sd_{dok}}}$ /.

Taka interpretacja jest równoznaczna z całkowitą poprawnością [38] programu $\hat{f}_{dek} / \hat{c}_x /$ względem rozkazu $\hat{c}_x$, warunków $\text{war}_{\hat{Tsd}_x}$, $\text{war}_{\hat{f}_{kon}}$, $\{\text{war}_{\hat{Tsd}_1^{in}}, \dots, \text{war}_{\hat{Tsd}_i^{in}}\}$, $\{\text{war}_{\hat{Tsd}_1^{out}}, \dots, \text{war}_{\hat{Tsd}_j^{out}}\}$ oraz $\text{war}_{\hat{Tsd}_{dok}}$.

3.3 Procesy wspomagane

Każda z par $\langle \text{rozkaz, struktura danych użytkownika} \rangle$, takich, że para $\langle \text{rozkaz, typ struktury danych użytkownika} \rangle$ należy do Dz_{KWP} , określa pewne działanie wspomagane i dlatego będzie nazywana działaniem wspomagany.

System umożliwia wykonanie ciągu działań wspomaganych, opisanych przez ciąg rozkazów $\hat{x} = \hat{c}_1 \dots \hat{c}_i \dots \hat{c}_n$, ciąg struktur danych użytkownika $\hat{y} = \hat{s}_1 \dots \hat{s}_i \dots \hat{s}_n$, jeżeli spełniony jest następujący warunek:

Istnieje ciąg stanów systemu $\hat{z} = \hat{s}_0 \hat{s}_1 \dots \hat{s}_i \dots \hat{s}_{n-1}$ taki, że:

Wr.3.1. $\hat{s}_0 = \emptyset$ / $\hat{s}_0$ jest stanem początkowym /,

Wr.3.2. / $\forall \langle \hat{c}_i, \hat{s}_i \rangle \quad 0 < i \leq n / \quad / \hat{f}_{in} \langle \hat{c}_i, \hat{Tsd}_i \rangle \subseteq \hat{s}_{i-1} \wedge$
 $\wedge \hat{f}_{kon} \langle \hat{c}_i, \hat{Tsd}_i \rangle \cap \hat{s}_{i-1} = \emptyset \wedge \hat{s}_i = \hat{s}_{i-1} - \hat{f}_{dem}$
 $\langle \hat{c}_i, \hat{Tsd}_i \rangle \cup \hat{f}_{out} \langle \hat{c}_i, \hat{Tsd}_i \rangle /$.

Para ciągów $\langle \hat{x}, \hat{y} \rangle$, gdzie $\hat{x} = \hat{c}_1 \dots \hat{c}_i \dots \hat{c}_n$ jest ciągiem rozkazów, a $\hat{y} = \hat{s}_1 \dots \hat{s}_i \dots \hat{s}_n$ ciągiem struktur danych użytkownika, dla której spełnione są warunki Wr.3.1, Wr.3.2 nazywana będzie procesem wspomagany.

Niech $\langle \hat{x}, \hat{y} \rangle$, gdzie $\hat{x} = \hat{c}_1 \dots \hat{c}_i \dots \hat{c}_n$ i $\hat{y} = \hat{s}_1 \dots \hat{s}_i \dots \hat{s}_n$ będzie procesem wspomagany. Działanie $\langle \hat{c}_i, \hat{s}_i \rangle \quad 1 \leq i < n$ jest zbędne jeżeli spełniony jest następujący warunek:

Wr.3.3. / $\exists \hat{Tsd}_i^{out} \in \hat{f}_{out} \langle \hat{c}_i, \hat{Tsd}_i \rangle / \quad // \forall \langle \hat{c}_j, \hat{Tsd}_j \rangle \quad i < j \leq n /$
 $/ \hat{Tsd}_i^{out} \notin \hat{f}_{in} \langle \hat{c}_j, \hat{Tsd}_j \rangle //$

Innymi słowy działanie $\langle \hat{c}_i, \hat{s}d_i \rangle$ jest zbędne, jeżeli dostarcza wyniki, które nie są wykorzystywane jako dane wejściowe przez żadne z następnych działań.

Niech $\langle \hat{x}, \hat{y} \rangle$ będzie procesem wspomaganym. Proces $\langle \hat{x}, \hat{y} \rangle$ jest poprawnym procesem wspomaganym, jeżeli każde z działań wspomaganych $\langle \hat{c}_i, \hat{s}d_i \rangle$ $1 \leq i < n$ nie jest zbędne.

Celem projektowania jest uzyskanie dokumentacji projektowej. Za proces projektowania może być zatem uważany proces, który ją dostarcza. Rezultatem takiego rozumowania jest następująca definicja:

Poprawny proces wspomagany $\langle \hat{x}, \hat{y} \rangle$ dla którego spełniony jest warunek $\hat{f}_{\text{dok}} \langle \hat{c}_n, \hat{T}sd_n \rangle \neq \hat{t}n$ nazywany będzie poprawnym wspomaganym procesem projektowania.

3.4. Zadania nadzoru i koordynacji

Poniżej zostaną scharakteryzowane typowe problemy, z którymi spotykają się użytkownicy systemu KWP oraz sformułowane zadania, które powinny być rozwiązywane przez oprogramowanie nadzorujące i koordynujące korzystanie z programów użytkowych.

Problem 1

Użytkownik przygotowuje poprzez zbudowanie ciągu rozkazów $\hat{x} = \hat{c}_1 \dots \hat{c}_i \dots \hat{c}_n$ i ciągu struktur danych $\hat{y} = \hat{s}d_1 \dots \hat{s}d_i \dots \hat{s}d_n$ opis procesu, którego wspomaganie planuje. Należy sprawdzić czy ciągi $\hat{x}$, $\hat{y}$ gwarantują poprawne wykonanie obliczeń. Rozstrzygnięcie tego problemu sprowadza się do rozwiązania następującego zadania:

Zadanie Z1

Dane: Ciąg rozkazów $\hat{x} = \hat{c}_1 \dots \hat{c}_i \dots \hat{c}_n$,

ciąg struktur danych $\hat{y} = \hat{s}d_1 \dots \hat{s}d_i \dots \hat{s}d_n$.

Szukane: Stwierdzenie - czy para $\langle \hat{x}, \hat{y} \rangle$ jest procesem wspomaganym?

Problem 2

Często zdarza się, że na skutek nieznamomości zasad działania systemu lub nadmiernej gorliwości generowane przez

użytkowników procesy wspomagane zawierają działania zbędne. Sytuacja taka prowadzi do oczywistych strat ekonomicznych. Odpowiedni program powinien zatem rozwiązywać zadanie Z2.

Zadanie Z2

Dane: Para $\langle \hat{x}, \hat{y} \rangle$.

Szukane: Stwierdzenie - czy $\langle \hat{x}, \hat{y} \rangle$ jest poprawnym procesem wspomaganiem?

Problem 3

Dialog z komputerem jest dla większości początkujących projektantów sytuacją stresową. Nie więc dziwne, że w czasie bezpośredniego kontaktu z komputerem - a zwłaszcza w sytuacjach niestandardowych - zapominają oni o regułach dialogu i o swoich planach. Oprogramowanie systemu powinno w takiej sytuacji wygenerować opisy wszystkich możliwych do wykonania działań wspomaganych, tzn. pary $\langle \text{rozkaż, typ struktury danych użytkownika} \rangle$ i w ten sposób problem projektantowi odnaleźć się. Rozstrzygnięcie tego problemu sprowadza się do rozwiązania następującego zadania:

Zadanie Z3

Dane: Stan systemu $\hat{s}_x$.

Szukane: Zbiór wszystkich par $\langle \hat{c}, \hat{t}_{sd} \rangle$ takich, że $\hat{f}_{in} \langle \hat{c}, \hat{t}_{sd} \rangle \subseteq s_x$ i $\hat{f}_{kon} \langle \hat{c}, \hat{t}_{sd} \rangle \cap s_x = \emptyset$.

Problem 4

Użytkownikowi systemu KWP największy problem sprawia odpowiedź na pytanie: jakie struktury danych musi przygotować i przy pomocy jakich programów wprowadzić oraz jakie programy użyć aby uzyskać, w postaci dokumentacji projektowej typu t_{sd_dok} , rozwiązanie zadania projektowego? Problem ten prowadzi do sformułowania zadania Z4.

Zadanie Z4

Dane: Typ struktury dokumentacji $\hat{t}_{sd_dok}$.

Szukane: Zbiór opisów poprawnych, wspomaganych procesów projektowania $\langle \hat{x}, \hat{y} \rangle$ $\hat{x} = \hat{c}_1 \dots \hat{c}_i \dots \hat{c}_n$,
 $\hat{y} = \hat{s}d_1 \dots \hat{s}d_i \dots \hat{s}d_n$ takich, że $f_{dok} \langle \hat{c}_n, T\hat{s}d_n \rangle = t\hat{s}d_{dok}$.

Problem 5

Zbiory rozwiązań dopuszczalnych, otrzymane w wyniku rozwiązania zadań Z3, Z4 mogą okazać się bardzo liczne. Może się zdarzyć, że projektant nie będzie dysponował dostatecznymi zasobami /przede wszystkim czasem/ czy umiejętnościami niezbędnymi do znalezienia rozwiązania optymalnego. Rozwiązanie zadania Z5 powinno wówczas pomóc wybrnąć z tej sytuacji.

Zadanie Z5

Dane: Zbiór rozwiązań zadania Z3 lub zadania Z4,
 zbiór K kryteriów wyboru optymalnego.

Szukane: Rozwiązanie optymalne ze względu na zbiór kryteriów K.

Między zadaniami Z1-Z4 i Z5 występują znaczne różnice. Rozwiązanie zadań Z1-Z4 prowadzi do analizy poprawności par ciągów $\langle \hat{x}, \hat{y} \rangle$ /zadanie Z1, Z2/ ewentualnie do wygenerowania zbioru wszystkich ciągów $\langle \hat{x}, \hat{y} \rangle$ o określonych cechach /zadania Z3, Z4/.

Przy ich rozwiązywaniu wystarczy skorzystać z opisu działania oprogramowania i opisów typów struktur danych. Zadanie Z5 należy do klasy zadań optymalizacji. Jego rozwiązanie wymaga opracowania charakterystyk rozwiązań dopuszczalnych, doboru kryteriów oraz sformułowania zadania optymalizacji. Rozwiązywanie tego zadania jest przedmiotem wielu badań [24], [25], [61] i nie będzie w pracy rozpatrywane. Autor ogranicza się jedynie do rozwiązania zadań Z1-Z4.

Zadania Z1, Z2, które polegają na porównywaniu par ciągów $\langle \hat{x}, \hat{y} \rangle$ z parami dopuszczalnymi, określonymi przez system zostają zaliczone do klasy zadań nadzoru.

Zadania Z3, Z4, polegające na wygenerowaniu /w celu ułatwienia korzystania z zasobów/ wszystkich rozwiązań dopuszczalnych zostają zaliczone do klasy zadań koordynacji.

Zadanie Z1 nazywane jest w dalszej części pracy pierwszym zadaniem nadzoru, Z2 - drugim zadaniem nadzoru, Z3 - pierwszym zadaniem koordynacji, Z4 - drugim zadaniem koordynacji.

Reprezentowanie typów struktur danych przez pojedyncze symbole /elementy zbioru $\hat{TSD}$ / umożliwia stosunkowo prostą formalizację rozważań oraz ma pewne znaczenie praktyczne. Przykładowo: w dokumentacjach systemów KWP poszczególne typy struktur są identyfikowane przez symbole - nazwy im przypisane; a w systemie DEDICO [62] pierwszy rekord struktur zawierał symbol typu struktury - tzw. identyfikator danych ID.

W systemach KWP w przypadku struktur danych wprowadzanych przez użytkowników, nawet jeżeli w strukturze pierwsza dana określa typ struktury, nie można a priori wnioskować, że jest ona strukturą danego typu np. $\hat{tsd}_x$. Wniosek taki można sformułować dopiero po sprawdzeniu czy spełnia ona określone w definicji typu warunki np. $\hat{war}_{\hat{tsd}_x}$.

Biorąc pod uwagę definicję typu struktury danych /podrozdział 3.2/ należy zauważyć, że dla typu struktury $\hat{tsd}_x$ zbiór warunków $\hat{war}_{\hat{tsd}_x}$ umożliwia jego identyfikację i dlatego może być uważany za $\hat{tsd}_x$ jego opis.

Uwzględniając przedstawione powyżej rozważania należy stwierdzić, że algorytmy rozwiązujące zadania Z1, Z2 muszą sprawdzać czy struktury danych spełniają warunki określone w definicji ich typów, zaś rozwiązujące zadania Z3, Z4 powinny oprócz ciągów symboli typów struktur danych wyprowadzać warunki jakie muszą spełniać struktury odpowiednich typów.

3.5. Języki procesów wspomaganych

Zdefiniowanie pojęć: procesu wspomaganego, poprawnego procesu wspomaganego, poprawnego wspomaganego procesu projektowania umożliwia i uzasadnia dokonanie klasyfikacji języków procesów wspomaganych.

Skończone ciągi z powtórzeniami elementów zbioru $\hat{C}$ będą nazywane w dalszej części pracy słowami nad alfabetem $\hat{C}$.

Def.3.1. Zbiór wszystkich słów $\hat{x} = \hat{c}_1 \dots \hat{c}_1 \dots \hat{c}_n$, dla których istnieją ciągi $\hat{y} = \hat{s}_1 \dots \hat{s}_1 \dots \hat{s}_n$ i $\hat{z} = \hat{s}_0 \hat{s}_1 \dots \hat{s}_1 \dots \hat{s}_{n-1}$ spełniające warunki Wr.3.1 i Wr.3.2 nazywany jest językiem procesów wspomaganych i oznaczany LPW/KWP/.

Def.3.2. Zbiór wszystkich słów $\hat{x} = \hat{c}_1 \dots \hat{c}_1 \dots \hat{c}_n$, dla których istnieje ciąg $\hat{y} = \hat{s}d_1 \dots \hat{s}d_1 \dots \hat{s}d_n$ taki, że $\langle \hat{x}, \hat{y} \rangle$ jest poprawnym procesem wspomaganym nazywany jest językiem poprawnych procesów wspomaganych i oznaczany LPPW/KWP/.

Def.3.3. Zbiór wszystkich słów $\hat{x} = \hat{c}_1 \dots \hat{c}_1 \dots \hat{c}_n$, dla których istnieje ciąg $\hat{y} = \hat{s}d_1 \dots \hat{s}d_1 \dots \hat{s}d_n$ taki, że $\langle \hat{x}, \hat{y} \rangle$ jest poprawnym wspomaganym procesem projektowania nazywany jest językiem poprawnych wspomaganych procesów projektowania i oznaczany LPWPP/KWP/.

Klasy języków LPW/KWP/, LPPW/KWP/, LPWPP/KWP/ oznaczane będą $\mathcal{L}_{PW/KWP/}$, $\mathcal{L}_{PPW/KWP/}$, $\mathcal{L}_{PWPP/KWP/}$.

4. ROZWIĄZANIA PIERWSZEGO ZADANIA NADZORU I PIERWSZEGO ZADANIA KOORDYNACJI

4.1. System generacyjny

Def.4.1. Systemem generacyjnym nazywać będziemy ósemkę $SG = \{V, \Sigma, D, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}\}$, gdzie:
 V, Σ, D - skończone alfabety zwane: alfabetem końcowym, alfabetem pomocniczym i alfabetem znaczników, $V \cap \Sigma = \emptyset$, $V \cap D = \emptyset$,
 $Dz_{SG} \subseteq V \times D$ - podzbiór iloczynu $V \times D$ zwany dziedziną systemu generacyjnego,
 $f_{in}, f_{out}, f_{dem}, f_{kon}$ - funkcje określone na Dz_{SG} -

$$f_{in} : Dz_{SG} \rightarrow 2^{\Sigma},$$

$$f_{out} : Dz_{SG} \rightarrow 2^{\Sigma},$$

$$f_{kon} : Dz_{SG} \rightarrow 2^{\Sigma},$$

$$f_{dem} : Dz_{SG} \rightarrow 2^{\Sigma}.$$

Zbiorem stanów systemu SG nazywany będzie skończony zbiór $S = \{s_0, s_1, \dots\}$ spełniający warunki:

$$1/ s_0 = \emptyset \quad / \text{stan początkowy},$$

$$2/ \quad \forall s_1 \quad s_1 \neq s_0 \quad / \quad /s_1 \in S \Rightarrow /s_1 \in 2^{\Sigma} \wedge s_1 \neq \emptyset //.$$

Śladem nazywany będzie dowolny ciąg zbudowany z elementów zbioru D.

Niech $\bar{w}_n = v_1 \dots v_i \dots v_n$, gdzie $\{v_1, \dots, v_i, \dots, v_n\} \subseteq V$

System SG wyprowadza słowo $\bar{w}_n$ /lub słowo w_n jest wyprowadzane w SG/, co zapisywane jest

$$\xRightarrow{SG} w_n$$

jeżeli istnieją: ślad $u_n = d_1 \dots d_i \dots d_n$ oraz ciąg stanów

$r_n = s_0 s_1 \dots s_i \dots s_{n-1}$ takie, że :

Wr.4.1. $s_0 = \emptyset$,

Wr.4.2. $\forall \langle v_i, d_i \rangle \quad 0 < i \leq n \quad /f_{in} \langle v_i, d_i \rangle \subseteq s_{i-1} \wedge$

$\wedge f_{kon} \langle v_i, d_i \rangle \cap s_{i-1} = \emptyset \wedge s_i = s_{i-1} - f_{dem} \langle v_i, d_i \rangle \cup f_{out} \langle v_i, d_i \rangle /$

Def.4.2. Zbiór wszystkich słów wyprowadzanych w SG nazywany jest językiem typu 1 definiowanym lub generowanym przez SG i oznaczany jest $L_1/SG/$.

Zgodnie z tą definicją :

$$L_1/SG/ = \{y: y \in V^* \wedge \xRightarrow{SG} y\}.$$

W dalszej części pracy klasa języków $L_1/SG/$ oznaczana będzie $\mathcal{L}_1/SG/$.

Jeżeli $w_j \in L_1/SG/$, $w_j = v_1 \dots v_i \dots v_j$ oraz $u_j = d_1 \dots d_i \dots d_j$

i $r_j = s_0 s_1 \dots s_i \dots s_{j-1}$ spełniają warunki Wr.4.1, Wr.4.2 to ciąg u_j nazywany jest śladem słowa w_j .

4.2. Regularność języka $L_1/SG/$

Tw.4.1. Dla każdego systemu generacyjnego SG język $L_1/SG/$ jest językiem regularnym.

Dowód W celu udowodnienia twierdzenia przedstawiony zostanie algorytm, który dla dowolnego systemu SG podaje sposób konstrukcji RS automatu A^1 takiego, że $L/A^1/ = L_1/SG/$.

Algorytm Al.4.1

Dane: $SG = \{V, \Sigma, D, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}\}$.

Wyniki: RS automat $A^1 = \{V, S, F, I, s_0\}$ taki, że $L/A^1 = L_1/SG/$.

Oznaczenia:

- i - numer kolejno generowanego stanu aktywnego automatu
- t - numer kolejno generowanej instrukcji części aktywnej automatu
- $\delta/I_t/$ - etykieta instrukcji I_t /element zbioru $D/$.

