Raport Badawczy

Research Report

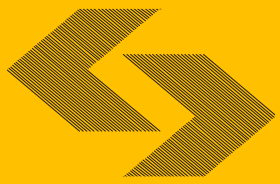
RB/52/2010

Ciągowe struktury symboliczne

K. Bartyzel

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences



CIĄGOWE STRUKTURY SYMBOLICZNE

Krzysztof Bartyzel

Studia Doktoranckie IBS PAN

Katolicki Uniwersytet Lubelski Jana Pawła II

Praca poświęcona została analizie głównych ciągowych struktur symbolicznych. Przedstawiono przegląd aktualnie rozwijanych metod, opisując bardziej szczegółowo algorytmy reprezentujące najważniejsze podejścia. Struktury symboliczne pozwalają na tworzenie systemów umożliwiających automatyzację procesu klasyfikacji, generalizacji oraz rozumienia obrazów cyfrowych. Największą ich zaletą jest brak ograniczeń co do zbioru oczekiwanych rozwiązań.

The paper contains the analysis of the main chains symbolic structures. The survey of the methods being developed is presented and the algorithms representing the most important approaches are described in more detail. Symbolic structure allows for the creation of systems capable of automating the process of classification, generalization and understanding of digital images. The biggest advantage is their lack of constraints on a set of desired solutions.

Key words: *chain code, symbolic structure, linguistic description*

Wprowadzenie

Bardzo ważnym aspektem automatyzacji rozumienia obrazów jest połączenie metodologii i matematycznego języka opisującego obraz. Powodem, dla którego często wybierany jest lingwistyczny opis obrazów jest fakt nieznajomości *a priori* klas i wzorców, jakie mogą zostać rozpoznane. w związku z tym, że liczba kategorii obiektów może zbliżać się do nieskończoności, ponieważ nie narzucamy jeszcze żadnych ograniczeń na zbiór oczekiwanych rozwiązań, potrzebne jest narzędzie umożliwiające wygenerowanie potencjalnie nieskończenie wielu kategorii. Każdy mechanizm generujący skończony zbiór elementów jest w tym przypadku nieprzydatny. Dzięki nieskończonej liczbie kategorii będzie możliwa klasyfikacja dowolnego obiektu na podstawie wektora cech opisującego dany obiekt do jednej ze zdefiniowanych kategorii.

Kolejnym powodem użycia metod lingwistycznych jest fakt, że po wstępnym przetworzeniu obrazu otrzymamy opis zawartości obrazu wolny od jakiejkolwiek klasyfikacji znanej *a priori*. Jest to możliwe, ponieważ w proces kon-

strukcji opisu lingwistycznego wbudowany jest bardzo ogólny mechanizm generalizacji. Technologia matematycznej lingwistycznej generalizacji jest polecana jako najlepsza metoda generalizacji[1].

Podstawową ideą wykorzystania struktur symbolicznych jest odwzorowanie obiektu lub modelu klasy obiektów. Najprostszą strukturą symboliczną jest ciąg, nazywany również łańcuchem. Ciąg składa się z pojedynczych prymitywnych elementów, będących najbardziej podstawowymi, niepodzielnymi częściami obrazu. Te niepodzielne elementy postrzegane są, jako całość lub część większego obiektu ale nigdy jako składające się z mniejszych części. Model obiektu tworzony jest poprzez łączone są ze sobą, poprzez konkatenację, pojedynczych prymitywnych elementów. Oprócz struktur ciągowych wyróżnia się również struktury drzewiaste, grafowe, relacyjne. Zostały one wprowadzone, ponieważ struktury ciągowe, mimo niewątpliwych zalet (np. łatwość generowania i porównywania), mają dość mocno ograniczone możliwości odwzorowywania obiektów 2 i 3-wymiarowych.

Dysponując strukturą nieznanego obiektu możemy go porównać z prototypami różnych klas w celu określenia ich podobieństwa. w rzeczywistym świecie nigdy nie uzyskamy idealnego podobieństwa, dwie porównywane struktury zawsze będą się różniły. Celem porównania struktur ma być przybliżone porównanie struktur, określenie kosztu transformacji jednej struktury w drugą. Koszt transformacji może być traktowany jak miara podobieństwa i jest nazywany odległością strukturalną.

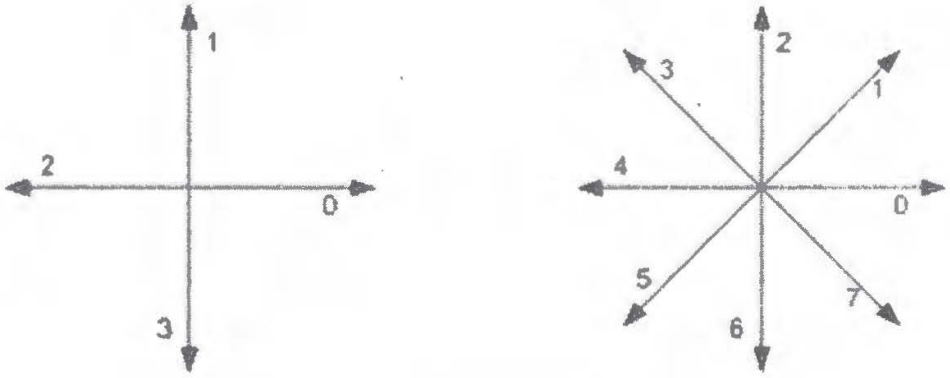
1. Kod łańcuchowy Freemana

Pierwsze próby reprezentacji cyfrowej krzywej przy pomocy kodów łańcuchowych zostały poczynione przez H. Freemana w 1961 roku [2]. Dalsze rozważania można również znaleźć w jego kolejnej pracy [3].

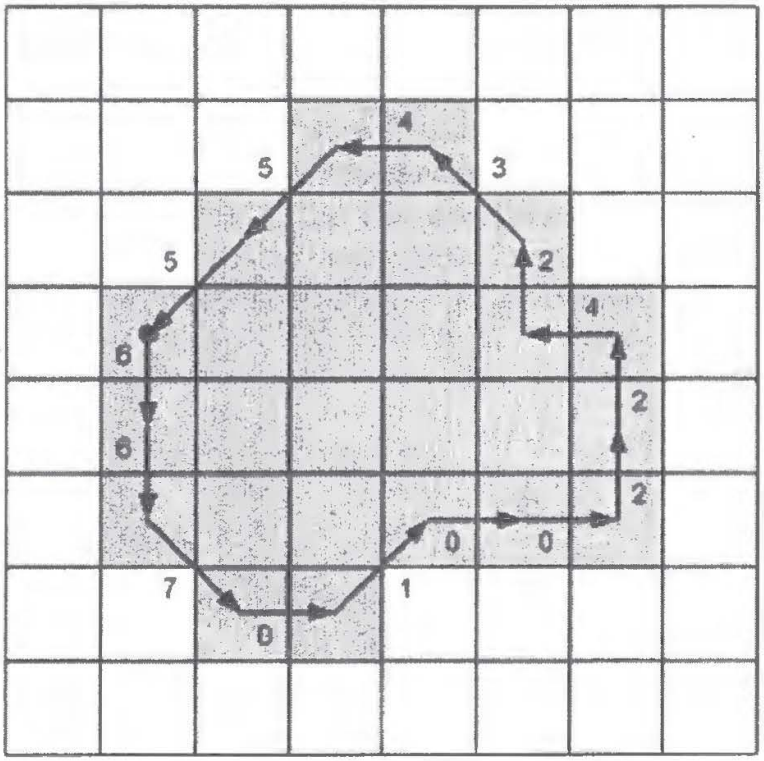
Zadaniem utworzonej przez niego metody była reprezentacja brzegu analizowanego obiektu w obrazie binarnym. Kod opisywał wzajemne położenie kolejnych elementów składających się na brzeg obiektu. Zaproponowane zostały tablice opisujące 4-kierunkowy oraz 8-kierunkowy kod Freemana (rys 1). Jeśli np. kolejny element brzegu obiektu znajduje się na prawo od aktualnie analizowanego, to taką relację opisujemy cyfrą 0. Postępując w analogiczny sposób możemy utworzyć opis całego brzegu. Przekształcanie to można wykonać albo podążając wzdłuż obiektu zgodnie z ruchem wskazówek zegara lub przeciwnie. Zazwyczaj wybierany jest kierunek przeciwny do ruchu wskazówek zegara.