Początek

1. $i = 0, t = 0, s_0 = \emptyset, S = \{s_0\}, F = \emptyset, I = \emptyset$.
2. Dla każdej pary $\langle v, d \rangle$ w Dz_{SG} takiej, że $f_{in} \langle v, d \rangle = \emptyset$;
 a/ jeżeli $f_{out} \langle v, d \rangle \notin S - \{s_0\}$, to kładzie się w A^1 $i=i+1$,
 $s_i = f_{out} \langle v, d \rangle, t = t+1, I_t = s_0 v \rightarrow s_i, \delta/I_t/ = d$,
 b/ jeżeli $f_{out} \langle v, d \rangle = \emptyset$ i $s_0 v \rightarrow s_0 \notin I$, to kładzie się
 w A^1 $t = t+1, I_t = s_0 v \rightarrow s_0, \delta/I_t/ = d$,
 c/ jeżeli istnieje w A^1 stan s_x taki, że $f_{out} \langle v, d \rangle = s_x$
 i jeżeli $s_0 v \rightarrow s_x \notin I$, to kładzie się $t = t+1$,
 $I_t = s_0 v \rightarrow s_x, \delta/I_t/ = d$.
3. Jeżeli w kroku 2.a nie wygenerowano żadnego stanu, to przejście do kroku 7.
4. Dla każdej pary $\langle v, d \rangle$ w Dz_{SG} i dla każdego stanu s_u - uzyskanego w ostatnio wykonanym kroku generującym stany /krok 2.a lub 4.a/, spełniającego warunki $f_{in} \langle v, d \rangle \subseteq s_u$ i $f_{kon} \langle v, d \rangle \cap s_u = \emptyset$ - kładzie się $s_x = s_u - f_{dem} \langle v, d \rangle \cup f_{out} \langle v, d \rangle$;
 a/ jeżeli $s_x \notin S$ to kładzie się w A^1 $i=i+1, s_i = s_x, t=t+1$,
 $I_t = s_u v \rightarrow s_x, \delta/I_t/ = d$,
 b/ jeżeli $s_x \in S$ i $s_u v \rightarrow s_x \notin I$, to kładzie się
 w A^1 $t=t+1, I_t = s_u v \rightarrow s_x, \delta/I_t/ = d$.
5. Jeżeli w kroku 4.a wygenerowano przynajmniej jeden stan, to przejście do kroku 4.

6. Jeżeli istnieją symbol $v^u \in V$ oraz stan $s^z \in S$ takie, że $s^z v^u \rightarrow s_0 \in I$, to $F = S$, w przeciwnym razie $F = S - \{s_0\}$.
7. Dla każdego stanu s^z i symbolu $v^u \in V$ takich, że $s^z v^u$ nie jest poprzednikiem żadnej instrukcji A^1 kładzie się $s^z v^u \rightarrow s$.
8. Dla każdego $v^u \in V$ kładzie się w A^1 $sv^u \rightarrow s$.

Koniec

Uwagi

1. Instrukcje otrzymane w krokach 7 i 8 konieczne są jedynie po to, aby zapewnić zupełność wymagana w definicji RS automatu.

2. Otrzymany automat można uważać za zwykły RS automat /zbiór stanów jest wówczas zbiorem symboli $\{s_0, s_1, \dots, s_t, s\}$ / lub uogólniony RS automat /stan aktywny jest wówczas podzbiorem zbioru Σ /.

3. $\delta/I_j/$ jest symbolem ze zbioru D przypisywanym instrukcji I_j /etykietą instrukcji/. Etykietowanie instrukcji elementami zbioru D odgrywa rolę przy dowodzeniu twierdzenia 4.1 oraz przy modelowaniu działania systemu generacyjnego, nie ma natomiast żadnego wpływu na język akceptowany przez automat. Jest to powodem niestosowania opisu automatu A^1 przy pomocy systemu $\{V, S, F, I, s_0, D, \delta\}$, gdzie $\{V, S, F, I, s_0\}$ jest zwykłym RS automatem, a δ funkcją częściową przyporządkowującą instrukcjom elementy zbioru D / $\delta: I \rightarrow D$ /.

4. Powyższe uwagi odnoszą się również do automatów A^2 i A^3 , omówionych w dalszej części pracy, otrzymanych w wyniku działania algorytmów Al.6.1 i Al.8.1, z tą różnicą, że stan który może być stanem dozwolonym automatu A^2 jest parą $\langle s_{11}, s_{12} \rangle$, a A^3 trójką $\langle s_{11}, s_{12}, s_{13} \rangle$, gdzie s_{11} jest stanem aktywnym automatu A^1 , a s_{12} i s_{13} są zdefiniowane przy opisie odpowiednich algorytmów.

L.4.1. Dla dowolnego systemu generacyjnego $SG = \{V, \Sigma, D, D_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}\}$, jeżeli A^1 jest automatem otrzymanym w wyniku działania algorytmu Al.4.1, to zachodzi równość $L/A^1/ = L_1 /SG/$.

Dowód W pierwszej części dowodu zostanie udowodnione zachowanie implikacji -

$$/ \forall w_j' \in V^* / \quad / w_j \in L_1 / SG / \Rightarrow w_j' \in L/A^1 // .$$

Niech $w_j \in L_1 / SG /$, $w_j = v_1 \dots v_i \dots v_j$ i niech

$u_j' = d_1' \dots d_i' \dots d_j'$ będzie śladem słowa w_j . Istnieje wówczas ciąg stanów $r_j' = s_0' s_1' \dots s_i' \dots s_{j-1}'$ spełniający warunki Wr.4.1 i Wr.4.2.

Stąd i ze sposobu konstrukcji A^1 wynika, że dla A^1 istnieje ciąg stanów $s_0 s_1 \dots s_i \dots s_{j-1} s_j$ oraz ciąg instrukcji $I^1 \dots I^i \dots I^j$ takich, że $s_0 = s_0'$, $s_1 = s_1'$, ..., $s_i = s_i'$, ..., $s_{j-1} = s_{j-1}'$, $s_j = s_j'$, gdzie $s_j = s_{j-1} \xrightarrow{f_{\text{dem}}} \langle v_j', d_j' \rangle \cup f_{\text{out}} \langle v_j', d_j' \rangle$;
 $I^1 = s_0 v_1 \rightarrow s_1, \dots, I^i = s_{i-1} v_i \rightarrow s_i, \dots, I^j = s_{j-1} v_j \rightarrow s_j$;

$$\delta / I^1 / = d_1, \dots, \delta / I^i / = d_i, \dots, \delta / I^j / = d_j$$

czyli, że słowo w_j akceptowane jest przez RS automat A^1 .

Obeonie udowodnione zostanie zachodzenie implikacji

$$/ \forall w_j \in V^* / \quad / w_j \in L/A^1 / \Rightarrow w_j \in L_1 / SG // .$$

Jeżeli $w_j = v_1 \dots v_i \dots v_j$, $w_j \in L/A^1 /$, istnieją

ciąg stanów $s_0 s_1 \dots s_i \dots s_{j-1} s_j$ i ciąg instrukcji $I_1 \dots I_i \dots I_j$ takich, że $I_1 = s_0 v_1 \rightarrow s_1, \dots, I_i = s_{i-1} v_i \rightarrow s_i, \dots, I_j = s_{j-1} v_j \rightarrow s_j$;

$$\delta / I_1 / = d_1, \dots, \delta / I_i / = d_i, \dots, \delta / I_j / = d_j .$$

Ze sposobu konstrukcji automatu A^1 wynika, że ciągi

$u_j = d_1 \dots d_i \dots d_j$ i $r_j = s_0 s_1 \dots s_i \dots s_{j-1}$ spełniają warunki Wr.4.1, Wr.4.2 czyli, że słowo w_j generowane jest przez system SG. ■

Dowód lematu 4.1 kończy dowód twierdzenia 4.1. ■

4.3. Regularność języka LPW/KWP/

Miedzy modelem systemu KWP i językiem LPW/KWP/, a systemem SG i językiem $L_1 / SG /$ zachodzą analogie, które pozwalają pokazać ciekawe własności systemu KWP i języka LPW/KWP/. Własności te zostaną sformułowane w postaci wniosków.

Wn.4.1 Dla systemu KWP istnieje system SG taki, że
 $LPW/KWP/ = L_1 / SG /$.

Uzasadnienie Niech $\hat{O}D\hat{O}_{KWP} = \{\hat{P}, \hat{C}, \hat{TSD}, \hat{Dz}_{KWP}, \hat{f}_{dek}, \hat{f}_{in}, \hat{f}_{out}, \hat{f}_{kon}, \hat{f}_{dem}, \hat{f}_{dok}\}$, gdzie $\hat{TSD} = \hat{TSD}_M \cup \hat{TSD}_U \cup \hat{TSD}_D$ będzie opisem działania oprogramowania dowolnego systemu KWP. Rozważmy system $SG = \{V, \Sigma, D, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}\}$, dla którego

$$V = \hat{C}, \quad \Sigma = \hat{TSD}_M, \quad D = \hat{TSD}_U, \quad Dz_{SG} = \hat{Dz}_{KWP},$$

$$f_{in} = \hat{f}_{in}, \quad f_{out} = \hat{f}_{out}, \quad f_{kon} = \hat{f}_{kon}, \quad f_{dem} = \hat{f}_{dem}.$$

Jak łatwo sprawdzić $L_1/SG/ = LPW/KWP/$. ■

Wn.4.2. Dla każdego systemu SG istnieje system KWP taki, że $L_1/SG/ = LPW/KWP/$.

Uzasadnienie Niech $SG = \{V, \Sigma, D, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}\}$. Rozważmy system KWP, dla którego opis działania oprogramowania $\hat{O}D\hat{O}_{KWP} = \{\hat{P}, \hat{C}, \hat{D}, \hat{Dz}_{KWP}, \hat{f}_{dek}, \hat{f}_{in}, \hat{f}_{out}, \hat{f}_{kon}, \hat{f}_{dem}, \hat{f}_{dok}\}$ gdzie: $\hat{D} = \hat{TSD}_M \cup \hat{TSD}_U \cup \hat{TSD}_D$ i $\hat{C} = V$, $\hat{TSD}_M = \Sigma$, $\hat{TSD}_U = D$, $\hat{Dz}_{KWP} = Dz_{SG}$, $\hat{f}_{in} = f_{in}$, $\hat{f}_{out} = f_{out}$, $\hat{f}_{kon} = f_{kon}$, $\hat{f}_{dem} = f_{dem}$. Oczywiście $LPW/KWP/ = L_1/SG/$. ■

Z wniosków 4.1 i 4.2 wynika bezpośrednio -

Wn.4.3. Klasy języków $\mathcal{L}_{PW/KWP/}$ i $\mathcal{L}_{L_1/SG/}$ są równoważne.

Z twierdzenia 4.1 i wniosku 4.3 wynika -

Wn.4.4. Dla każdego systemu KWP język $LPW/KWP/$ jest językiem regularnym.

4.4. Algorytm rozwiązania zadania Z1

Automat A^1 będący wynikiem algorytmu A1.4.1 /oraz opisane w dalszej części pracy automaty A^2 , $A^3/$ posiada następującą własność:

Wł.4.1. Dla każdego automatu $A^1 /A^2, A^3/ = \{V, S, F, I, s_0\}$ otrzymanego w wyniku działania algorytmu A1.4.1 /6.1, 8.1/, jeżeli instrukcja $I_x \in I$, $I_x = s_x^1 v_x \rightarrow s_x^2$, $\delta/I_x/ = d_x$, to nie istnieje

instrukcja $I_y \in I - \{I_x\}$, $I_y = s_y^1 v_y \rightarrow s_y^2$, $\delta/I_y/ = d_y$
 taka, że $s_y^1 = s_x^1$, $v_y = v_x$, $s_y^2 \neq s_x^2$, $d_x = d_y$.

Automat A^1 wykorzystany zostanie do rozwiązania zadań Z1 i Z3.

Algorytm A1.4.2

Rozwiązanie zadania Z1

Dane: Automat $A^1 = \{\hat{V}, S, F, I, s_0\}$ taki, że $L/A^1/ = LPW/KWP/$,
 para $\langle \hat{w}_n, \hat{u}_n \rangle$ $\hat{w}_n = \hat{v}_1 \dots \hat{v}_2 \dots \hat{v}_n$, $\hat{u}_n = \hat{s}d_1 \dots \hat{s}d_1 \dots \hat{s}d_n$.

zbiory warunków, które muszą spełniać typy struktur należących do $T\hat{S}D_D$.

Wyniki: Stwierdzenie - czy słowo $\hat{w}_n$ akceptowane jest przez A^1
 oraz czy każda ze struktur $\hat{s}d_i$ $1 \leq i \leq n$ spełnia warunki $\text{war}_{T\hat{S}d_i}$.

Oznaczenia:

- t - liczba instrukcji opisujących część aktywną automatu
 /Zakłada się, że zbiór instrukcji indeksowany jest liczbami naturalnymi./
- $I_j = s_j^1 \hat{v}_j \rightarrow s_j^2$ - j-ta instrukcja części aktywnej automatu
- spełwarunki $/T\hat{s}d_i/$ - funkcja, która przyjmuje wartość true jeżeli struktura danych $\hat{s}d_i$ spełnia warunki $\text{war}_{T\hat{S}d_i}$.

begin

$s^1 := s_0$; $p := \text{true}$; $i := 1$; writeln;

while $i \leq n \wedge p \neq \text{false}$ do

begin

$p := \text{false}$;

for $j := 1$ to t do

if $s_j^1 = s^1 \wedge \hat{v}_j = \hat{v}_i \wedge \delta/I_j/ = T\hat{s}d_i$ then

if spełwarunki $/T\hat{s}d_i/ = \text{true}$ then

begin $p := \text{true}$; $s^1 := s_j^2$; $i := i+1$ end

end;

if $i = n+1$ then

writeln /'Błędny opis procesu, działanie' $\langle \hat{v}_1, \hat{s}d_1 \rangle$ 'nie może być wykonane przez system'/

else writeln /'Para ciągów' $\langle \hat{w}_n, \hat{u}_n \rangle$ 'opisuje proces, który może być wykonany przez system'/

end.

Uwagi o czasie wykonania algorytmu A1.4.2

Przyjmując, że czas wykonania instrukcji elementarnych $/s^1 := s_0, p := \text{true itp.}/$ jest pomijalny z czasem obliczania wartości funkcji spełwarunki maksymalny czas wykonania algorytmu

$T_{\max}^{A1.4.2}$ można oszacować następująco:

$$T_{\max}^{A1.4.2} = n \cdot t_{\max}^{s.w.},$$

gdzie - $t_{\max}^{s.w.}$ - maksymalny /dla danego systemu KWP/ czas wykonania funkcji spełwarunki /czas sprawdzenia czy struktura $\hat{s}d_x$ spełnia warunki war $_{T\hat{s}d_x}$ /.

Postać wzoru pozwala stwierdzić, że dla danego systemu KWP maksymalny czas wykonania algorytmu jest liniową funkcją ilości elementów ciągu $\hat{w}_n$.

Przykład 4.1

Rozważmy system KWP o opisie działania oprogramowania $\hat{ODO}_{KWP}$ przedstawionym w tabeli 4.1.

Na rysunku 4.1 przedstawiono diagram stanów automatu A^1 akceptującego język LPW/KWP/.

Niech $\hat{d}_1, \hat{d}_2, \dots, \hat{d}_7$ będą typami struktur $\hat{s}d_1, \hat{s}d_2, \dots, \hat{s}d_7$.

Z rysunku 4.1 wynika, że procesy $\langle \hat{v}_6, \hat{s}d_6 \rangle \langle \hat{v}_1, \hat{s}d_1 \rangle$

$\langle \hat{v}_2, \hat{s}d_2 \rangle \langle \hat{v}_3, \hat{s}d_3 \rangle$, $\langle \hat{v}_7, \hat{s}d_7 \rangle \langle \hat{v}_6, \hat{s}d_6 \rangle \langle \hat{v}_1, \hat{s}d_1 \rangle$ są poprawnymi procesami wspomaganyimi, natomiast procesy

$\langle \hat{v}_6, \hat{s}d_7 \rangle \langle \hat{v}_1, \hat{s}d_1 \rangle$ czy $\langle \hat{v}_7, \hat{s}d_7 \rangle \langle \hat{v}_2, \hat{s}d_2 \rangle \langle \hat{v}_5, \hat{s}d_5 \rangle$ nimi nie są.

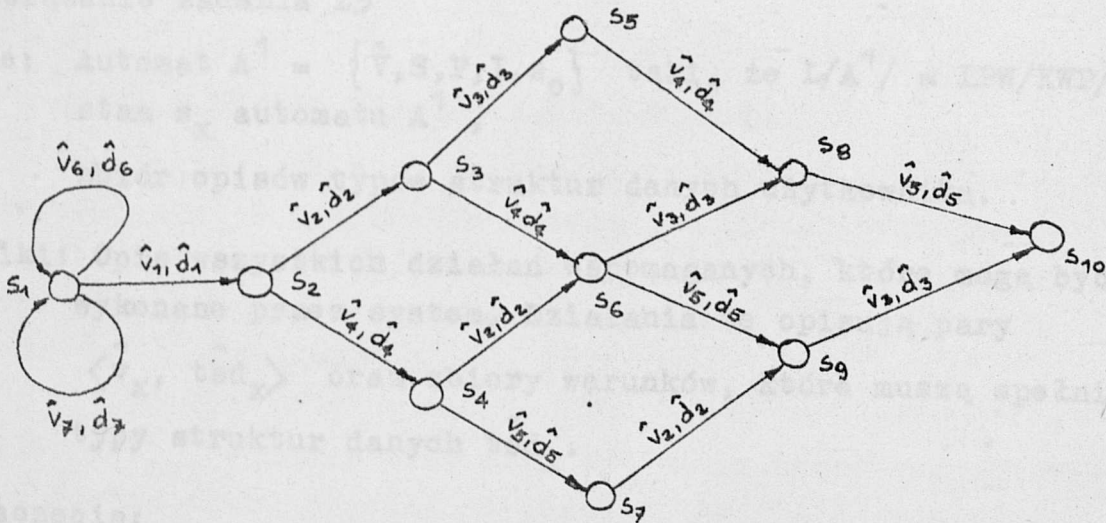
Tab.4.1.

Opis działania oprogramowania systemu KWP z przykładu 4.1

$\hat{P}$	$\hat{C}$	$\langle \hat{v}_x, \hat{d}_x \rangle$	$\hat{t}_{in}$ $\langle \hat{v}_x, \hat{d}_x \rangle$	$\hat{t}_{out}$ $\langle \hat{v}_x, \hat{d}_x \rangle$	$\hat{t}_{kon}$ $\langle \hat{v}_x, \hat{d}_x \rangle$	$\hat{t}_{dem}$ $\langle \hat{v}_x, \hat{d}_x \rangle$	$\hat{t}_{dou}$ $\langle \hat{v}_x, \hat{d}_x \rangle$
$\hat{P}_1$	$\hat{V}_1$	$\langle \hat{V}_1, \hat{d}_1 \rangle$	$\emptyset$	$\hat{A}$	$\hat{A}, \hat{B}, \hat{G}, \hat{D}, \hat{E}$	$\emptyset$	$\hat{t}\hat{A}$
$\hat{P}_2$	$\hat{V}_2$	$\langle \hat{V}_2, \hat{d}_2 \rangle$	$\hat{A}$	$\hat{B}$	$\hat{B}, \hat{G}$	$\emptyset$	$\hat{t}\hat{B}$
$\hat{P}_3$	$\hat{V}_3$	$\langle \hat{V}_3, \hat{d}_3 \rangle$	$\hat{B}$	$\hat{G}$	$\hat{G}$	$\hat{B}$	$\hat{d}_3^d$
$\hat{P}_4$	$\hat{V}_4$	$\langle \hat{V}_4, \hat{d}_4 \rangle$	$\hat{A}$	$\hat{D}$	$\hat{D}, \hat{E}$	$\emptyset$	$\hat{t}\hat{D}$
$\hat{P}_5$	$\hat{V}_5$	$\langle \hat{V}_5, \hat{d}_5 \rangle$	$\hat{D}$	$\hat{E}$	$\hat{E}$	$\hat{D}$	$\hat{d}_5^d$
$\hat{P}_6$	$\hat{V}_6$	$\langle \hat{V}_6, \hat{d}_6 \rangle$	$\emptyset$	$\emptyset$	$\hat{A}, \hat{B}, \hat{G}, \hat{D}, \hat{E}$	$\emptyset$	$\hat{d}_6^d$
$\hat{P}_7$	$\hat{V}_7$	$\langle \hat{V}_7, \hat{d}_7 \rangle$	$\emptyset$	$\emptyset$	$\hat{A}, \hat{B}, \hat{G}, \hat{D}, \hat{E}$	$\emptyset$	$\hat{d}_7^d$

$$\hat{P} = \{ \hat{P}_1, \hat{P}_2, \hat{P}_3, \hat{P}_4, \hat{P}_5, \hat{P}_6, \hat{P}_7 \}, \quad \hat{C} = \{ \hat{V}_1, \hat{V}_2, \hat{V}_3, \hat{V}_4, \hat{V}_5, \hat{V}_6, \hat{V}_7 \}, \quad \hat{TSD} = \hat{TSD}_M \cup \hat{TSD}_U \cup \hat{TSD}_D,$$

$$\hat{TSD}_M = \{ \hat{A}, \hat{B}, \hat{D}, \hat{E}, \hat{G} \}, \quad \hat{TSD}_U = \{ \hat{d}_1, \hat{d}_2, \hat{d}_3, \hat{d}_4, \hat{d}_5, \hat{d}_6, \hat{d}_7 \}, \quad \hat{TSD}_D = \{ \hat{d}_3^d, \hat{d}_5^d, \hat{d}_6^d, \hat{d}_7^d, \hat{t}\hat{A} \}$$

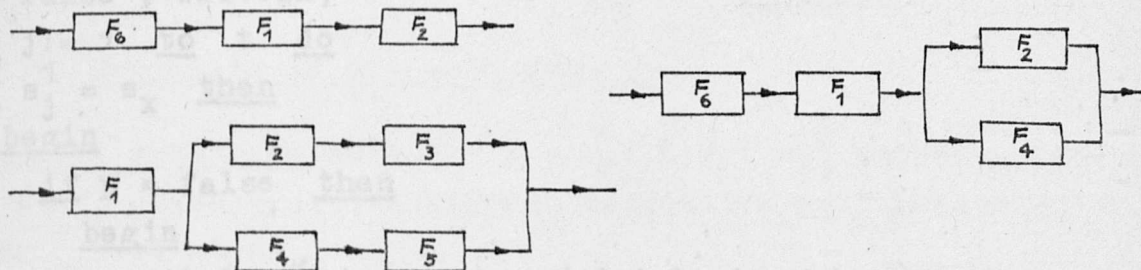


$$s_1 = \emptyset, \quad s_2 = \{ \hat{A} \}, \quad s_3 = \{ \hat{A}, \hat{B} \}, \quad s_4 = \{ \hat{A}, \hat{D} \}, \quad s_5 = \{ \hat{A}, \hat{G} \}, \quad s_6 = \{ \hat{A}, \hat{B}, \hat{D} \}$$

$$s_7 = \{ \hat{A}, \hat{E} \}, \quad s_8 = \{ \hat{A}, \hat{G}, \hat{D} \}, \quad s_9 = \{ \hat{A}, \hat{B}, \hat{E} \}, \quad s_{10} = \{ \hat{A}, \hat{E}, \hat{G} \}$$

Rys.4.1. Diagram stanów automatu A^1 akceptującego język LPW systemu KWP z przykładu 4.1

Na rysunku 4.1. przedstawiono przykładowe struktury procesów, które mogą być wspomagane przez rozważony system /przyjęto założenie, że działaniu F_1 odpowiada działanie wspomagane $\langle \hat{v}_1, \hat{s}d_1 \rangle$ $i = 1, \dots, 7$ /.



Rys.4.2. Przykładowe struktury procesów, które mogą być wspomagane przez system KWP z przykładu 4.1

4.5. Algorytm rozwiązania zadania Z3

Algorytm Al.4.3

Rozwiązanie zadania Z3

Dane: Automat $A^1 = \{\hat{V}, S, F, I, s_0\}$ taki, że $L/A^1/ = LPW/KWP/$,
stan s_x automatu A^1 ,

zbiór opisów typów struktur danych użytkownika.

Wyniki: Opis wszystkich działań wspomaganych, które mogą być wykonane przez system. Działania te opisują pary

$\langle \hat{v}_x, \hat{tsd}_x \rangle$ oraz zbiory warunków, które muszą spełniać typy struktur danych $\hat{tsd}_x$.

Oznaczenia:

- t, I_j - symbole mają takie same znaczenia jak w przypadku algorytmu Al.4.2
- wyprsymb $/\hat{v}_j, \hat{tsd}_j/$ - procedura, która wyprowadza na urządzenie wyjścia rozkaz $\hat{v}_j$ i symbol typu struktury danych $\hat{tsd}_j$
- opiswarunków $/\hat{tsd}_j/$ - procedura, która wyprowadza na urządzenie

wyjścia opis warunków, które musi spełniać typ struktury danych $\hat{t}sd_j$, opis warunków wyprowadzany jest tylko raz dla danego typu struktury.

begin

p:= false ; writeln;

for j:= 1 to t do

if $s_j^1 = s_x$ then

begin

if p = false then

begin

writeln /możesz wykonać działania opisane przez
następujące pary: '/ ;

p:= true

end;

$\hat{t}sd_j := \delta / I_j$;

wypr symb / $\hat{v}_j, \hat{t}sd_j$ / ;

opiswarunków / $\hat{t}sd_j$ /

end;

if p = false then

writeln /'nie możesz wykonać żadnego działania'/

end.

Uwagi o czasie wykonania algorytmu Al.4.3

Przyjmując, że czas wykonania instrukcji elementarnych jest pomijalny w porównaniu z czasem obliczeń wykonywanych przez procedurę opiswarunków, maksymalny czas $T_{\max}^{Al.4.3}$ wykonania algorytmu Al.4.3 można oszacować następująco:

$$T_{\max}^{Al.4.3} = t \cdot t_{\max}^{O.W.},$$

gdzie: - $t_{\max}^{O.W.}$ - maksymalny czas wykonania procedury opiswarunków.

Postać wzoru pozwala stwierdzić, że dla danego systemu KWP czas wykonania algorytmu jest stały.

Przykład 4.2

Rozważmy system KWP z przykładu 4.1. Przyjmijmy, że system znajduje się w stanie $\hat{s}_6 = \{\hat{A}, \hat{B}, \hat{D}\}$. Możliwe do wykonania działania wspomagane mogą być reprezentowane przez pary $\langle \hat{v}_3, \hat{d}_3 \rangle$, $\langle \hat{v}_5, \hat{d}_5 \rangle$.

Algorytm A1.4.3 rozwiązujący zadanie Z3 dostarczy w tym przypadku 2 pary $\langle \hat{v}_3, \hat{d}_3 \rangle$ $\langle \hat{v}_5, \hat{d}_5 \rangle$ oraz opisy typów struktur danych $\hat{d}_3, \hat{d}_5$ /warunki $\text{war}_{\hat{d}_3}$ i $\text{war}_{\hat{d}_5}$ /.

5. WŁASNOŚCI KLAS JĘZYKÓW $\mathcal{L}_{1/SG/}$ i $\mathcal{L}_{PW/KWP/}$

5.1. Definicja RS automatu typu 1

Stanem zakazanym automatu $A = \{V, S, F, I, s_0\}$ nazywany jest stan $s \in F$, po osiągnięciu którego niemożliwe jest osiągnięcie żadnego innego stanu.

W rozdziałach 5 i 7 rozważane będą RS automaty posiadające dokładnie jeden stan zakazany /każdy RS automat można doprowadzić do takiej postaci/. Stan zakazany oznaczany jest s .

Def.5.1. RS automat $A = \{V, S, F, I, s_0\}$ spełniający warunek -

Wr.5.1.a. Jeżeli w zbiorze instrukcji automatu A istnieje instrukcja, której następnikiem jest stan początkowy s_0 , to $F = S - \{s\}$, lub warunek -

Wr.5.1.b. Jeżeli w zbiorze instrukcji automatu A nie istnieje instrukcja, której następnikiem jest stan s_0 , to $F = S - \{s, s_0\}$,

nazywany będzie automatem Rabina - Scotta typu 1 lub krótko RS automatem typu 1.

RS automat typu 1 jest, więc RS automatem posiadającym jeden stan zakazany s , dla którego każdy ze stanów $s_i \in S - \{s\}$ /ewentualnie $S - \{s, s_0\}$ / jest stanem końcowym.

Języki definiowane przez RS automaty typu 1 nazywane będą językami regularnymi typu 1. Klasa języków regularnych typu 1 oznaczana będzie $\mathcal{L}_{R1}$.

5.2. Równoważność klas $\mathcal{L}_{R1}$ i $\mathcal{L}_{1/SG/}$

Tw.5.1. Klasy $\mathcal{L}_{R1}$ i $\mathcal{L}_{1/SG/}$ są równoważne.

Dowód Należy udowodnić zachodzenie następujących zależności:

1/ $\forall L_1 \in \mathcal{L}_{1/SG/} \quad / \exists L_2 \in \mathcal{L}_{R1} \quad / L_1 = L_2 /$,

2/ $\forall L_1 \in \mathcal{L}_{R1} \quad / \exists L_2 \in \mathcal{L}_{1/SG/} \quad / L_1 = L_2 /$.

Jako pierwsza udowodniona zostanie zależność 1, którą formuluje się jako lemat 5.1.

L.5.1. Dla każdego języka typu 1 L_1 generowanego przez system SG istnieje język regularny typu 1 L_2 taki, że $L_1 = L_2$.

Dowód wynika bezpośrednio z twierdzenia 4.1 i algorytmu 4.1 syntezy RS automatu A^1 akceptującego język typu 1 generowany przez system SG. Prosta analiza automatu A^1 prowadzi do stwierdzenia, że jest on RS automatem typu 1. ■

Udowodniony zostanie teraz lemat 5.2 będący wersją zależności 2.

L.5.2. Dla każdego RS automatu typu 1 A^1 istnieje język typu 1 generowany przez system SG L_2 taki, że $L_2 = L/A^1/$.

Dowód Niech $A^1 = \{V^1, S, F, I, s_0\}$ będzie RS automatem typu 1. Bez straty ogólności rozważań można założyć, że A^1 jest deterministycznym RS automatem typu 1. Skonstruowany zostanie system SG taki, że $L_1 /SG/ = L /A^1/$. Niech $SG = \{V, \Sigma, D, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}\}$, gdzie: $V = V^1$, $\Sigma = S - \{s\}$, $D = \Sigma$, Dz_{SG} jest zbiorem par $\langle a_1, s_1 \rangle$, dla których instrukcja $s_j a_1 \rightarrow s_1$ należy do części aktywnej automatu A^1 , a funkcje f_{in} , f_{out} , f_{kon} , f_{dem} określone są następująco:

1. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_i, s_j \in \{s_0, s\}$ w A^1 kładzie się w SG $f_{in} \langle a_1, s_i \rangle = \{s_j\}$, $f_{out} \langle a_1, s_i \rangle = \{s_i\}$, $f_{kon} \langle a_1, s_i \rangle = \emptyset$, $f_{dem} \langle a_1, s_i \rangle = \{s_j\}$.
2. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j = s_0$, $s_i \in \{s_0, s\}$ w A^1 kładzie się w SG $f_{in} \langle a_1, s_i \rangle = \emptyset$, $f_{out} \langle a_1, s_i \rangle = \{s_i\}$, $f_{kon} \langle a_1, s_i \rangle = S - \{s_0\}$, $f_{dem} \langle a_1, s_i \rangle = \{s_j\}$.
3. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j, s_i = s_0$ w A^1 kładzie się w SG $f_{in} \langle a_1, s_i \rangle = \emptyset$, $f_{out} \langle a_1, s_i \rangle = \emptyset$, $f_{kon} \langle a_1, s_i \rangle = S - \{s_0\}$, $f_{dem} \langle a_1, s_i \rangle = \emptyset$.
4. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j \in \{s_0, s\}$, $s_i = s_0$ w A^1 kładzie się w SG $f_{in} \langle a_1, s_i \rangle = \{s_j\}$, $f_{out} \langle a_1, s_i \rangle = \emptyset$, $f_{kon} \langle a_1, s_i \rangle = \emptyset$, $f_{dem} \langle a_1, s_i \rangle = \{s_j\}$.

Jak łatwo sprawdzić $L_1 /SG/ = L /A^1/$. ■

Równoważność klas $\mathcal{L}_{R1}$ i $\mathcal{L}_{1/SG/}$ zostanie wykorzystana do udowodnienia algebraicznych własności klasy języków $\mathcal{L}_{PW/KWP/}$.

5.3. Algebraiczne własności klasy $\mathcal{L}_{R1}$

Tw.5.2. Klasa $\mathcal{L}_{R1}$ jest zamknięta ze względu na operację podstawiania.

Dowód Niech L będzie językiem regularnym typu 1 nad alfabetem $V = \{a_1, a_2, \dots, a_n\}$. Każdemu symbolowi a_i tego alfabetu przyporządkowany zostaje nowy alfabet W^i . Rozważmy dowolne podstawienie $\varphi: V^* \rightarrow 2^{W^*}$, gdzie $W = \bigcup_{i=1}^n W^i$ oraz dla każdego $i = 1, \dots, n$ $\varphi/a_i/$ jest językiem regularnym typu 1 nad W^i . Pokażemy, że $\varphi/L/$ jest językiem regularnym typu 1.

Niech $A = \{V, S, F, I, s_0\}$ będzie RS automatem typu 1 definiującym język L i niech $A^i = \{V^i, S^i, F^i, I^i, s_0^i\}$ będą, dla $i = 1, \dots, n$ RS automatami typu 1 definiującymi języki $\varphi/a_1/, \dots, \varphi/a_n/$.

Automat definiujący język $\varphi/L/$ określamy następująco

$$A^0 = \{W, S^0, F^0, I^0, s_0^0\}, \text{ gdzie:}$$

$S^0 = \{s_{jik} : j=1, \dots, m, i=1, \dots, n, k=1, \dots, m_i\} \cup \{s_0^0, s\}$, przy czym m jest liczbą elementów zbioru F , m_i liczbą elementów zbioru F^i dla $i=1, \dots, n$. Każdy stan s_{jik} reprezentuje parę uporządkowaną stanów $\langle s_j, s_k^i \rangle$ gdzie $s_j \in F$, $s_k^i \in F^i$.

$F^0 = \{s_{jik}, s_j \in F \wedge s_k^i \in F^i\}$, a więc stany końcowe automatu A^0 reprezentują pary stanów końcowych automatu A i automatów A^i . Zakładamy ponadto, że jeżeli $\xi \in \varphi/L/$, to wtedy i tylko wtedy do F^0 dołączamy stan s_0^0 .

Listę instrukcji I^0 konstruujemy w oparciu o zbiór przedstawionych poniżej 6 reguł. Reguły 1-3, 6 dotyczą części aktywnej automatu A^0 , a 4, 5 części pasywnej.

1. Dla każdego $i=1, \dots, n$ i każdej pary instrukcji $s_0 a_i \rightarrow s_j$ w A oraz $s_0^i b_p \rightarrow s_k^i$ w A^i kładziemy w A^0 $s_0^0 b_p \rightarrow s_{jik}$.
2. Dla każdego $i=1, \dots, n$ i każdej instrukcji $s_k^i b_p \rightarrow s_r^i$ takiej, że $s_k^i, s_j^i \in F^i$ w A^i kładziemy w A^0 $s_{jik} b_p \rightarrow s_{jir}$, gdzie $j = 1, \dots, m$.
3. Dla każdego $i = 1, \dots, n$ każdego stanu $s_j^i \in F^i$ oraz każdego $l=1, \dots, n$ i każdej pary instrukcji $s_0^l b_q \rightarrow s_w^l$ w A^l oraz $s_j a_l \rightarrow s_r$ w A kładziemy w A^0 $s_{jir} b_q \rightarrow s_{rlw}$.

4. Dla każdego stanu $s_{jik} \in S^0$ i symbolu $b_p \in W$, dla których $s_{jik} b_p$ nie jest poprzednikiem żadnej instrukcji kładziemy w A^0 $s_{jik} b_p \rightarrow s$.
5. Dla każdego symbolu $b_p \in W$ kładziemy w A^0 $s b_p \rightarrow s$.
6. Jeżeli $\varepsilon \in \varphi/L$, to dołączamy do zbioru I^0 instrukcję $s_0^0 \varepsilon \rightarrow s_0^0$.

Aby udowodnić tezę twierdzenia należy udowodnić następujące lematy:

L.5.3. A^0 jest RS automatem typu 1.

L.5.4. $\varphi/L/ = L/A^0/$.

Dowód lematu 5.3.

Zauważmy, że jedyną instrukcją w A^0 spełniającą warunek 5.1.a może być instrukcja $s_0^0 \varepsilon \rightarrow s_0^0$. Jeżeli A^0 posiada taką instrukcję to $F^0 = S^0 - \{s\}$, w przeciwnym razie $F^0 = S^0 - \{s_0^0, s\}$. A^0 jest więc RS automatem typu 1.

Szkic dowodu lematu 5.4.

Dowód wynika z oczywistej inkluzji $\mathcal{L}_{R1} \subset \mathcal{L}_R / \mathcal{L}_R$ oznacza klasę języków regularnych/ oraz dowodu twierdzenia 2.3.5 [6]:

Klasa $\mathcal{L}_R$ jest zamknięta ze względu na operację podstawiania, w którym pokazano, dla prawie identycznie skonstruowanego automatu A^0 , zachodzenie dla dowolnego języka regularnego L równości $\varphi/L/ = L/A^0/$.

Tw.5.3. Jeżeli klasa $\mathcal{K}$ jest zamknięta ze względu na operację podstawiania oraz istnieją takie symbole a i b , że języki $\{a, b\}$, $\{a\}^*$ należą do $\mathcal{K}$, to klasa $\mathcal{K}$ jest zamknięta ze względu na operację sumy i domknięcia języków.

Twierdzenie 5.3 jest wersją twierdzenia 1.1.1 [6].

Tw.5.4. Klasa $\mathcal{L}_{R1}$ jest zamknięta ze względu na operację sumy i domknięcia języków.

Dowód Rozważmy dwa RS automaty typu 1 A_1, A_2 :

1. $A_1 = \{V_1, S_1, F_1, I_1, s_{01}\}$, gdzie:
 - $V_1 = \{a, b\}$,
 - $S_1 = \{s_{01}, s_{11}, s_{21}\}$,
 - $F_1 = \{s_{11}, s_{21}\}$,

$$I_1 = \{ s_{01}a \rightarrow s_{11}, s_{01}b \rightarrow s_{21} \}, \\ L/A_1/ = \{a, b\};$$

$$2. \quad A_2 = \{V_2, S_2, F_2, I_2, s_{02}\}, \quad \text{gdzie:} \\ V_2 = \{a, \varepsilon\}, \\ S_2 = \{s_{02}, s_{12}\}, \\ F_2 = \{s_{02}, s_{12}\}, \\ I_2 = \{s_{02}\varepsilon \rightarrow s_{02}, s_{02}a \rightarrow s_{12}, s_{12}a \rightarrow s_{12}\}, \\ L/A_2/ = \{a\}^*.$$

Na podstawie twierdzenia 5.3 klasa $\mathcal{L}_{R1}$ jest zamknięta ze względu na operacje sumy i domknięcia. ■

Def.5.2. Przedłużeniem języka L_1 językiem L_2 nazywamy język $L_2 \stackrel{\text{def}}{=} L_1 \cup L_1L_2$.

L.5.5. Klasa $\mathcal{L}_{R1}$ jest zamknięta ze względu na operacje przedłużania.

Dowód Niech L_1 i L_2 będą dowolnymi językami należącymi do klasy $\mathcal{L}_{R1}$. Rozważmy podstawienie f określone na $\{a, ab\}$ takie, że $f/a/ = L_1$, $f/b/ = L_2$. Wówczas $f/\{a, ab\}/ = L_1 \cup L_1L_2$. Ponieważ $\{a, ab\} \in \mathcal{L}_{R1}$, więc na podstawie twierdzenia 5.2 $/L_1 \cup L_1L_2/ \in \mathcal{L}_{R1}$. ■

Przed rozważeniem zamkniętości klasy $\mathcal{L}_{R1}$ ze względu na operacje złożenia, różnicy i uzupełnianie - przedstawiony zostanie lemat formułujący ważną własność języka regularnego typu 1.