Na rysunku 2 umieszczono przykładową figurę i wyznaczono przebieg kodu łańcuchowego przez brzeg obiektu. Początek łańcucha oznaczono czarnym kółkiem. Dla przedstawionego obiektu odpowiadający mu 8-kierunkowy kod łańcuchowy ma postać:

66701002242345**5**



Rysunek 1: 4- i 8- kierunkowy kod Freemana



Rysunek 2: Kod Freemana - przykład opisu brzegu obiektu

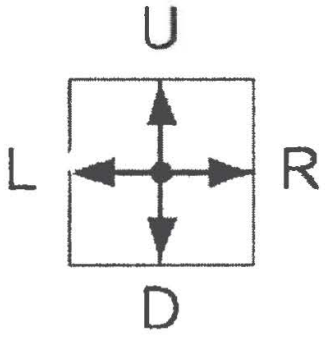
Oczywiście, jeśli generacja kodu rozpoczęta by była z innego punktu lub gdyby kod był generowany w przeciwnym kierunku to otrzymany kod łańcuchowy mógłby znacząco różnić się od przedstawionego.

Jak wspomniano, H. Freeman był autorem pierwszego kodu łańcuchowego. Na podstawie jego pracy powstało wiele artykułów rozszerzających i modyfikujących jego dzieło. J. Bons i A. Kegel [4] wprowadzili różnicowy kod łańcuchowy (ang. DCC - *differential chain code*). Udoskonalenie tej metody zostało zaprezentowane przez Y. T. Hwanga w [5]. Y. Liu and B. Zalik [6] przedstawili metodę bazującą na 8-kierunkowym kodzie Freemana, gdzie elementy łańcucha były kodowe jako względny kąt pomiędzy bieżącym a poprzednim kierunkiem. Wykorzystane zostało również kodowanie Huffmana w celu skompresowania łańcucha.

2. Łańcuchy Browna

Łańcuchy Browna (ang. *Brownian Strings*) powstały z połączenia metody Freemana i techniki segmentacji obrazów Kassa [7]. Po raz pierwszy technika ta została opisana przez R. P. Grzeszczuka i D.N. Levina w [8]. w artykule tym opisano technikę segmentacji obrazu, w której arbitrażowo naszkicowany kontur był stochastycznie deformowany do momentu dopasowania się do żądanego kształtu obiektu.

Segmentacja obiektów polega na poprowadzeniu zamkniętej skierowanej krzywej łamanej złożonej z tzw. segmentów „*crack edges*" [9]. Każdy z tych segmentów oddziela na obrazie dwa sąsiadujące ze sobą piksele. Ważny do odnotowania jest fakt, że skoro każdy segment oddziela dwa piksele, to również długość każdego segmentu jest identyczna i równa długości piksela na obrazie. Oczywiście pod warunkiem, że wszystkie piksele na obrazie mają identyczną wysokość i szerokość.



Rysunek 3: Łańcuchy Browna - Możliwe kierunki segmentów oddzielających piksele

Ponieważ segmenty oddzielają piksele, więc istnieją cztery możliwe kierunki segmentów oddzielające piksele: z lewej na prawą, z prawej na lewą, z góry do dołu i z dołu do góry. Zostały one symbolicznie nazwane R (ang. *Right*), L (ang. *Left*), D (ang. *Down*) i U (ang. *Up*). Na rysunku 3 przedstawione zostały kierunki segmentów.

Po wprowadzeniu skierowania segmentów, można już zapisać kontur obiektu, jako kod łańcuchowy krzywej łamanej składający się ze skierowanych segmentów. Na rysunku 4 umieszczono obiekt oraz kontur składający się ze skierowanych segmentów. Dla takiego obiektu kod łańcuchowy Browna może mieć postać:

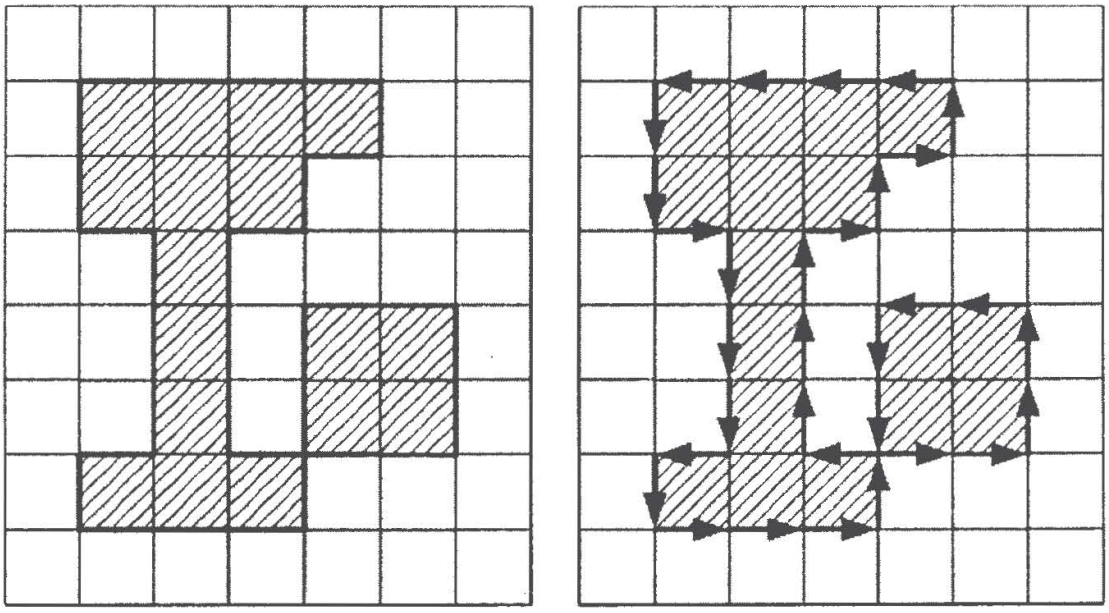
DDRDDDLDRRRURRUULLDDLUUURURULLLL

Kodowanie rozpoczęto w lewym górnym rogu obiektu. Ponieważ jednak kontur jest reprezentowany przez zamkniętą krzywą, więc kontur obiektu umieszczonego na rysunku 4 równie dobrze można opisać przy użyciu kodu łańcuchowego jako:

LLLLDDRDDDLDRRRURRUULLDDLUUURURU

W związku z tą niejasnością, aby zapewnić jednoznaczność zapisu należy albo ustalić sposób wyboru punktu startowego albo jawnie go podać. Po ustaleniu punktu startowego, w lewym górnym rogu obiektu (rysunek 5), kod łańcuchowy może mieć postać:

(1,1)DDRDDDLDRRRURRUULLDDLUUURURULLLL

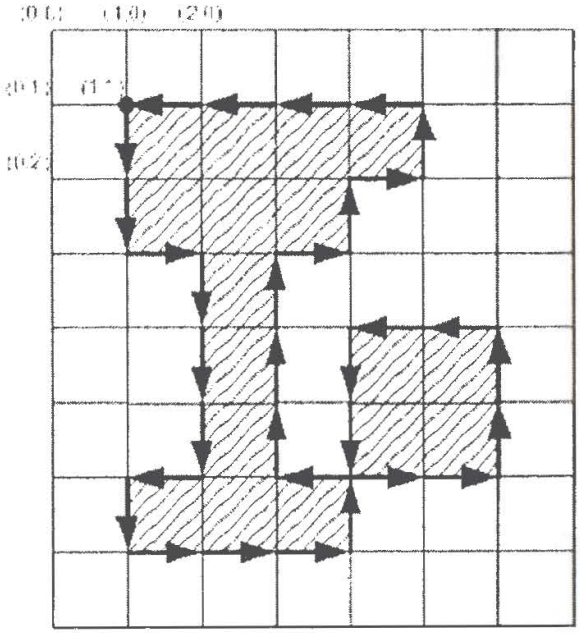


Rysunek 4: Łańcuchy Browna - Segmentacja obiektu przy wykorzystaniu skierowanego konturu

3. Język PDL

Język PDL (ang. *Picture Description Language*) jest jednym z pierwszych języków pozwalający opisać obraz w lingwistyczny sposób. Został on zaprezentowany w [10,11,12]. Jego autorem jest Alan C. Shaw.