L.5.6. Dla każdego $L_1 \in \mathcal{L}_{R1}$, jeżeli $x_n \in L_1$, $x_n = a_1 \dots a_{n-1}a_n$, to słowo $x_{n-1} = a_1 \dots a_{n-1}$ również należy do L_1 .

Dowód - bezpośrednio z definicji RS automatu typu 1. ■

L.5.7. Klasa $\mathcal{L}_{R1}$ nie jest zamknięta ze względu na operację złożenia.

Dowód Niech $L_1 = \{a\}$, $L_2 = \{b\}$; oczywiście $L_1, L_2 \in \mathcal{L}_{R1}$. $L_1L_2 = \{ab\}$. Na mocy lematu 5.6 język ten nie należy do klasy $\mathcal{L}_{R1}$. ■

L.5.8. Klasa $\mathcal{L}_{R1}$ jest właściwą podklasą klasy $\mathcal{L}_R$. ■

Dowód $\{ab\}$ jest językiem regularnym, a nie jest językiem regularnym typu 1 /lemat 5.6/. ■

L.5.9. Klasa $\mathcal{L}_{R1}$ nie jest zamknięta ze względu na operację różnicy.

Dowód Rozważmy języki $L_1 = \{a, ab, abc\}$, $L_2 = \{a, ab, abc, abcd\}$ oraz język $L_3 = L_2 - L_1 = \{abcd\}$. Na mocy lematu 5.6 $L_3 \notin \mathcal{L}_{R1}$. ■

L.5.10. Klasa $\mathcal{L}_{R1}$ nie jest zamknięta ze względu na uzupełnianie.

Dowód Rozważmy dwa języki L_1, L_2 ; $L_1 = \{a, ab\}$, $L_2 = \{abc\}$; $L_1 \in \mathcal{L}_{R1}$, a $L_2 \notin \mathcal{L}_{R1}$. Zauważmy, że $L_2 = /V^* - L_1/$.

Założmy, że $/V^* - L_1/ \in \mathcal{L}_{R1}$. Wówczas na mocy lematu 5.6 powinna zachodzić implikacja $/abc \in /V^* - L_1/ \Rightarrow /ab \in /V^* - L_1/$. Jednak $ab \in L_1$, a zatem $/V^* - L_1/ \notin \mathcal{L}_{R1}$. Stąd klasa $\mathcal{L}_{R1}$ nie jest zamknięta ze względu na uzupełnianie. ■

Tw.5.5. Klasa języków $\mathcal{L}_{R1}$ jest zamknięta ze względu na operację iloczynu.

Dowód Niech A_1 i A_2 będą dowolnymi RS automatami typu 1. Bez straty na ogólności rozważań można założyć, że A_1 i A_2 są deterministycznymi RS automatami typu 1. Niech

$A_1 = \{V_1, S_1, F_1, M_1, s_0^1\}$, gdzie $S_1 = \{s_0^1, s_1^1, \dots, s_{m_1}^1, s^1\}$;

$A_2 = \{V_2, S_2, F_2, M_2, s_0^2\}$, gdzie $S_2 = \{s_0^2, s_1^2, \dots, s_{m_2}^2, s^2\}$,

M jest funkcją zdefiniowaną w pracy [6] na stronie 45 a s^1, s^2 są stanami zakazanymi automatów A_1, A_2 .

Niech $V_1 = V_2 = V$. Pokażemy, że wychodząc od automatów A_1 i A_2 można skonstruować automat akceptujący język $L/A_1/ \cap L/A_2/$ oraz, że automat ten jest RS automatem typu 1.

Język $L/A_1/ \cap L/A_2/$ jest definiowany - jak łatwo sprawdzić przez automat A_3 :

$A_3 = A_1 \times A_2 = \{V, S_1 \times S_2, F_1 \times F_2, M_1 \times M_2, \langle s_0^1, s_0^2 \rangle\}$, przy czym

$M_1 \times M_2 // s_i^1, s_k^2 / a_j / = / M_1 / s_i^1, a_j / , / M_2 / s_k^2, a_j /$, gdzie:

$i = 1, \dots, m_1 + 2$, $k = 1, \dots, m_2 + 2$, a $a_j \in V$.

A_3 jest oczywiście deterministycznym RS automatem.

Dla automatu A_3 istnieje efektywna metoda sprawdzenia, czy język $L/A_3/$ jest niepusty. Wynika ona z twierdzenia 2.3.12 [6]:

Niech A będzie RS automatem o liczbie stanów równej n . Język $L/A/$ jest niepusty wtedy, i tylko wtedy, gdy istnieje słowo o długości mniejszej od n akceptowane przez A .

W dalszej części dowodu przyjmujemy nie, że $L/A_3 \neq \emptyset$.

Niech s oznacza każdy ze stanów $\langle s_{x1}, s_{y2} \rangle$ automatu A_3 taki, że $s_{x1} \notin /F \cup \{s_0\} /$ lub $s_{y2} \notin /F \cup \{s_0\} /$, a s'' oznacza stan $\langle s_0^1, s_0^2 \rangle$ tego automatu.

Rozważmy automat

$A_3' = \{V, S_3', F_3', M_3', s_0^3\}$, dla którego:

$F_3' = F_1 \times F_2$,

$S_3' = F_3' \cup \{s_0^3, s^3\}$,

$s_0^3 = s''$ - stan początkowy,

$s^3 = s^*$ - stan zakazany,

$M_3': V \times S_3' \rightarrow S_3'$, przy czym

$M_3' // s_i^1, s_k^2, a_j / = /M_1/s_i^1, a_j /, M_2/s_k^2, a_j /$, gdzie:

$i = 1, \dots, m_1+2$, $k = 1, \dots, m_2+2$ i każdy stan $\langle s_{x1}, s_{y2} \rangle$ taki, że $s_{x1} \notin /F \cup \{s_0\} /$ lub $s_{y2} \notin /F \cup \{s_0\} /$ zastąpiony jest stanem s^3 , a stan $\langle s_0^1, s_0^2 \rangle$ stanem s_0^3 .

Oczywiście $L/A_3' = L/A_3 = L/A_1 \cap L/A_2$ - co więcej - ponieważ F_3' spełnia warunek Wr.5.1.a lub Wr.5.1.b, A_3' jest również RS automatem typu 1. ■

L.5.11. Klasa $\mathcal{L}_{R1}$ nie jest zamknięta ze względu na operację odbicia zwierciadlanego.

Dowód Rozważmy język $L = \{a, ab\}$. Jak łatwo pokazać, jest on językiem regularnym typu 1. Język $L^R = \{a, ba\}$ nie jest definiowany przez żaden RS automat typu 1, gdyż gdyby tak było, to język definiowany przez taki automat musiałby /na podstawie lematu 5.6/ zawierać słowo $b \notin L^R$. ■

Wniosek 5.1 podsumowuje wyniki osiągnięte w tym podrozdziale

Wn.5.1. Klasa języków $\mathcal{L}_{R1}$ jest zamknięta ze względu na operacje podstawiania, sumy, domknięcia, przedłużania i iloczynu, a nie jest zamknięta ze względu na uzupełnianie i operacje złożenia, różnicy i odbicia zwierciadlanego.

5.4. Algebraiczne własności klas $\mathcal{L}_{1/SG/}$ i $\mathcal{L}_{PW/KWP/}$

Tw.5.6. Klasa języków $\mathcal{L}_{1/SG/}$ jest zamknięta ze względu na operacje podstawiania, sumy, domknięcia, przedłużania i iloczynu, a nie jest zamknięta ze względu na uzupełnianie i operacje złożenia, różnicy i odbicia zwierciadlanego.

Dowód - bezpośrednio z twierdzenia 5.1 i wniosku 5.1. ■

Wn.5.2. Klasa języków $\mathcal{L}_{PW/KWP/}$ jest zamknięta ze względu na operacje podstawiania, sumy, domknięcia, przedłużania i iloczynu, a nie jest zamknięta ze względu na uzupełnianie i operacje złożenia, różnicy i odbicia zwierciadlanego.

Uzasadnienie Wniosek wynika bezpośrednio z twierdzenia 5.6 i wniosku 4.3. ■

Tw.5.7. Klasy $\mathcal{L}_R$ i $\mathcal{L}_{1/SG/}$ nie są równoważne.

Dowód wynika bezpośrednio z twierdzenia 5.1 i lematu 5.7. ■

Wr.5.3. Klasy $\mathcal{L}_{PW/KWP/}$ i $\mathcal{L}_R$ nie są równoważne.

Uzasadnienie - wniosek wynika bezpośrednio z twierdzenia 5.7 i wniosku 4.3. ■

Większość operacji rozpatrywanych w rozdziale - obok interpretacji algebraicznej - posiada znaczenie praktyczne. Przykładowo - w przypadku systemu KWP część operacji może być uważana za operacje odpowiadające operacjom wykonywanym na oprogramowaniu użytkowym.

Operacja sumy języków może być uważana jako operacja odpowiadająca sumowaniu /"scalaniu"/ bibliotek programów użytkowych całkowicie niezależnych systemów.

Operacje podstawiania można traktować jako operację odpowiadającą zastępowaniu pojedynczych programów ciągami programów.

Zamkniętość języka - ze względu na operacje podstawiania - umożliwia modularną /wielopoziomową/ definicję języka.

Operacja przedłużania odpowiada dodawaniu do biblioteki programów wspomagających kolejne etapy procesu projektowania np. etapowo sporządzających dodatkową, skomplikowaną dokumentację.

Zamkniętość klasy $\mathcal{L}$, ze względu na operację $\otimes$ oznacza, że metody opracowane dla języka $L \in \mathcal{L}$ mogą być stosowane również dla systemów ulegających zmianom, którym w kategoriach językowych odpowiada operacja $\otimes$.

6. ROZWIĄZANIE DRUGIEGO ZADANIA NADZORU

6.1. Definicja języka $L_2/SG/$

Def.6.1. Niech $SG = \{V, \Sigma, D, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}\}$ będzie systemem generacyjnym i niech $w_n \in L_1/SG/$ $w_n = v_1 \dots v_i \dots v_n$. Symbol v_i $1 \leq i < n$ jest symbolem zbędnym słowa w_n , jeżeli dla każdego śladu u_n , $u_n = d_1 \dots d_i \dots d_n$ słowa w_n spełniony jest warunek:

$$\text{Wr.6.1. } \exists d_i^{out} \in f_{out}\langle v_i, d_i \rangle // \forall \langle v_j, d_j \rangle \ i < j \leq n / \\ d_i^{out} \notin f_{in}\langle v_j, d_j \rangle /.$$

Def.6.2. Zbiór wszystkich słów $w_n \in L_1/SG/$, które nie zawierają symboli zbędnych, nazywany jest językiem typu 2 generowanym przez SG i oznaczany $L_2/SG/$.

Klasa wszystkich języków typu 2 generowanych przez systemy generacyjne oznaczana będzie $\mathcal{L}_2/SG/$.

6.2. Regularność języka $L_2/SG/$

Niech $w_j \in L_1/SG/$, $w_j = v_1 \dots v_i \dots v_j$ oraz $u_j = d_1 \dots d_i \dots d_j$ będzie śladem w_j . Przez $b\langle w_j, u_j \rangle$ oznaczany będzie podzbiór zbioru Σ wyznaczony w następujący sposób:

$$\begin{aligned} b\langle w_0, u_0 \rangle &= \emptyset, \quad b\langle w_1, u_1 \rangle = f_{out}\langle v_1, d_1 \rangle, \\ b\langle w_2, u_2 \rangle &= b\langle w_1, u_1 \rangle - f_{in}\langle v_2, d_2 \rangle \cup f_{out}\langle v_2, d_2 \rangle, \\ &\vdots \\ b\langle w_i, u_i \rangle &= b\langle w_{i-1}, u_{i-1} \rangle - f_{in}\langle v_i, d_i \rangle \cup f_{out}\langle v_i, d_i \rangle, \\ &\vdots \\ b\langle w_j, u_j \rangle &= b\langle w_{j-1}, u_{j-1} \rangle - f_{in}\langle v_j, d_j \rangle \cup f_{out}\langle v_j, d_j \rangle. \end{aligned}$$

Z definicji 6.2 i sposobu określenia $b\langle w_j, u_j \rangle$ wynika -

L.6.1. Dla dowolnego systemu generacyjnego SG i słowa w_j następujące warunki są równoważne:

- 1/ $w_j \in L_1/SG/$ i istnieje ślad u_j słowa w_j taki, że
 $b \langle w_j, u_j \rangle = f_{out} \langle v_j, d_j \rangle$ oraz $/b \langle w_{j-1}, u_{j-1} \rangle = f_{in}$
 $\langle v_j, d_j \rangle / \cap f_{out} \langle v_j, d_j \rangle = \emptyset$,
- 2/ $w_j \in L_2/SG/$.

Lemat ten zostanie wykorzystany w dowodzie następującego twierdzenia:

Tw.6.1. Dla każdego systemu generacyjnego SG język $L_2/SG/$ jest językiem regularnym.

Dowód W celu udowodnienia twierdzenia przedstawiony zostanie algorytm, który dla dowolnego systemu SG podaje sposób konstrukcji RS automatu A^2 takiego, że $L/A^2/ = L_2/SG/$. A^2 może nie być automatem minimalnym.

Algorytm A1.6.1

Dane: $SG = \{V, \Sigma, D, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}\}$.

Wyniki: RS automat $A^2 = \{V, S, F, I, s_0\}$ taki, że $L/A^2/ = L_2/SG/$.

Oznaczenia:

- i - numer kolejno generowanego stanu automatu A^2 , który może być stanem aktywnym tego automatu
- t - numer kolejno generowanej instrukcji automatu A^2 , która może należeć do części aktywnej tego automatu
- $\delta/I_t/$ - ma takie samo znaczenie jak w przypadku algorytmu 4.1.

Stan s_i , który może być stanem aktywnym automatu A^2 , jest parą $\langle s_{i1}, s_{i2} \rangle$, gdzie s_{i1} odpowiada stanowi automatu A^1 z algorytmu 4.1, a s_{i2} wartości $b \langle w_n, u_n \rangle$.

Początek

1. $i=0, t=0, s_0 = \langle \emptyset, \emptyset \rangle, S = \{s_0\}, F = \emptyset, I = \emptyset$.
2. Dla każdej pary $\langle v, d \rangle$ w Dz_{SG} takiej, że $f_{in} \langle v, d \rangle = \emptyset$;
 - a/ jeżeli $\langle f_{out} \langle v, d \rangle, f_{out} \langle v, d \rangle \rangle \notin S$, to kładzie się w A^2
 $i=i+1, s_i = \langle f_{out} \langle v, d \rangle, f_{out} \langle v, d \rangle \rangle, t=t+1, I_t = s_0 v \rightarrow s_i,$
 $\delta/I_t/ = d, F = F \cup \{s_i\},$

b/ jeżeli $f_{out} \langle v, d \rangle = \emptyset$ i $s_0 v \rightarrow s_0 \notin I$, to kładzie się w A^2 $t=t+1$, $I_t = s_0 v \rightarrow s_0$, $\delta/I_t = d$,

c/ jeżeli istnieje $s_x \in S$ taki, że $f_{out} \langle v, d \rangle = s_{x1} = s_{x2}$ i jeżeli $s_0 v \rightarrow s_x \notin I$, to $t=t+1$, $I_t = s_0 v \rightarrow s_x$, $\delta/I_t = d$.

3. Jeżeli w kroku 2.a nie wygenerowano żadnego stanu, to przejście do kroku 7.

4. Dla każdej pary $\langle v, d \rangle$ w Dz_{SG} i dla każdego stanu s_u uzyskanego w ostatnio wykonanym kroku generującym stany /krok 2.a lub 4.a/ spełniającego warunki - $s_{u1} \cap f_{kon} \langle v, d \rangle = \emptyset$, $f_{in} \langle v, d \rangle \subseteq s_{u1}$ - kładzie się $s_{x1} = s_{u1} - f_{dem} \langle v, d \rangle \cup f_{out} \langle v, d \rangle$, $s_{x2} = s_{u2} - f_{in} \langle v, d \rangle \cup f_{out} \langle v, d \rangle$, $s_x = \langle s_{x1}, s_{x2} \rangle$;

a/ jeżeli $s_x \notin S$, to kładzie się w A^2 $i=i+1$, $s_i = s_x$ $t=t+1$, $I_t = s_u v \rightarrow s_i$, $\delta/I_t = d$ jeżeli $s_{x2} = f_{out} \langle v, d \rangle$ i $/s_{u2} - f_{in} \langle v, d \rangle / \cap f_{out} \langle v, d \rangle = \emptyset$, to $F = F \cup \{s_x\}$,

b/ jeżeli $s_x \in S$ i $s_u v \rightarrow s_x \notin I$, to kładzie się w A^2 $t=t+1$, $I_t = s_u v \rightarrow s_x$, $\delta/I_t = d$.

5. Jeżeli w kroku 4.a otrzymano przynajmniej jeden stan, to przejście do kroku 4.

6. Jeżeli istnieją w A^2 stan $s_y \in S$ i symbol $v_y \in V$ takie, że $s_y v_y \rightarrow s_0 \in I$, to $F = F \cup \{s_0\}$.

7. Jeżeli istnieją symbol $v^1 \in V$ oraz stan $s^z \in S$ takie, że $s^z v^1$ nie jest poprzednikiem żadnej instrukcji w A^2 , to kładzie się $s^z v^1 \rightarrow s$.

8. Dla każdego $v^u \in V$ kładzie się w A^2 $sv^u \rightarrow s$.

Koniec

L.6.2. Dla każdego słowa $w_j = v_1 \dots v_j \dots v_j$, jeżeli A^2 jest automatem otrzymanym w wyniku działania algorytmu A1.6.1 i $w_j \in L/A^2/$, to $w_j \in L_2/SG/$.

Dowód Jeżeli $w_j \in L/A^2/$, to istnieją ciąg stanów $s_0 s_1 \dots s_i \dots s_{j-1} s_j$ i ciąg instrukcji $I_1 \dots I_i \dots I_j$ takich, że $I_1 = s_0 v_1 \rightarrow s_1$, $\dots, I_i = s_{i-1} v_i \rightarrow s_i, \dots, I_j = s_{j-1} v_j \rightarrow s_j$, $\delta/I_1 = d_1, \dots, \delta/I_i = d_i, \dots, \delta/I_j = d_j$ i $s_{j2} = f_{out} \langle v_j, d_j \rangle$.

Ze sposobu wyznaczania stanów i instrukcji automatu A^2 wynika, że:

1. ciągi $s_{01}s_{11}\dots s_{i1}\dots s_{/j-1/1}$ i $d_1\dots d_{i1}\dots d_j$ spełniają warunki Wr.4.1 i Wr.4.2 czyli, że $w_j \in L_1/SG/$,
2. $/s_{/j-1/2} - f_{in} \langle v_j, d_j \rangle / \wedge f_{out} \langle v_j, d_j \rangle = \emptyset$.

Spełniony jest zatem warunek 1 lematu 6.1. Możemy więc wnioskować, że $w_j \in L_2/SG/$.

L.6.3. Dla każdego systemu generacyjnego SG i dla każdego słowa $w'_j \in L_2/SG/$, jeżeli A^2 jest automatem otrzymanym w wyniku działania algorytmu A1.6.1, to $w'_j \in L/A^2/$.

Dowód Niech $u'_j = d'_1\dots d'_{i1}\dots d'_j$ będzie śladem w'_j takim, że $b \langle w'_j, u'_j \rangle = f_{out} \langle w'_j, d'_j \rangle$ oraz $/b \langle w'_{j-1}, u'_{j-1} \rangle - f_{in} \langle v'_j, d'_j \rangle / \wedge f_{out} \langle v'_j, d'_j \rangle = \emptyset$. Istnieje wówczas ciąg stanów systemu SG $r'_j = s'_0s'_1\dots s'_{j-1}$ taki, że para r'_j, u'_j spełnia warunki Wr.4.1 i Wr.4.2.

Ze sposobu konstrukcji automatu A^2 wynika, że istnieją w A^2 ciąg stanów $s_0s_1\dots s_{i1}\dots s_j$ oraz ciąg instrukcji $I^1\dots I^j$ takich, że $s_{01}=s'_0$, $s_{11}=s'_1$, $s_{i1}=s'_{i1}$, ..., $s_{/j-1/1}=s'_{j-1}$, $s_{j1}=s'_j$, gdzie $s'_j = s'_{j-1} - f_{dem} \langle v'_j, d'_j \rangle \cup f_{out} \langle v'_j, d'_j \rangle$;
 $I^1 = s_0v'_1 \rightarrow s_1, \dots, I^i = s_{i-1}v'_i \rightarrow s_i, \dots, I^j = s_{j-1}v'_j \rightarrow s_j$ oraz $s_{j2} = f_{out} \langle v'_j, d'_j \rangle$ i $/s_{/j-1/2} - f_{in} \langle v'_j, d'_j \rangle / \wedge f_{out} \langle v'_j, d'_j \rangle = \emptyset$.

Stąd i ze sposobu konstrukcji zbioru stanów końcowych F automatu A^2 wynika, że $w'_j \in L/A^2/$.

Dowód lematu 6.3 kończy dowód twierdzenia 6.1.

6.3. Regularność języka LPPW/KWP/

Między modelem systemu KWP i językiem LPPW/KWP/, a systemem SG i językiem $L_2/SG/$ istnieją analogie, które pozwalają pokazać ciekawe własności systemu KWP i języka LPPW/KWP/. Własności te zostaną sformułowane w postaci wniosków.

Wn.6.1. Dla systemu KWP istnieje system SG taki, że $LPPW/KWP/ = L_2/SG/$.

Wn.6.2. Dla każdego systemu generacyjnego SG istnieje system KWP taki, że $L_2/SG/ = LPPW/KWP/$.

Uzasadnienie wniosków 6.1 i 6.2 jest bardzo podobne do uzasadnień wniosków 4.1 i 4.2.

Z wniosków 6.1 i 6.2 wynika bezpośrednio -

Wn.6.3. Klasy języków $\mathcal{L}_{PPW/KWP/}$ i $\mathcal{L}_{2/SG/}$ są równoważne.

Z twierdzenia 6.1 i wniosku 6.3 wynika -

Wn.6.4. Dla każdego systemu KWP język LPPW/KWP/ jest językiem regularnym.

6.4. Algorytm rozwiązania zadania Z2

Automat A^2 /wynik algorytmu A1.6.1/ o minimalnej liczbie stanów zostanie wykorzystany do rozwiązania zadania Z2. Algorytm 6.2 będący opisem rozwiązania zadania jest podobny do algorytmu A1.4.2.

Algorytm A1.6.2

Rozwiązanie zadania Z2

Dane: Automat $A^2 = \{\hat{V}, S, F, I, s_0\}$ o minimalnej liczbie stanów taki, że $L/A^2/ = LPPW/KWP/$.

Para $\langle \hat{w}_n, \hat{u}_n \rangle$ $\hat{w}_n = \hat{v}_1 \dots \hat{v}_i \dots \hat{v}_n$, $\hat{u}_n = \hat{s}d_1 \dots \hat{s}d_1 \dots \hat{s}d_n$.

Zbiór warunków, które muszą spełniać typy struktur należące do podzbioru TSD_u .

Wyniki: Stwierdzenie - czy słowo $\hat{w}_n$ akceptowane jest przez automat A^2 oraz czy każda ze struktur $\hat{s}d_i$ $1 \leq i \leq n$ spełnia warunki $var_{T\hat{s}d_i}$.

Oznaczenia: Identyczne jak dla algorytmu A1.4.2.

begin

$s^1 := s_0$; $p := \text{true}$; $i := 1$; writeln;

while $i \leq n \wedge p \neq \text{false}$ do

begin

$p := \text{false}$;

for $j := 1$ to i do

if $s_j^1 = s^1 \wedge \hat{v}_j = \hat{v}_1 \wedge \delta/I_j/ = T\hat{s}d_i$ then

if spełwarunki / $\hat{T}sd_i$ / = true then

begin p:=true ; $s^1:=s_j^2$; i:=i+1 end

end;

if $i \neq n+1$ then

writeln /'Błędny opis procesu, działanie' $\langle \hat{v}_i, \hat{s}_i \rangle$ 'nie może być wykonane przez system'/'

else

if $s^1 \notin F$ then

writeln /'Para ciągów' $\langle \hat{w}_n, \hat{u}_n \rangle$ 'opisuje proces, który nie jest poprawnym procesem wspomaganym'/'

else

writeln /'Para ciągów' $\langle \hat{w}_n, \hat{u}_n \rangle$ 'opisuje poprawny proces wspomagany'/'

end.

Oszacowanie maksymalnego czasu wykonania algorytmu 6.2

- Oszacowanie to jest identyczne jak w przypadku algorytmu 4.2.

Przykład 6.1

Rozważmy system KWP o opisie działania oprogramowania $\hat{ODO}_{KWP}$ przedstawionym w tabeli 6.1.

Na rysunku 6.1 przedstawiono diagram stanów automatu A^2 akceptującego język LPPW/KWP/.

Niech $\hat{d}_1, \hat{d}_2, \dots, \hat{d}_5$ będą typami struktur $\hat{s}_1, \hat{s}_2, \dots, \hat{s}_5$.

Z rysunku 6.1 wynika, że procesy $\langle \hat{v}_1, \hat{s}_1 \rangle \langle \hat{v}_2, \hat{s}_2 \rangle \langle \hat{v}_4, \hat{s}_4 \rangle$ $\langle \hat{v}_1, \hat{s}_1 \rangle \langle \hat{v}_3, \hat{s}_3 \rangle$ są poprawnymi procesami wspomaganymi, natomiast procesy $\langle \hat{v}_1, \hat{s}_1 \rangle \langle \hat{v}_3, \hat{s}_3 \rangle \langle \hat{v}_2, \hat{s}_2 \rangle$, czy $\langle \hat{v}_1, \hat{s}_1 \rangle \langle \hat{v}_2, \hat{s}_2 \rangle \langle \hat{v}_2, \hat{s}_2 \rangle$ nimi nie są.

Na rysunku 6.2 przedstawiono przykładowe struktury procesów, których wspomaganie opisane jest przez poprawne procesy wspomagane /przyjęto założenie, że działaniu F_i odpowiada działanie wspomagane $\langle \hat{v}_i, \hat{s}_i \rangle$, $i = 1, \dots, 5$ /.

Tab.6.1

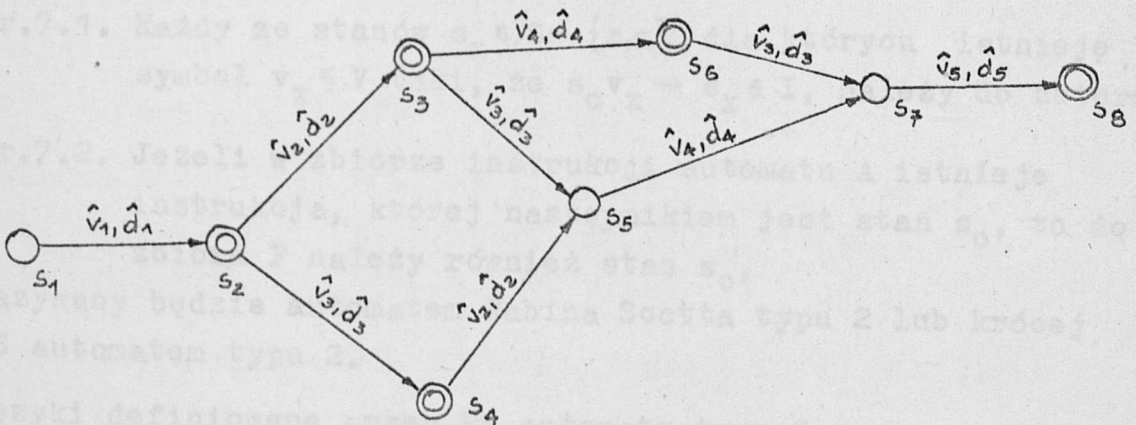
Opis działania oprogramowania systemu KWP z przykładu 6.1.

$\hat{P}_x$	$\hat{C}_x$	$\langle \hat{V}_x, \hat{d}_x \rangle$	$\hat{f}_{in}$ $\langle \hat{V}_x, \hat{d}_x \rangle$	$\hat{f}_{out}$ $\langle \hat{V}_x, \hat{d}_x \rangle$	$\hat{f}_{kon}$ $\langle \hat{V}_x, \hat{d}_x \rangle$	$\hat{f}_{dem}$ $\langle \hat{V}_x, \hat{d}_x \rangle$	$\hat{f}_{dok}$ $\langle \hat{V}_x, \hat{d}_x \rangle$
$\hat{P}_1$	$\hat{V}_1$	$\langle \hat{V}_1, \hat{d}_1 \rangle$	$\emptyset$	$\hat{A}$	$\hat{A}, \hat{B}, \hat{C}, \hat{D}, \hat{E}$	$\emptyset$	$+\hat{B}$
$\hat{P}_2$	$\hat{V}_2$	$\langle \hat{V}_2, \hat{d}_2 \rangle$	$\hat{A}$	$\hat{B}$	$\hat{D}, \hat{E}, \hat{D}$	$\emptyset$	$+\hat{B}$
$\hat{P}_3$	$\hat{V}_3$	$\langle \hat{V}_3, \hat{d}_3 \rangle$	$\hat{A}$	$\hat{C}$	$\hat{E}, \hat{C}$	$\emptyset$	$\hat{d}_3^d$
$\hat{P}_4$	$\hat{V}_4$	$\langle \hat{V}_4, \hat{d}_4 \rangle$	$\hat{B}$	$\hat{D}$	$\hat{D}, \hat{E}$	$\hat{B}$	$\hat{d}_4^d$
$\hat{P}_5$	$\hat{V}_5$	$\langle \hat{V}_5, \hat{d}_5 \rangle$	$\hat{D}, \hat{C}$	$\hat{E}$	$\hat{E}$	$\emptyset$	$\hat{d}_5^d$

$$\hat{P} = \{ \hat{P}_1, \hat{P}_2, \hat{P}_3, \hat{P}_4, \hat{P}_5 \}, \quad \hat{C} = \{ \hat{V}_1, \hat{V}_2, \hat{V}_3, \hat{V}_4, \hat{V}_5 \}, \quad TSD = \hat{D}_M \cup \hat{D}_U \cup \hat{D}_D,$$

$$\hat{D}_M = \{ \hat{A}, \hat{B}, \hat{C}, \hat{D}, \hat{E} \}, \quad \hat{D}_U = \{ \hat{d}_1, \hat{d}_2, \hat{d}_3, \hat{d}_4, \hat{d}_5 \}, \quad \hat{D}_D = \{ \hat{d}_3^d, \hat{d}_4^d, \hat{d}_5^d, +\hat{B} \}$$

⊙ - stan końcowy



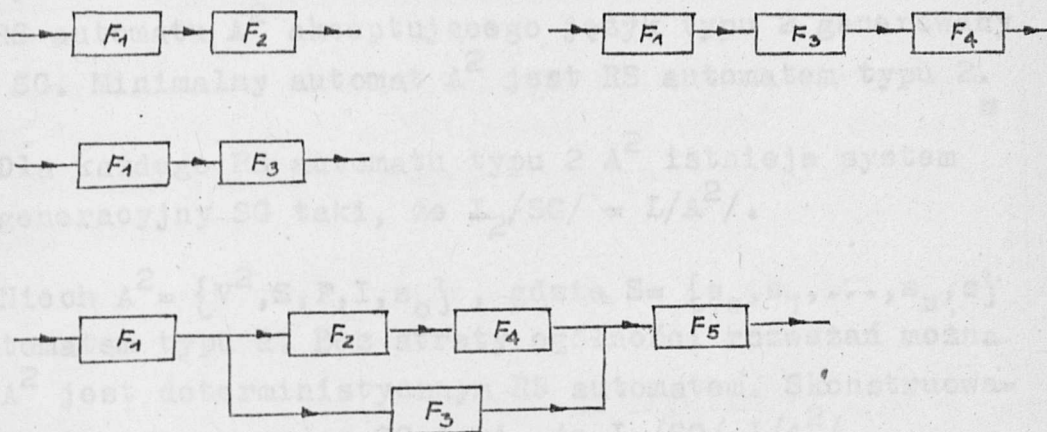
$$s_1 = \langle \emptyset, \emptyset \rangle, \quad s_2 = \langle \{ \hat{A} \}, \{ \hat{A} \} \rangle, \quad s_3 = \langle \{ \hat{A}, \hat{B} \}, \{ \hat{B} \} \rangle, \quad s_4 = \langle \{ \hat{A}, \hat{C} \}, \{ \hat{C} \} \rangle,$$

$$s_5 = \langle \{ \hat{A}, \hat{B}, \hat{C} \}, \{ \hat{B}, \hat{C} \} \rangle, \quad s_6 = \langle \{ \hat{A}, \hat{D} \}, \{ \hat{D} \} \rangle, \quad s_7 = \langle \{ \hat{A}, \hat{C}, \hat{D} \}, \{ \hat{C}, \hat{D} \} \rangle,$$

$$s_8 = \langle \{ \hat{A}, \hat{C}, \hat{D}, \hat{E} \}, \{ \hat{E} \} \rangle,$$

$$F = \{ s_2, s_3, s_4, s_6, s_8 \}$$

Rys.6.1. Diagram stanów automatu A^2 akceptującego język LPPW systemu KWP z przykładu 6.1



Rys.6.2. Przykładowe struktury procesów, których wspomaganie jest opisane przez poprawne procesy wspomagane

7. WŁASNOŚCI KLAS JĘZYKOW $\mathcal{L}_{2/SG/}$ i $\mathcal{L}_{PPW/KWP/}$

7.1. Definicja RS automatu typu 2

Def.7.1. RS automat $A = \{V, S, F, I, s_0\}$ spełniający warunki:

Wr.7.1. Każdy ze stanów $s_x \in S - \{s_0\}$, dla których istnieje symbol $v_x \in V$ taki, że $s_0 v_x \rightarrow s_x \in I$, należy do zbioru F .

Wr.7.2. Jeżeli w zbiorze instrukcji automatu A istnieje instrukcja, której następnikiem jest stan s_0 , to do zbioru F należy również stan s_0 ,

nazywany będzie automatem Rabina Scotta typu 2 lub krócej RS automatem typu 2.

Języki definiowane przez RS automaty typu 2 nazywane będą językami regularnymi typu 2. Klasa tych języków oznaczana będzie $\mathcal{L}_{R2}$.

L.7.1. Dla każdego języka typu 2 - generowanego przez system generacyjny SG - istnieje RS automat typu 2 A^2 taki, że $L/A^2/ = L_{2/SG/}$.

Dowód wynika bezpośrednio z twierdzenia 5.2 i algorytmu konstrukcji RS automatu A^2 akceptującego język typu 2 generowany przez system SG. Minimalny automat A^2 jest RS automatem typu 2. ■

L.7.2. Dla każdego RS automatu typu 2 A^2 istnieje system generacyjny SG taki, że $L_2/SG/ = L/A^2/$.

Dowód Niech $A^2 = \{V^2, S, F, I, s_0\}$, gdzie $S = \{s_0, s_1, \dots, s_n, s\}$ będzie RS automatem typu 2. Bez straty ogólności rozważań można założyć, że A^2 jest deterministycznym RS automatem. Skonstruowany zostanie system generacyjny SG taki, że $L_2/SG/ = L/A^2/$.

Niech $SG = \{V, \Sigma, D, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}\}$, gdzie $V = V^2$, $\Sigma = S - \{s\} \cup \{\Delta\}$, Δ - jest symbolem nienależącym do $S - \{s\}$, $D = S - \{s\}$, Dz_{SG} jest zbiorem par $\langle a_1, s_1 \rangle$, dla których instrukcja $s_y a_1 \rightarrow s_1$ należy do części aktywnej automatu A^2 , a funkcje f_{in} , f_{out} , f_{kon} , f_{dem} określone są w następujący sposób:

1. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j = s_0$, $s_i \in S - \{s_0, s\}$ / kładzie się w SG $f_{in} \langle a_1, s_i \rangle = \emptyset$, $f_{out} \langle a_1, s_i \rangle = \{s_i, \Delta\}$, $f_{kon} \langle a_1, s_i \rangle = S - \{s_0\}$, $f_{dem} \langle a_1, s_i \rangle = \emptyset$.
2. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j \in S - \{s_0, s\}$, $s_i \notin F \cup \{s, s_0\}$ / w A^2 kładzie się w SG $f_{in} \langle a_1, s_i \rangle = \{s_j\}$, $f_{out} \langle a_1, s_i \rangle = \{s_i\}$, $f_{kon} \langle a_1, s_i \rangle = \emptyset$, $f_{dem} \langle a_1, s_i \rangle = \{s_j\}$.
3. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j \in S - \{s, s_0\}$, $s_i \in F - \{s_0\}$ / w A^2 kładzie się w SG $f_{in} \langle a_1, s_i \rangle = \{s_j, \Delta\}$, $f_{out} \langle a_1, s_i \rangle = \{s_i, \Delta\}$, $f_{kon} \langle a_1, s_i \rangle = \emptyset$, $f_{dem} \langle a_1, s_i \rangle = \{s_j\}$.
4. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j \in S - \{s, s_0\}$, $s_i = s_0$ w A^2 kładzie się w SG $f_{in} \langle a_1, s_i \rangle = \{s_j, \Delta\}$, $f_{out} \langle a_1, s_i \rangle = \emptyset$, $f_{kon} \langle a_1, s_i \rangle = \emptyset$, $f_{dem} \langle a_1, s_i \rangle = \{s_j, \Delta\}$.
5. Dla instrukcji $s_0 a_1 \rightarrow s_0$ w A^2 kładzie się w SG $f_{in} \langle a_1, s_i \rangle = \emptyset$, $f_{out} \langle a_1, s_i \rangle = \emptyset$, $f_{kon} \langle a_1, s_i \rangle = S - \{s_0\}$, $f_{dem} \langle a_1, s_i \rangle = \emptyset$.

Jak łatwo sprawdzić $L_2/SG/ = L/A^2/$. ■

Tw.7.1. Klasy języków $\mathcal{L}_{R2}$ i $\mathcal{L}_{2/SG/}$ są równoważne.

Dowód - bezpośrednio z lematów 7.1 i 7.2. ■

7.2. Algebraiczne własności klasy $\mathcal{L}_{R2}$

Poniżej omówione zostaną algebraiczne własności klasy $\mathcal{L}_{R2}$. Ponieważ część dowodów lematów i twierdzeń jest bardzo podobna do dowodów z podrozdziału 5.3, zostaną one przedstawione skrótowo - w postaci szkiców.

Tw.7.2. Klasa języków $\mathcal{L}_{R2}$ jest zamknięta ze względu na operację podstawiania.

Szkic dowodu Niech L będzie językiem regularnym typu 2 nad alfabetem $V = \{a_1, \dots, a_1, \dots, a_n\}$. Każdemu symbolowi a_i tego alfabetu zostaje przyporządkowany nowy alfabet W^i . Rozważmy dowolne podstawienie $\varphi: V^* \rightarrow 2^{W^*}$, gdzie $W = \bigcup_{i=1}^n W_i$ oraz dla każdego $i=1, \dots, n$, $\varphi/a_i/$ jest językiem regularnym typu 2 nad W^i . Należy pokazać, że $\varphi/L/$ jest językiem regularnym typu 2.

Niech $A = \{V, S, F, I, s_0\}$ będzie RS automatem typu 2 definiującym język L i niech $A^i = \{V^i, S^i, F^i, I^i, s_0\}$ będą, dla $i=1, \dots, n$ RS automatami typu 2 definiującymi języki $\varphi/a_1/, \dots, \varphi/a_n/$.

W dalszej części dowód przebiega według następującego schematu:

1. Podobnie jak w dowodzie twierdzenia 5.2 konstruuje się automat A^0 .
2. Pokazuje się, że A^0 jest RS automatem typu 2.
3. Pokazuje się, że $\varphi/L/ = L/A^0/$.

L.7.3. Klasa $\mathcal{L}_{R2}$ jest zamknięta ze względu na operację sumy i domknięcia języków.

Dowód Rozważmy dwa RS automaty A_1, A_2 opisane w dowodzie twierdzenia 5.4. Są one zarówno RS automatami typu 1 jak i typu 2 i definiują języki $\{a, b\}$ i $\{a\}^*$. Na podstawie twierdzenia 5.3 klasa $\mathcal{L}_{R2}$ jest zamknięta ze względu na operacje sumy i domknięcia języków. ■

L.7.4. Klasa $\mathcal{L}_{R2}$ jest zamknięta ze względu na operację przedłużania.

Dowód można otrzymać przez przekopiowanie dowodu lematu 5.5 i zastąpienie symbolu $\mathcal{L}_{R1}$ symbolem $\mathcal{L}_{R2}$. ■

L.7.5. Dla każdego $L_1 \in \mathcal{L}_{R2}$, jeżeli $x_n \in L_1$, $x_n = a_1 \dots a_{n-1} a_n$, to słowo $x_1 = a_1$ również należy do L_1 .

Dowód - bezpośrednio z definicji RS automatu typu 2. ■

L.7.6. Klasa $\mathcal{L}_{R2}$ nie jest zamknięta ze względu na operację złożenia.

Dowód można otrzymać przez przekopiowanie dowodu lematu 5.7 i zastąpienie symbolu $\mathcal{L}_{R1}$ symbolem $\mathcal{L}_{R2}$ i powołanie się na lemat 7.5. ■

L.7.7. Klasa $\mathcal{L}_{R2}$ jest właściwą podklasą klasy $\mathcal{L}_R$.

Dowód Język $\{ab\}$ należy do klasy $\mathcal{L}_R$, a nie należy do klasy $\mathcal{L}_{R2}$. ■

L.7.8. Klasy $\mathcal{L}_{R1}$ i $\mathcal{L}_{R2}$ nie są równoważne.

Dowód Rozważmy języki $L_1 = \{a, ab, abc\}$ i $L_2 = \{a, abc\}$; $L_1 \in \mathcal{L}_{R1}$, $L_2 \in \mathcal{L}_{R2}$. Na mocy lematu 5.5 $L_2 \notin \mathcal{L}_{R1}$, co kończy dowód. ■

L.7.9. Klasa $\mathcal{L}_{R2}$ nie jest zamknięta ze względu na operację różnicy.

Dowód można otrzymać przez przekopiowanie dowodu lematu 5.9, zastąpienie symbolu $\mathcal{L}_{R1}$ symbolem $\mathcal{L}_{R2}$ i powołanie się na lemat 7.5. ■

L.7.10. Klasa języków $\mathcal{L}_{R2}$ nie jest zamknięta ze względu na uzupełnianie.

Dowód Rozważmy dwa języki L_1, L_2 ; $L_1 = \{a, ab\}$, $L_2 = \{abc\}$; $L_1 \in \mathcal{L}_{R2}$, a $L_2 \notin \mathcal{L}_{R2}$. Oczywiście $L_2 \in /V^* - L_1/$. Założmy, że $/V^* - L_1/ \in \mathcal{L}_{R2}$. Wówczas na mocy lematu 7.5 powinna zachodzić implikacja $/abc \in /V^* - L_1// \Rightarrow /a \in /V^* - L_1//$. Jednak $a \in L_1$, a zatem $/V^* - L_1/ \notin \mathcal{L}_{R2}$. Klasa $\mathcal{L}_{R2}$ nie jest więc zamknięta ze względu na uzupełnianie. ■

Tw.7.3. Klasa języków $\mathcal{L}_{R2}$ jest zamknięta ze względu na operację iloczynu.

Szkic dowodu Niech A_1 i A_2 będą dowolnymi deterministycznymi RS automatami typu 2 i niech:

$A_1 = \{V_1, S_1, F_1, M_1, s_0^1\}$, gdzie $S_1 = \{s_0^1, s_1^1, \dots, s_{m_1}^1, s^1\}$;

$A_2 = \{V_2, S_2, F_2, M_2, s_0^2\}$, gdzie $S_2 = \{s_0^2, s_1^2, \dots, s_{m_2}^2, s^2\}$,

a s^1, s^2 są stanami zakazanymi automatów A_1, A_2 . Bez straty ogólności rozważań można założyć, że A_1 i A_2 mają wspólny alfabet $V=V_1=V_2$.

Należy pokazać, że wychodząc od automatów A_1, A_2 można skonstruować automat A_3 akceptujący język $L/A_1/\cap L/A_2/$ oraz, że A_3 można przekształcić do postaci RS automatu typu 2.

W dalszej części dowód przebiega według następującego schematu:

1. Analogicznie jak w dowodzie twierdzenia 5.5 konstruuje się automat A_3 .
2. Pokazuje się, że A_3 można przekształcić do postaci RS automatu typu 2.
3. Pokazuje się, że $L/A_3/ = L/A_1/\cap L/A_2/$.

L.7.11. Klasa $\mathcal{L}_{R2}$ nie jest zamknięta ze względu na operację odbicia zwierciadlanego.

Dowód Rozważmy język $L = \{a, ab\}$. Oczywiście $L \in \mathcal{L}_{R2}$.

Język $L^R = \{a, ba\}$ nie jest definiowany przez żaden RS automat typu 2, gdyż gdyby tak było, to język definiowany przez automat akceptujący L^R musiałby /na podstawie lematu 7.5/ zawierać słowo $b \notin L^R$.

Wniosek 7.1 podsumowuje wyniki osiągnięte w tym podrozdziale.

Wn.7.1. Klasa języków $\mathcal{L}_{R2}$ jest zamknięta ze względu na operacje podstawiania, sumy, przedłużania, domknięcia i iloczynu, a nie jest zamknięta ze względu na operacje złożenia, różnicy i odbicia zwierciadlanego oraz uzupełnianie.

7.3. Algebraiczne własności klas $\mathcal{L}_{2/SG/}$ i $\mathcal{L}_{PPW/KWP/}$

Tw.7.4. Klasa języków $\mathcal{L}_{2/SG/}$ jest zamknięta ze względu na operacje podstawiania, sumy, przedłużania, domknięcia i iloczynu, a nie jest zamknięta ze względu na operacje złożenia, różnicy i odbicia zwierciadlanego oraz uzupełnianie.

Dowód - bezpośrednio z twierdzenia 7.1 i wniosku 7.1. ■

Wn.7.2. Klasa języków $\mathcal{L}_{PPW/KWP/}$ jest zamknięta ze względu na operacje podstawiania, sumy, przedłużania, domknięcia i iloczynu, a nie jest zamknięta ze względu na operacje złożenia, różnicy i odbicia zwierciadlanego oraz uzupełnianie.

Uzasadnienie Wniosek wynika bezpośrednio z twierdzenia 7.4 i wniosku 6.3. ■

Tw.7.5. Klasy $\mathcal{L}_R$ i $\mathcal{L}_{2/SG/}$ nie są równoważne.

Dowód wynika bezpośrednio z twierdzenia 7.1 i lematu 7.7. ■

Tw.7.6. Klasy $\mathcal{L}_{1/SG/}$ i $\mathcal{L}_{2/SG/}$ nie są równoważne.

Dowód - bezpośrednio z twierdzeń 5.1 i 7.1 oraz lematu 7.8. ■

Wn.7.3. Klasy $\mathcal{L}_R$ i $\mathcal{L}_{PPW/KWP/}$ nie są równoważne.

Uzasadnienie Wniosek wynika z twierdzenia 7.5 i wniosku 6.3. ■

Wn.7.4. Klasy $\mathcal{L}_{PW}$ i $\mathcal{L}_{PPW/KWP/}$ nie są równoważne.

Uzasadnienie Wniosek wynika z twierdzenia 7.6 i wniosków 4.3, 6.3. ■

8. ROZWIĄZANIE DRUGIEGO ZADANIA KOORDYNACJI

8.1. Rozszerzony system generacyjny RSG

Def.8.1. Rozszerzonym systemem generacyjnym RSG nazywać będziemy system $RSG = \{V, \Sigma, D, E, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}, f_{dok}\}$, gdzie: $\{V, \Sigma, D, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}\}$ - system generacyjny
 E - skończony niepusty zbiór zawierający symbol $\dot{a}$ taki, że system $\{E, \circ, \dot{a}\}$ jest półgrupą z jednością,
 f_{dok} - funkcja - $f_{dok}: Dz_{SG} \rightarrow E$.

Def.8.2. Zbiór wszystkich słów $w_n \in L_2/SG/$, $w_n = v_1 \dots v_i \dots v_n$, dla których istnieją ślady $u_n = d_1 \dots d_i \dots d_n$ takie, że $f_{dok} \langle v_n, d_n \rangle \neq \dot{a}$ nazywany będzie językiem generowanym przez RSG i oznaczany $L/RSG/$.

Klasa wszystkich języków generowanych przez rozszerzone systemy generacyjne oznaczana będzie $\mathcal{L}_{RSG}$.

8.2. Regularność języka $L/RSG/$

Tw.8.1. Dla każdego rozszerzonego systemu generacyjnego RSG język $L/RSG/$ jest językiem regularnym.

Szkic dowodu W celu udowodnienia twierdzenia przedstawiony zostanie algorytm, który dla dowolnego RSG podaje sposób konstrukcji RS automatu A^3 takiego, że $L/A^3/ = L/RSG/$. Dowód zachodzenia równości $L/A^3/ = L/RSG/$ jest bardzo podobny do dowodów lematów 6.2 i 6.3 i dlatego zostanie pominięty.

Algorytm A1.8.1

Dane: Rozszerzony system generacyjny $RSG = \{V, \Sigma, D, E, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}, f_{dok}\}$.

Wyniki: RS automat $A^3 = \{V, S, F, I, s_0\}$ taki, że $L/A^3/ = L/RSG/$.

Oznaczenia:

- i - numer kolejno generowanego stanu automatu A^3 , który może należeć do części aktywnej

- t - numer kolejno generowanej instrukcji automatu A^3 , która może należeć do części aktywnej
- $\delta/I_t/$ - ma takie same znaczenie jak w przypadku algorytmu 4.1.

Stan s_i automatu A^3 jest trójką $\langle s_{i1}, s_{i2}, s_{i3} \rangle$, gdzie para $\langle s_{i1}, s_{i2} \rangle$ odpowiada stanowi automatu A^2 z algorytmu A1.6.1, a s_{i3} reprezentuje element zbioru E.

Początek

1. $i=0, t=0, s_0 = \langle \emptyset, \emptyset, \dot{a} \rangle, S = \{s_0\}, F = \emptyset, I = \emptyset$.
2. Dla każdej pary $\langle v, d \rangle$ w Dz_{SG} takiej, że $f_{in} \langle v, d \rangle = \emptyset$;
 - a/ jeżeli $\langle f_{out} \langle v, d \rangle, f_{out} \langle v, d \rangle, f_{dok} \langle v, d \rangle \rangle \notin S$, to kładzie się w A^3 $i=i+1, s_i = \langle f_{out} \langle v, d \rangle, f_{out} \langle v, d \rangle, f_{dok} \langle v, d \rangle \rangle$, $t=t+1, I_t = s_0 v \rightarrow s_i, \delta/I_t/ = d$, jeżeli $f_{dok} \langle v, d \rangle \neq \dot{a}$, to $F = F \cup \{s_i\}$,
 - b/ jeżeli $f_{out} \langle v, d \rangle = \emptyset$ i $f_{dok} \langle v, d \rangle = \dot{a}$ i $s_0 v \rightarrow s_0 \notin I$, to kładzie się w A^3 $t=t+1, I_t = s_0 v \rightarrow s_0, \delta/I_t/ = d$,
 - c/ jeżeli istnieje $s_x \in S$ taki, że $f_{out} \langle v, d \rangle = s_{x1} = s_{x2}$ i $f_{dok} \langle v, d \rangle = s_{x3}$ i jeżeli $s_0 v \rightarrow s_x \notin I$, to $t=t+1, I_t = s_0 v \rightarrow s_x, \delta/I_t/ = d$.
3. Jeżeli w kroku 2.a nie wygenerowano żadnego stanu, to przejście do kroku 6.
4. Dla każdej pary $\langle v, d \rangle$ w Dz_{SG} i dla każdego stanu s_u uzyskanego w ostatnio wykonanym kroku generującym stany /krok 2.a lub 4.a/ spełniającego warunki $f_{in} \langle v, d \rangle \subseteq s_{u1}$ i $s_{u1} \cap f_{kon} \langle v, d \rangle = \emptyset$ kładzie się $s_{x1} = s_{u1} - f_{dem} \langle v, d \rangle \cup f_{out} \langle v, d \rangle$, $s_{x2} = s_{u2} - f_{in} \langle v, d \rangle \cup f_{out} \langle v, d \rangle, s_{x3} = f_{dok} \langle v, d \rangle$, $s_x = \langle s_{x1}, s_{x2}, s_{x3} \rangle$;
 - a/ jeżeli $s_x \notin S$, to kładzie się w A^3 $i=i+1, s_i = s_x, t=t+1, I_t = s_u v \rightarrow s_i, \delta/I_t/ = d$ i jeżeli $s_{x2} = f_{out} \langle v, d \rangle$ oraz $/s_{u2} - f_{in} \langle v, d \rangle / \cap f_{out} \langle v, d \rangle = \emptyset$ i $f_{dok} \langle v, d \rangle \neq \dot{a}$, to $F = F \cup \{s_x\}$,
 - b/ jeżeli $s_x \in S$ i $s_u v \rightarrow s_x \notin I$, to kładzie się w A^3 $t=t+1, I_t = s_u v \rightarrow s_x, \delta/I_t/ = d$.
5. Jeżeli w kroku 4.a wygenerowano przynajmniej jeden stan, to przejście do kroku 4.

6. Jeżeli istnieją symbol $v^1 \in V$ oraz stan $s^z \in S$ takie, że $s^z v^1$ nie jest poprzednikiem żadnej instrukcji w A^3 , to kładzie się $s^z v^1 \rightarrow s$.
7. Dla każdego $v^u \in V$ kładzie się w A^3 $sv^u \rightarrow s$.

Koniec

8.3. Algebraiczne własności klasy $\mathcal{L}_{RSG}$

L.8.1. Dla każdego RS automatu A^3 istnieje rozszerzony system generacyjny RSG taki, że $L/RSG/ = L/A^3/$.

Dowód Niech $A = \{V^3, S, F, I, s_0\}$ będzie dowolnym deterministycznym RS automatem posiadającym dokładnie jeden stan zakazany s . Skonstruowany zostanie RSG taki, że $L/RSG/ = L/A/$.

Niech $RSG = \{V, \Sigma, D, E, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}, f_{dok}\}$, gdzie: $V = V^3$, $\Sigma = S - \{s\} \cup \{\Delta\}$, Δ - jest symbolem nienależącym do $S - \{s\}$, $D = S - \{s\}$, $E = F \cup \{\dot{\alpha}\}$, Dz_{SG} jest zbiorem par $\langle a_1, s_1 \rangle$, dla których instrukcja $s_1 a_1 \rightarrow s_1$ należy do części aktywnej automatu A^3 , a funkcje f_{in} , f_{out} , f_{kon} , f_{dem} , f_{dok} określone są w następujący sposób:

1. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j = s_0$, $s_i \notin F \cup \{s_0, s\}$ / w A^3 kładzie się w RSG $f_{in} \langle a_1, s_i \rangle = \emptyset$, $f_{out} \langle a_1, s_i \rangle = \{s_i, \Delta\}$, $f_{kon} \langle a_1, s_i \rangle = S - \{s_0\}$, $f_{dem} \langle a_1, s_i \rangle = \emptyset$, $f_{dok} \langle a_1, s_i \rangle = \dot{\alpha}$.
2. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j = s_0$, $s_i \in F - \{s_0\}$ / w A^3 kładzie się w RSG $f_{in} \langle a_1, s_i \rangle = \emptyset$, $f_{out} \langle a_1, s_i \rangle = \{s_i, \Delta\}$, $f_{kon} \langle a_1, s_i \rangle = S - \{s_0\}$, $f_{dem} \langle a_1, s_i \rangle = \emptyset$, $f_{dok} \langle a_1, s_i \rangle = s_i$.
3. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j \in S - \{s_0, s\}$, $s_i \notin F \cup \{s, s_0\}$ / w A^3 kładzie się w RSG $f_{in} \langle a_1, s_i \rangle = \{s_j\}$, $f_{out} \langle a_1, s_i \rangle = \{s_i\}$, $f_{kon} \langle a_1, s_i \rangle = \emptyset$, $f_{dem} \langle a_1, s_i \rangle = \{s_j\}$, $f_{dok} \langle a_1, s_i \rangle = \dot{\alpha}$.
4. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j \in S - \{s, s_0\}$, $s_i \in F - \{s_0\}$ / w A^3 kładzie się w RSG $f_{in} \langle a_1, s_i \rangle = \{s_j, \Delta\}$, $f_{out} \langle a_1, s_i \rangle = \{s_i, \Delta\}$, $f_{kon} \langle a_1, s_i \rangle = \emptyset$, $f_{dem} \langle a_1, s_i \rangle = \{s_j\}$, $f_{dok} \langle a_1, s_i \rangle = s_i$.
5. Dla każdej instrukcji $s_j a_1 \rightarrow s_i$, $s_j \in S - \{s, s_0\}$, $s_i = s_0$ w A^3 kładzie się w RSG $f_{in} \langle a_1, s_i \rangle = \{s_j, \Delta\}$, $f_{out} \langle a_1, s_i \rangle = \emptyset$,

$f_{kon} \langle a_1, s_i \rangle = \emptyset$, $f_{dem} \langle a_1, s_i \rangle = \{s_j, \Delta\}$; jeżeli $s_0 \notin F$, to kładzie się $f_{dok} \langle a_1, s_i \rangle = \hat{a}$, w przeciwnym razie $f_{dok} \langle a_1, s_i \rangle = s_0$.

6. Dla każdej instrukcji $s_0 \ a_1 \rightarrow s_0$ w A^3 kładzie się w RSG $f_{in} \langle a_1, s_i \rangle = \emptyset$, $f_{out} \langle a_1, s_i \rangle = \emptyset$, $f_{kon} \langle a_1, s_i \rangle = S - \{s_0\}$, $f_{dem} \langle a_1, s_i \rangle = \emptyset$; jeżeli $s_0 \notin F$, to $f_{dok} \langle a_1, s_i \rangle = \hat{a}$, w przeciwnym razie $f_{dok} \langle a_1, s_i \rangle = s_0$.

Jak łatwo sprawdzić $L/RSG/ = L/A^3/$.

Tw.8.2. Klasy języków $\mathcal{L}_R$ i $\mathcal{L}/RSG/$ są równoważne.

Dowód bezpośrednio z twierdzenia 8.1 i lematu 8.1.

Tw.8.3. Klasa języków $\mathcal{L}_{RSG}$ jest zamknięta ze względu na operacje podstawiania, sumy, złożenia, domknięcia, przedłużania, iloczynu, różnicy, odbicia zwierciadlanego i uzupełnianie.

Dowód - wynika z twierdzenia 8.2 i zamkniętości klasy $\mathcal{L}_R$ ze względu na wymienione w treści twierdzenia operacje.

8.4. Równoważność klas $\mathcal{L}_{RSG}$ i $\mathcal{L}_{PWPP/KWP/}$

Między modelem systemu KWP i językiem LPWPP/KWP/, a rozszerzonym systemem generacyjnym RSG i językiem L/RSG/ zachodzą analogie, które pozwalają pokazać ciekawe własności systemu KWP i języka LPWPP/KWP/. Własności te zostaną sformułowane w postaci wniosków.

Wn.8.1. Dla systemu KWP istnieje rozszerzony system generacyjny RSG taki, że $LPWPP/KWP/ = L/RSG/$.

Uzasadnienie Niech $ODO_{KWP} = \{\hat{P}, \hat{C}, \hat{TSD}, \hat{Dz}_{KWP}, \hat{f}_{dek}, \hat{f}_{in}, \hat{f}_{out}, \hat{f}_{kon}, \hat{f}_{dem}, \hat{f}_{dok}\}$, gdzie $TSD = TSD_M \cup TSD_U \cup TSD_D$ będzie opisem działania oprogramowania dowolnego systemu KWP. Rozważmy system $RSG = \{V, \Sigma, D, E, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}, f_{dok}\}$, dla którego $V = \hat{C}, \Sigma = \hat{TSD}_M, D = \hat{TSD}, E = \hat{TSD}_D$ i $\hat{a} = \hat{t}\hat{u}$, $Dz_{SG} = \hat{Dz}_{KWP}$ oraz $\hat{f}_{in} = \hat{f}_{in}$, $\hat{f}_{out} = \hat{f}_{out}$, $\hat{f}_{kon} = \hat{f}_{kon}$, $\hat{f}_{dem} = \hat{f}_{dem}$, $\hat{f}_{dok} = \hat{f}_{dok}$. Jak łatwo sprawdzić $L/RSG/ = LPWPP/KWP/$.

Wn.8.2. Dla każdego rozszerzonego systemu generacyjnego RSG istnieje system KWP taki, że $LPWPP/KWP/ = L/RSG/$.

Uzasadnienie Niech $RSG = \{V, \Sigma, D, E, Dz_{SG}, f_{in}, f_{out}, f_{kon}, f_{dem}, f_{dok}\}$. Rozważmy system KWP o opisie działania oprogramowania $\hat{O}DO_{KWP} = \{\hat{P}, \hat{C}, \hat{TSD}, \hat{Dz}_{KWP}, \hat{F}_{dek}, \hat{f}_{in}, \hat{f}_{out}, \hat{f}_{kon}, \hat{f}_{dem}, \hat{f}_{dok}\}$, gdzie $\hat{TSD} = \hat{TSD}_M \cup \hat{TSD}_U \cup \hat{TSD}_D$, dla którego $\hat{C} = V$, $\hat{TSD}_M = \Sigma$, $\hat{TSD}_U = D$, $\hat{TSD}_D = E$ i $\hat{t}\hat{b} = \hat{b}$, $\hat{Dz}_{KWP} = Dz_{SG}$ oraz $\hat{f}_{in} = f_{in}$, $\hat{f}_{out} = f_{out}$, $\hat{f}_{kon} = f_{kon}$, $\hat{f}_{dem} = f_{dem}$, $\hat{f}_{dok} = f_{dok}$.

Oczywiście $LPWPP/KWP/ = L/RSG/$.

Z wniosków 8.1 i 8.2 wynika bezpośrednio,

Wn.8.3. Klasy języków $\mathcal{L}_{PWPP/KWP/}$ i $\mathcal{L}_{RSG}$ są równoważne.

Z twierdzenia 8.1 i wniosku 8.3 wynika -

Wn.8.4. Dla każdego systemu KWP język $LPWPP/KWP/$ jest językiem regularnym.

Z twierdzenia 8.3 i wniosku 8.4 wynika -

Wn.8.5. Klasa języków $\mathcal{L}_{PWPP/KWP/}$ jest zamknięta ze względu na operacje podstawiania, sumy, złożenia, domknięcia, przedłużania, iloczynu, różnicy, odbicia zwierciadlanego i uzupełnianie.

8.5. Algorytm rozwiązania zadania Z4

Automat A^3 uzyskany w wyniku działania algorytmu A1.8.1 zostanie wykorzystany przy rozwiązywaniu zadania Z4 - drugiego zadania koordynacji. Zadanie to polega na znalezieniu opisów wszystkich par ciągów $\langle \hat{x}, \hat{y} \rangle$ $\hat{x} = \hat{v}_1 \dots \hat{v}_i \dots \hat{v}_n$, $\hat{y} = \hat{s}\hat{d}_1 \dots \hat{s}\hat{d}_i \dots \hat{s}\hat{d}_n$ takich, że $\hat{f}_{dok} \langle \hat{v}_n, \hat{Tsd}_n \rangle = \hat{tsd}_{dok}$ i $\langle \hat{x}, \hat{y} \rangle$ jest poprawnym wspomaganym procesem projektowania.

Algorytm rozwiązujący zadanie Z4 składa się z 3 części.

W części pierwszej automat $A^3 = \{\hat{V}, S, F, I, s_0\}$ jest przekształcany w automat $A^3' = \{\hat{V}, S, F', I, s_0\}$, dla którego zbiór F' jest podzbiorem zbioru F , takim, że dla każdego $s_x \in F'$ $s_{x3} = \hat{tsd}_{dok}$.

W części drugiej na urządzenie wyjścia wyprowadzane są /z zaznaczeniem stanów końcowych/ instrukcje i etykiety instrukcji należących do części aktywnej.

W części trzeciej na urządzenie wyjścia wyprowadzane są opisy wszystkich typów struktur danych, których symbole są etykietami instrukcji należących do części aktywnej.

Algorytm ten jest najprostszą formą rozwiązania zadania Z4. Wymaga on jednak aktywnego udziału użytkownika systemu, który analizując wyprowadzone instrukcje może uzyskać interesujące go rozwiązanie.

Oczywiście można wyobrazić sobie algorytmy całkowicie automatyczne /bez udziału użytkownika/ rozwiązujące zadanie Z4. Najbardziej naturalnymi rozwiązaniami wydają się algorytmy, które zamiast zbioru instrukcji wyprowadzałyby wyrażenie regularne, określone przez A^3 , czy diagram stanów A^3 . Rozwiązania takie są jednak znacznie bardziej złożone od algorytmu przedstawionego w pracy.

Algorytm A1.8.2

Rozwiązanie zadania Z4

Dane: Automat $A^3 = \{\hat{V}, S, F, I, s_0\}$ taki, że $L/A^3/ = LPWP/KWP/$, typ struktury danych $\hat{tsd}_{dok} \in TSD_D$, zbiór opisów typów struktur danych użytkownika.

Wyniki: Opis poprawnych wspomaganych procesów projektowania dostarczających dokumentację typu $\hat{tsd}_{dok}$. Opis procesów uzyskuje się poprzez analizę instrukcji odpowiedniego automatu.

Oznaczenia:

- $I_j = s_j^1 \hat{v}_j \rightarrow s_j^2$ - j-ta instrukcja części aktywnej automatu A^3 ,
- automat $/j_i, I', \Delta I', jt, \hat{TSD}_U/$ - procedura przekształcająca automat A^3 w automat A^3 . Parametrami wyjściowymi procedury są:
 - j_i - liczba instrukcji części aktywnej automatu A^3
 - I' - część aktywna automatu A^3 $I' = \{I_1, \dots, I_{j_i}\}$
 - $\Delta I'$ - etykiety instrukcji należących do części aktywnej automatu A^3 $\Delta I' = \{\delta/I_1/, \dots, \delta/I_{j_i}/\}$
 - jt - liczba różnych etykiet części aktywnej A^3
 - $\hat{TSD}_U$ - zbiór etykiet części aktywnej A^3 $\hat{TSD}_U = \{\hat{tsd}_1, \dots, \hat{tsd}_{jt}\}$

- instrukcja $/I_j/$ - procedura wyprowadzająca na urządzenie zewnętrzne, instrukcję $I_j \in I'$ i jej etykietę $\delta/I_j/$ oraz zaznaczająca stany końcowe A^3 ,
- opis warunków $/tsd_i/$ - procedura, która wyprowadza opis warunków, które musi spełniać struktura typu $tsd_i \in TSD_U$.

begin

automat $/ji, I', \Delta I', jt, TSD_U/;$

if $ji = 0$ then writeln $/$ 'rozwiązanie nie istnieje'/

else

begin

, for $j=1$ to ji do instrukcja $/I_j/;$

for $i=1$ to jt do opiswarunków $/tsd_i/$

end

end.

Uwagi o efektywności algorytmu A1.8.2

Czas wykonania algorytmu zależy od parametrów automatów A^3 i A^3 oraz parametrów zbioru typów struktur danych, których symbole są etykietami części aktywnej automatu A^3 . Z powodu braku miejsca nie zostanie podane oszacowanie czasu wykonania algorytmu. Ponieważ operacje przez niego wykonywane są proste, należy spodziewać się, że jest on algorytmem "stosunkowo" szybkim.

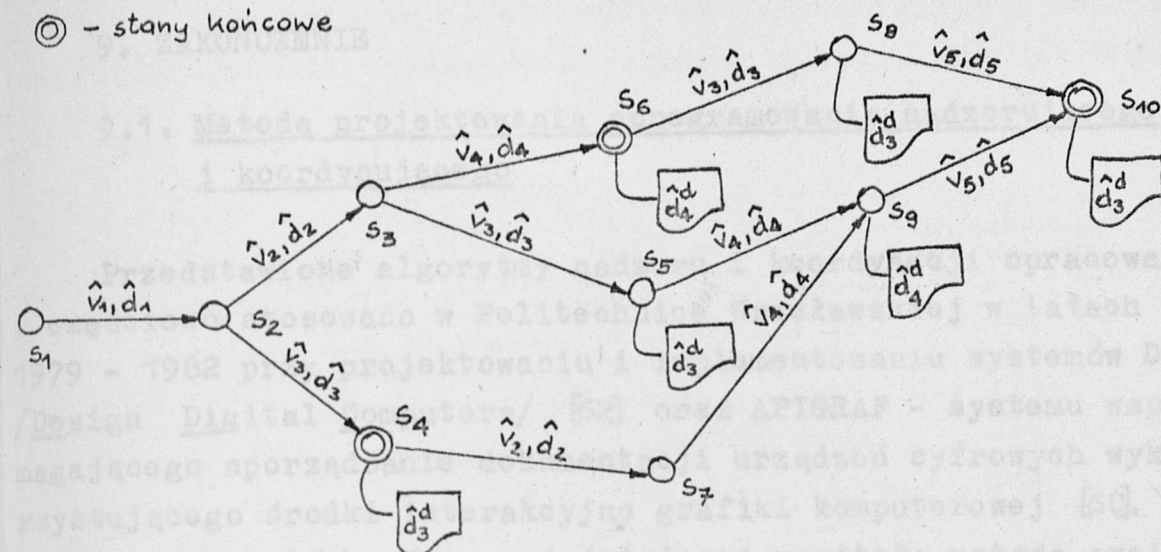
Przykład 8.1

Rozważmy system KWP o opisie działania oprogramowania $\hat{O}DO_{KWP}$ przedstawionym w tabeli 6.1. Na rysunku 8.1 przedstawiono diagram stanów automatu A^3 akceptującego język LPWPP/KWP/ tego systemu /na rysunku wyraźnie zaznaczono typy dokumentacji generowanej przez system/.

Z rysunku wynika, że proces $\langle \hat{v}_1, \hat{s}d_1 \rangle \langle \hat{v}_2, \hat{s}d_2 \rangle \langle \hat{v}_4, \hat{s}d_4 \rangle \langle \hat{v}_3, \hat{s}d_3 \rangle \langle \hat{v}_5, \hat{s}d_5 \rangle$ jest poprawnym wspomaganym procesem projektowania, natomiast proces $\langle \hat{v}_1, \hat{s}d_1 \rangle \langle \hat{v}_3, \hat{s}d_3 \rangle \langle \hat{v}_2, \hat{s}d_2 \rangle$ nim nie jest.

Założmy, że użytkownik systemu jako cel projektowania stawia sobie uzyskanie dokumentacji typów a/ $\hat{d}_4^d$, b/ $\hat{d}_5^d$.

⊙ - stany końcowe



$$\begin{aligned}
 s_1 &= \langle \phi, \phi, t\hat{a} \rangle, \quad s_2 = \langle \{\hat{A}\}, \{\hat{A}\}, t\hat{a} \rangle, \quad s_3 = \langle \{\hat{A}, \hat{B}\}, \{\hat{B}\}, t\hat{a} \rangle, \\
 s_4 &= \langle \{\hat{A}, \hat{C}\}, \{\hat{C}\}, \hat{d}_3^d \rangle, \quad s_5 = \langle \{\hat{A}, \hat{B}, \hat{C}\}, \{\hat{B}, \hat{C}\}, \hat{d}_3^d \rangle, \\
 s_6 &= \langle \{\hat{A}, \hat{D}\}, \{\hat{B}\}, \hat{d}_4^d \rangle, \quad s_7 = \langle \{\hat{A}, \hat{B}, \hat{C}\}, \{\hat{B}, \hat{C}\}, t\hat{a} \rangle, \\
 s_8 &= \langle \{\hat{A}, \hat{C}, \hat{D}\}, \{\hat{C}, \hat{D}\}, \hat{d}_3^d \rangle, \quad s_9 = \langle \{\hat{A}, \hat{B}, \hat{C}\}, \{\hat{B}, \hat{C}\}, \hat{d}_4^d \rangle, \\
 s_{10} &= \langle \{\hat{A}, \hat{B}, \hat{C}, \hat{D}\}, \{\hat{E}\}, \hat{d}_5^d \rangle,
 \end{aligned}$$

$$F = \{s_4, s_6, s_{10}\}$$

Rys.8.1. Diagram stanów automatu A^3 akceptującego język LPWPP systemu KWP o opisie działania oprogramowania z tabeli 6.1

W przypadku "a" algorytm A1.8.2 rozwiązujący zadanie Z4 wyprowadzi na urządzenie wyjścia następujące instrukcje i etykiety: $s_1 \hat{v}_1 \rightarrow s_2, \hat{d}_1$; $s_2 \hat{v}_2 \rightarrow s_3, \hat{d}_2$; $s_3 \hat{v}_4 \rightarrow \underline{s_6}, \hat{d}_4$ oraz opisy typów struktur $\hat{d}_1, \hat{d}_2, \hat{d}_4$. Podkreślenie stanu s_6 oznacza, że jest on stanem końcowym.

W przypadku "b" algorytm wyprowadzi wszystkie instrukcje i etykiety instrukcji należących do części aktywnej automatu A^3 /w przykładzie 8.1 $A^3 = A^3$ / oraz opisy typów struktur $\hat{d}_1, \hat{d}_3, \hat{d}_4, \hat{d}_5$.

9. ZAKOŃCZENIE

9.1. Metoda projektowania oprogramowania nadzorującego i koordynującego

Przedstawione algorytmy nadzoru i koordynacji opracowano i częściowo stosowano w Politechnice Wrocławskiej w latach 1979 - 1982 przy projektowaniu i implementowaniu systemów DEDICO /Design Digital Computers/ [62] oraz APIGRAF - systemu wspomagającego sporządzanie dokumentacji urządzeń cyfrowych wykorzystującego środki interakcyjne grafiki komputerowej [60]. W wyniku zdobytych wówczas doświadczeń powstała metoda projektowania nadzorującego i koordynującego. Na rysunku 1.1 przedstawiono kolejne kroki tej metody.

W systemie DEDICO automat A^1 wykorzystany jest do modelowania działania systemu oraz rozwiązywania pierwszego zadania nadzoru. W systemie APIGRAF automat ten służy do rozwiązywania pierwszego zadania koordynacji.

Automaty A^2 i A^3 były wykorzystywane przy projektowaniu oraz analizowaniu i dokumentowaniu działania systemu.

Praktyka wykazała, że w przypadku powrotów do etapów 1,2,3 tylko w ostateczności powtarza się je. Zwykle analizuje się poprawność danego etapu, niekiedy zaś zmienia się istniejące już rozwiązania [33].

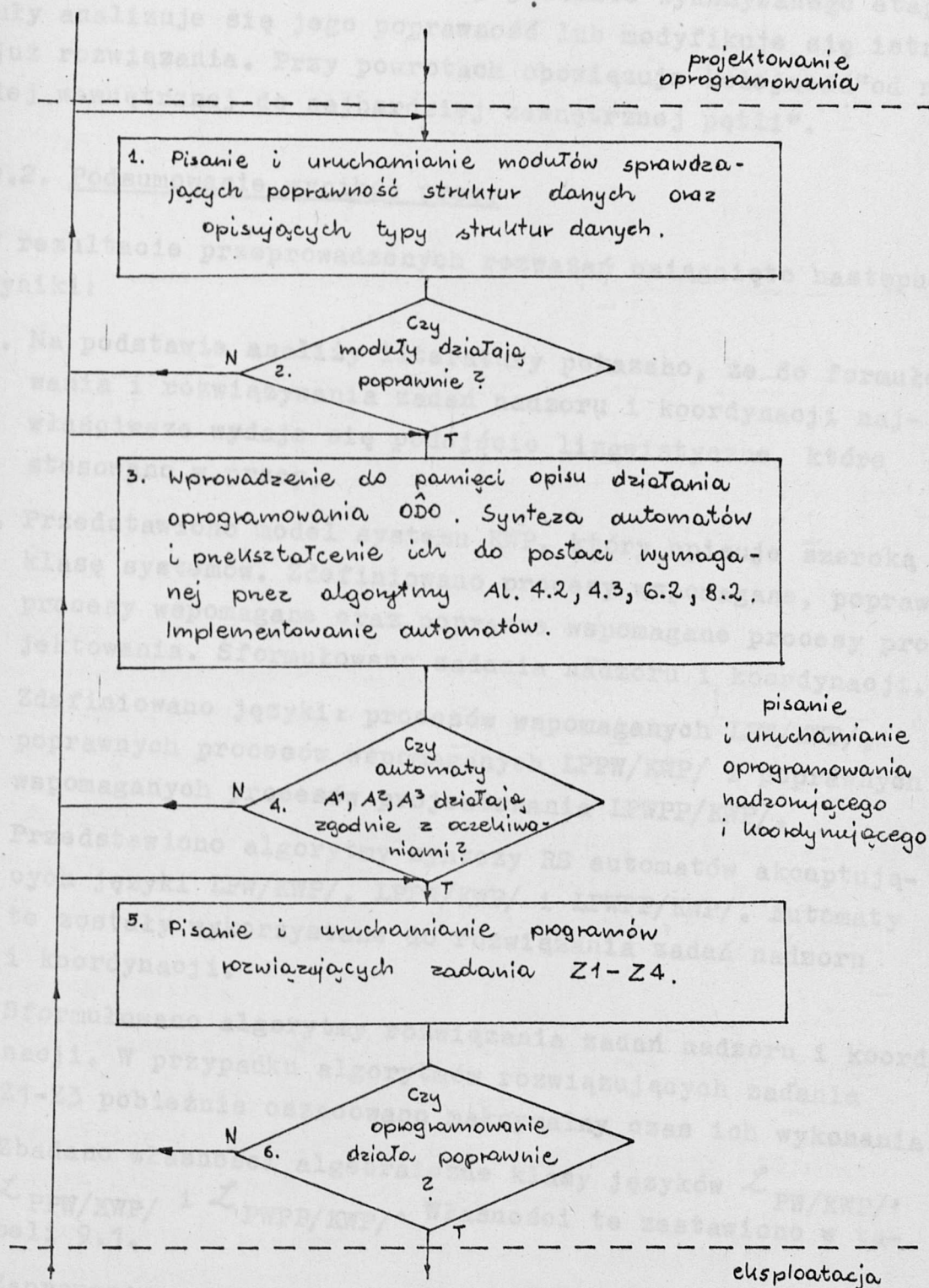
W szczególności dla etapów:

- pierwszego - do /lub z/ zbioru działań F lub procesów PR, które mają być wspomagane dodaje się /lub usuwa/ określone działania czy procesy,
- drugiego - do /lub z/ zbioru programów P dodaje się /lub usuwa/ pewne programy lub łączy się programy tworząc programy wielowariantowe,
- trzeciego - zmienia się opis działania pojedynczych programów, zmienia się typy struktur danych - często dokonuje się ich dekompozycji i ponownej syntezy, modyfikuje się projekty baz danych.

Przy wykonywaniu pętli obowiązuje kolejność "od najbardziej wewnętrznej do najbardziej zewnętrznej".

Doświadczenia zdobyte przy implementowaniu systemów DEDICO i APIGRAF mogą stanowić punkt wyjścia do zaproponowania metody

pisania i uruchamiania oprogramowania nadzorującego i koordynującego korzystanie z zasobów użytkowych systemu KWP /rys.9.1 /



Rys.9.1. Schemat metody pisania i uruchamiania oprogramowania nadzorującego i koordynującego systemu KWP

Należy zauważyć, że w przypadku rzetelnie wykonanego makroetapu projektowania tego oprogramowania, zaznaczone na rysunku powroty do odpowiednich etapów, wykonywane są rzadko. Podobnie jak dla projektowania, w razie powrotu do poprzednio wykonywanego etapu, z reguły analizuje się jego poprawność lub modyfikuje się istniejące już rozwiązania. Przy powrotach obowiązuje kolejność "od najbardziej wewnętrznej do najbardziej zewnętrznej pętli".

9.2. Podsumowanie wyników pracy

W rezultacie przeprowadzonych rozważań osiągnięto następujące wyniki:

1. Na podstawie analizy literatury pokazano, że do formułowania i rozwiązywania zadań nadzoru i koordynacji właściwsze wydaje się podejście lingwistyczne, które stosowano w pracy.
2. Przedstawiono model systemu KWP, który opisuje szeroką klasę systemów. Zdefiniowano procesy wspomagane, poprawne procesy wspomagane oraz poprawne wspomagane procesy projektowania. Sformułowano zadania nadzoru i koordynacji.
3. Zdefiniowano języki: procesów wspomaganych LPW/KWP/, poprawnych procesów wspomaganych LPPW/KWP/ i poprawnych wspomaganych procesów projektowania LPWPP/KWP/.
4. Przedstawiono algorytmy syntezy RS automatów akceptujących języki LPW/KWP/, LPPW/KWP/ i LPWPP/KWP/. Automaty te zostały wykorzystane do rozwiązania zadań nadzoru i koordynacji.
5. Sformułowano algorytmy rozwiązania zadań nadzoru i koordynacji. W przypadku algorytmów rozwiązujących zadania Z1-Z3 pobieżnie oszacowano maksymalny czas ich wykonania.
6. Zbadano własności algebraiczne klasy języków $\mathcal{L}_{PW/KWP/}$, $\mathcal{L}_{PPW/KWP/}$ i $\mathcal{L}_{PWPP/KWP/}$. Własności te zestawiono w tabeli 9.1.
7. Zaproponowano metodę projektowania oprogramowania nadzorującego i koordynującego korzystanie z zasobów programowych systemu KWP.

Wydaje się, że osiągnięte wyniki w wystarczającym stopniu dowodzą tezy pracy.

Tab.9.1.

Zestawienie własności algebraicznych klas języków

$\mathcal{L}_{PW/KWP/}$, $\mathcal{L}_{PPW/KWP/}$, $\mathcal{L}_{PWPP/KWP/}$, $\mathcal{L}_1/SG/$,
 $\mathcal{L}_2/SG/$, $\mathcal{L}_{RSG}$, $\mathcal{L}_{R1}$, $\mathcal{L}_{R2}$, $\mathcal{L}_R$

operacja Klasa języków	podstawianie	suma	złożenie	domknięcie	przedłużanie	różnica	iloczyn	odbicie zwierciadlane	uzupełnianie
$\mathcal{L}_{PW/KWP/}$, $\mathcal{L}_1/SG/$, $\mathcal{L}_{R1}$	+	+	-	+	+	-	+	-	-
$\mathcal{L}_{PPW/KWP/}$, $\mathcal{L}_2/SG/$, $\mathcal{L}_{R2}$	+	+	+	+	+	-	+	-	-
$\mathcal{L}_{PWPP/KWP/}$, $\mathcal{L}_{RSG}$, $\mathcal{L}_R$	+	+	+	+	+	+	+	+	+

Oznaczenia:

- + - klasa jest zamknięta ze względu na operację
- - klasa nie jest zamknięta ze względu na operację

Za dodatkowe rezultaty pracy można uznać zdefiniowanie oraz zbadanie abstrakcyjnych systemów, równoważnych pod względem akceptowania i generowania języków z prezentowanym modelem systemu KWP.

W szczególności:

1. Zdefiniowano system generacyjny SG oraz języki $\mathcal{L}_1/SG/$ i $\mathcal{L}_2/SG/$ przez niego definiowane.
2. Zdefiniowano rozszerzony system generacyjny RSG oraz język $\mathcal{L}/RSG/$ przez niego generowany.
3. Przedstawiono algorytmy syntezy RS automatów A^1 , A^2 , A^3 akceptujących języki $\mathcal{L}_1/SG/$, $\mathcal{L}_2/SG/$ i $\mathcal{L}/RSG/$.
4. Zbadano własności algebraiczne klas języków $\mathcal{L}_1/SG/$, $\mathcal{L}_2/SG/$, $\mathcal{L}_{RSG}$.

5. Zdefiniowano RS automaty typu 1 i typu 2 oraz zbadano własności algebraiczne klas języków $\mathcal{L}_{R1}$ i $\mathcal{L}_{R2}$ przez nie akceptowanych /tab.9.1/.

Uzyskane wyniki mogą znaleźć zastosowanie przy projektowaniu systemów komputerowych podobnych do systemu KWP. Przykładowo wydaje się, że system generacyjny i rozszerzony system generacyjny oraz algorytmy syntezy automatów A^1 , A^2 , A^3 mogą być przydatne przy projektowaniu dowolnej biblioteki programów komunikujących się poprzez wspólną bazę danych, ze szczególnym uwzględnieniem nadzorowania i koordynowania korzystania z tej biblioteki.

9.3. Kierunki dalszych badań

Rozpatrywane w pracy: model systemu KWP, zadania, algorytmy nadzoru i koordynacji opisane są w sposób ogólny i dzięki temu mogą służyć jako punkt wyjścia do dalszych badań szczegółowych. W związku z tym poniżej przedstawiono kierunki badań, które można prowadzić wykorzystując formalizmy i pomysły zamieszczone w pracy.

1. Badania struktur danych systemów KWP.

Badania takie powinny obejmować metody syntezy i analizy języków opisu danych oraz formalne metody wnioskowania o ich poprawności /badanie specjalnych rachunków predykatów/. Problemy te są szczególnie istotne dla dokładnego sformułowania i analizy złożoności algorytmów rozwiązań zadań Z1 - Z4.

2. Badania - bardziej wyrafinowanych niż regularne - języków zarządzania zasobami systemu.

3. Opracowanie metod czytelnego wyprowadzania rozwiązania zadania Z4.

Wydaje się, że idealnym rozwiązaniem powinno być wyprowadzenie diagramu stanów.

4. Sformułowanie i rozwiązanie bardziej złożonych niż przedstawione w pracy zadań nadzoru i koordynacji. Jak łatwo zauważyć duża liczba rozwiązań praktycznych

10. BIBLIOGRAFIA

- [1] Allan J.J./ed/: CAD Systems. Proceeding of the IFIP Working Conference on Computer-Aided Design Systems. North-Holland, Amsterdam - New York - Oxford 1977.
- [2] Automatyzacja prac w projektowaniu. Mat. IV Krajowej Konferencji Naukowo-Technicznej n.t. Komputerowe Wspomaganie Projektowania. Tom II. Poznań 1981.
- [3] Babiński C.: Elementy nauki o projektowaniu. WNT, Warszawa 1972.
- [4] Babiński C., Wolpe M.: Automatyzacja projektowania. Arkady, Warszawa 1976.
- [5] Benc Cz., Dawidowski J.: Komputerowo wspomagane generowanie dokumentacji projektowej /w projektowaniu dowolnej dziedziny projektowej/. W [2] ; s.181-196.
- [6] Blikle A.: Automaty i gramatyki. PWN, Warszawa 1971.
- [7] Brinch - Hansen P.: Podstawy systemów operacyjnych. WNT, Warszawa 1976.
- [8] Bromirski J.: Teoria automatów. WNT, Warszawa 1969.
- [9] Buchole K., Tyszer J.: Automatyczne generowanie schematów układów cyfrowych jako przykład wspomaganego komputerem tworzenia dokumentacji projektowej. W [2] ; s.89-102.
- [10] Butlin G.A.: Techniques for processing interactions in FORTRAN. W [1] ; p.35-60.
- [11] CAD in Medium Sized and Small Industries. First European Conference on Computer Aided Design in Medium Sized and Small Industries MICAD 80, Paris, France, 23-26 Sept.1980 /Amsterdam, Netherlands: North-Holland 1981/.
- [12] Case P.W., Correira M., Ginopulos W., Heller W.R., Ofek H., Raymond T.C., Simek R.L., Stieglitz C.R.: Design automation in IBM. W ; IBM J.Res.& Dev. /USA/, vol.25/5/, p.631-46, Sept.1981.
- [13] Coyle F.E.: Optical design using interactive graphics. W [11] ; p.609-19.

- [14] Dahl O.J., Dijkstra E.W., Hoare C.S.R.: Structured programming, Academic Press, London and New York, 1972.
- [15] Dąbrowski Z., Kierzkowski Z.: Inwariantne oprogramowanie współczesnych systemów komputerowego wspomaganie projektowania. W [44]; s.161-166.
- [16] Dąbrowski Z., Kierzkowski Z.: Koncepcja struktury systemu KWP. W [34]; s.19-32 /mat.pokonf., t.II/.
- [17] Dawidowski J., Owczarczak P., Wiśniewski M.: Automatyczne generowanie dokumentacji projektowej. W [2]; s.103-116.
- [18] Dietrych J.: Projektowanie i konstruowanie. WNT, Warszawa 1974.
- [19] Dietrych J.: System i Konstrukcja. WNT, Warszawa 1978.
- [20] Dijkstra E.W.: Umiejętność programowania. WNT, Warszawa 1978.
- [21] Fulton R.E.: CAD-CAM Approach to improving industry productivity gathers momentum. Astro-Aeoron 20/2/, p.64-70, 82.
- [22] Everheim W. i inni: Application of suitable data structure for CAD in the machine tool industry. W [1]; p.327-348.
- [23] Gero J.S., Julien W.G.: High level non-graphical interaction in computer-aided design. W [1]; p.241-267.
- [24] Goliński J.: Metody optymalizacyjne w projektowaniu technicznym. WNT, Warszawa 1974.
- [25] Goliński J.: Adaptacyjne systemy poszukiwania konstrukcji optymalnej. PWN, Warszawa 1976.
- [26] Kaposi A.A., Popowie L.M., Rzevski G.: An algebra of algorithms and its application to program design and maintenance. Computer Aided Design 10 /1/, p.74-77, Jan.1978.
- [27] Kaposi A.A.: An engineer's guide to algorithmic structures. Computer Aided Design 9/1/, p.58-65, Jan.1977.
- [28] Ketil B.: A text analyzer and text generator for interactive command languages in CAD. W [1]; p.203-215.
- [29] Kierzkowski Z.: Elementy informatyki /technika, metody, zastosowania/. PWN, Warszawa-Poznań 1976.
- [30] Kierzkowski Z.: Modele i synteza oprogramowania systemów Komputerowego wspomaganie projektowania. Wyd.Pol.Pozn., seria: rozprawy, nr 122. Poznań 1980.

- [31] Kierzkowski Z.: Problemy syntezy oprogramowania systemów komputerowego wspomaganie działań projektowych. W [54]; s.66-110.
- [32] Kierzkowski Z., Rogala M.: Język komend w systemie komputerowego wspomaganie projektowania. W [42]; cz.II, s.181-186.
- [33] Kołodziej R.: Zagadnienia doboru struktury programowej systemu komputerowego wspomaganie projektowania. W [42]; cz.II, s.187-196.
- [34] Komputerowe wspomaganie projektowania. III Konferencja: Automatyzacja prac w projektowaniu - Zbiór referatów cz.1-3, mat.pokonf.t.I-II.SIMP-Poznań, Środowiskowy Ośrodek Inform. Pol.Pozn., Pracownia Metodologii Projekt.Inst. Filozofii i Socjologii PAN. Poznań 1978.
- [35] Krause F.L., Vassilakopoulos V.: A way to computer supported systems for integrated design and production process planning. W [1]; p.5-39.
- [36] Latombe J.C.: Artificial intelligence in CAD: the "TROPIC" system. W [1]; p.61-121.
- [37] Latombe J.C. /ed./: Artificial intelligence and pattern recognition in Computer Aided Design. Proceedings of the IFIP Working Conference /Grenoble, France, March 17-19, 1978/. North-Holland, Amsterdam - New York - Oxford 1978.
- [38] Manna Z.: Mathematical theory of computation. McGraw Hill 1974.
- [39] Masse P., Coulomb J.L., Ancelle B.: System-design methodology in CAD. Programs based on the finite-element meth. IEEE Magnet. 18/2/, p.609-616, 82.
- [40] Metodologia i komputerowe wspomaganie projektowania - Metody i środki automatyzacji prac projektowych. Prace nauk.Inst. Cybern.PWr., nr 39, seria: konferencje, nr 11, Wrocław 1976.
- [41] Myers W.: CAD-CAM - the need for a broader focus. Computer 15/1/, p.105-117, 82.
- [42] Nowoczesne metody projektowania. Kraj.Konferencja - Materiały cz.I-II.SEP Wrocław - MERA - ELWRO - PTC - Wrocław.Walbrzych 1980.

- [43] Pendo M.H.: SARA as a tool for software design: building-block modeling and composition. W : Proceedings of the NBS/IEEE/ACM Software Tool Fair /NBS-SP-500-80/, San Diego, CA, USA, 10-12 March 1981, p.76-82.
- [44] Projektowanie III. Badania, kształcenie, praktyka, Materiały III Ogólnopolskiej Konf.Nauk. w zakresie teorii i praktyki projektowania /Komitet Naukoznawstwa PAN i ICT Pol.Wrocł./ . Prace nauk.Inst.Cybern.PWr., nr 54, seria:konferencje nr 19, t.1-2,Wrocław 1978.
- [45] Pszczółowski T.: Mała encyklopedia prakseologii i teorii organizacji. Wyd.Ossolineum, Wrocław-Warszawa-Kraków-Gdańsk 1978.
- [46] Roos D.: ICES system design.MIT 1967.
- [47] Sielicki A.: Informatyczne systemy nauczania: wymagania metody syntezy. Mat.Konf.Infogryf 78: Informatyka w dydaktyce, Kołobrzeg 1978.
- [48] Sielicki A., Jeleniewski T.: Elementy metodologii projektowania technicznego. WNT, Warszawa 1980.
- [49] SolarSKI E.: Kodowanie konturów graficznych przy pomocy języka proceduralnego. W [2]; s.117-128.
- [50] Staniszewski R.: Rozwój systemów projektowania. Wyd.Ossolineum, seria: najnowsze osiągnięcia nauki, Wrocław 1981.
- [51] Subieta K.: Zarys teorii baz danych - podejście lingwistyczne. PWN, Warszawa - Łódź 1981.
- [52] Tolmann F., Westermann R.: A pilot version of a building code independent CAD-system. W [11]; p.477-86.
- [53] Turski W.M.: Metodologia programowania. WNT,Warszawa 1978.
- [54] Teoretyczne problemy projektowania. Mat.I Sesji Szkoły Metodologii Projektowania. Prace nauk.Inst.Cybern.PWr., nr 56, seria: konferencje nr 20, Wrocław 1979.
- [55] Vlietstra J, Wielinga R.F./ed./: Computer-Aided Design. Proceedings of the IFIP Working Conference on Principles of Computer - Aided Design /Eindhoven, The Netherlands, October 16-18,1972/. North-Holland, Amsterdam-London 1973.

- [56] Warenen E.A., The possibility for the automatic protection of command language. W [1] ; p.219-241.
- [57] Warman E.A.: Computer Aided Design: on intersection of ideas. W [37] ; p.1-18.
- [58] Wolpe M.: Zintegrowane systemy automatyzacji projektowania SAPRO, ICES, GENESYS. Arkady, Warszawa 1977.
- [59] Zahradnik G.J., Ragsdell K.M., Lindenloub J.C., Cipra R.J.: The design, implementation, and testing of CAI/CMI/CAD environment for machine design. W : Frontiers in Education Eleventh Annual Conference, Rapid City, SD, USA, 19-21 Oct. 1981, p.332-9.

Prace niepublikowane

- [60] APIGRAF - System wspomagający sporządzanie i aktualizację dokumentacji logicznej urządzeń cyfrowych, wykorzystujący środki interakcyjne grafiki komputerowej. Raport Instytutu Komputerowych Systemów Automatyki i Pomiarów. Wrocław 1981. Kołodziej R. i inni.
- [61] Grocholski L.: Przekształcenia optymalizujące program działań w procesie projektowania. Inst.Cybern.PWr., raport nr 37/79.
- [62] Podsystem projektowania konstrukcji cyfrowych. Raport Inst.Cybern.PWr., nr 40/80. Raport powstał w wyniku realizacji problemu węzłowego pt. Rozwój komputeryzacji systemów automatyki i pomiarów, nr 06.1. Grupa tematyczna 2 - System dydaktyczny projektowania wspomagane komputerowo. Sielicki A., Chałon M., Jacewicz R., Kołodziej R., Chęciński M., Czerwiński A., Grocholski L., Sibilski W..

LINGUISTIC COORDINATION AND SUPERVISION ALGORITHMS IN COMPUTER AIDED DESIGN

This paper is a proposal of the application of the mathematical linguistic theory for defining and studying the computational task descriptive languages in computer-aided design /CAD/ systems. An application of the formal-linguistic convention in solving the coordination and supervision tasks was treated in a particular way. The supervision task is understood as a comparison of computational task descriptions with the admissible descriptions; the purpose of the coordination task is to generate /with the purpose of facilitating exercise of resources/ admissible descriptions.

The purpose of this paper is to work out the concepts and formal bases for synthesising software, coordinating and supervising the use of utilitarian CAD system software.

The dissertation thesis is the following:

The represented models of: the utilitarian programs library, CAD system functioning and the designer-computer dialog enable the working out of affective coordination and supervision algorithms for use in utilitarian CAD system software.

Employing the formal - linguistic convention the computational task description is a sequence of symbols and CAD system functioning is modeled by Rabin's - Scott's automata. The CAD system model, represented in this paper enables the defining of the aided process language and the derivative languages and also formulating synthesis algorithms for Rabin's - Scott's automata which accept these languages.

The coordination and supervision task solutions reduce to solving analysis and synthesis problems of appropriate languages for which the automata accepting them are used.

Systems equivalent to the CAD system model in respect to defining languages are also discussed in the paper. Automata and sequence generating systems which accept and generate the aided process language and the derivative languages are defined and their fundamental properties are examined. This discussion makes it possible to examine the algebraic properties of

the aided process language class and the derivative language classes. The proven properties allow the claim that represented algorithms can be applied in systems undergoing changes.

Working out the design method of coordinating and supervising software is the merit of this dissertation. The method is a version /in regard to coordinating nad supervising software/ of Kierzkowski's method of designing computer-aided design system software.

1. Biblioteka Główna PWR.	1 egz.
2. Biblioteka Międzyinstytutowa, CIBT	3 egz.
3. Dyrektor	1 egz.
4. Asasent	2 egz.
5. Asasent Dyrektora i Asasent Izolowania Kadry	1 egz.
6. Autor	2 egz.
7. Asasent	3 egz.