Alan Shaw pracował przy akceleratorze cząstek (ang. *particle accelerator* lub *atom smasher*) i potrzebował mechanizmu umożliwiającego przechowanie i późniejszą analizę toru ruchu cząstek elementarnych. Jego sposób opisu stosuje się do zapisu „sztucznych obrazów" takich, jak: linie, krzywe, schematy, proste struktury. Sformułowane zostały zasady tworzenia i przekształcania struktur, aby umożliwić ich generowanie, porównywanie i prymitywne rozpoznawanie. Opracowany przez Shawa język opisuje elementy dwuwymiarowe, ale możliwe jest również uogólnienie problemu na większą liczbę wymiarów bez wprowadzenia znaczących zmian.



Rysunek 5: Łańcuchy Brawna - Skierowany kontur obiektu z zaznaczeniem punktu startu

Język PDL umożliwia opis złączeń prymitywnych elementów (prymitywów) na obrazie. Wybór prymitywów zależy tylko i wyłącznie od założeń konkretnego problemu. Prymitywami mogą być dowolne dwuwymiarowe niepodzielne obiekty-obrazy, posiadające dwa wyróżnione punkty: ogon (ang. *tail*) i głowę (ang. *head*). Wyróżnione punkty służą do złączania prymitywów. Ponieważ obiekty te są niepodzielne, to postrzegamy je jako całość, a nie przez pryzmat elementów składowych. Prymitywy łączone są tylko za pomocą wy-

różnionych punktów. Formalnie, język PDL przekształca obraz w etykietowany graf skierowany. Stosując terminologię grafów, prymitywom z wyróżnionymi dwoma punktami odpowiadają krawędzie skierowane. w związku z dużymi podobieństwami stosowane jest również określenie „graf obrazu".

Prymitywy obrazu powinny być przedstawiane przy użyciu prymitywnych klas, które z kolei, przedstawiane są przy użyciu listy atrybutów. Lista taka powinna zawierać nazwę klasy, specyfikację głowy i ogona, oraz listę dodatkowych atrybutów jednoznacznie opisujących prymityw. Na liście dodatkowych atrybutów klasy mogą pojawić się nadmiarowe informacje. Można je pozostawić, jeśli w dalszych przekształceniach lub analizie mogą być istotne z jakiegoś powodu.

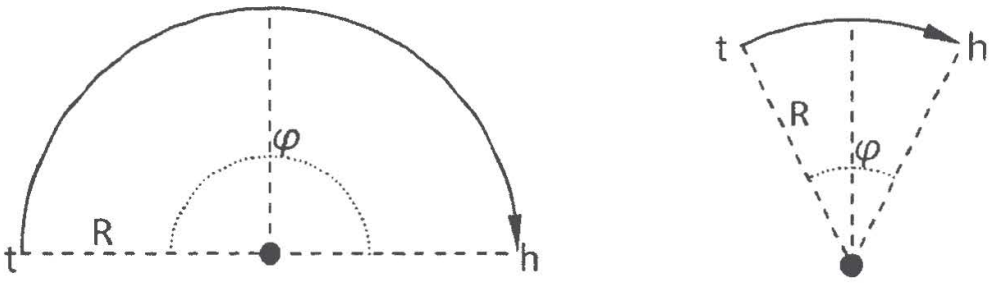
$$klasa\ prymitywu \triangleq \left\{ \begin{array}{c} \langle Nazwa\ klasy \rangle, \\ \langle Specyfikacja\ ogona \rangle, \\ \langle Specyfikacja głowy \rangle, \\ \langle 1\ atrybut \rangle, \\ \ldots, \\ \langle n\ atrybut \rangle \end{array} \right\}$$

Zdefiniujmy np. klasę łuku na okręgu o dowolnym promieniu, skierowanym zgodnie z ruchem wskazówek zegara a kącie mniejszym niż 180° i łuku. Nazwijmy tę klasę ARC. Opis tej klasy może mieć postać:

$$ARC \triangleq$$

$$\left\{ \begin{array}{c} ARC, \\ koniec\ krzywej,\ wcześniejszy\ zgodnie\ z\ ruchem\ wskazówek\ zegara, \\ koniec\ krzywej,\ późniejszy\ zgodnie\ z\ ruchem\ wskazówek\ zegara, \\ R > 0, \\ \varphi < 180° \end{array} \right\}$$

Na rysunku 6 przedstawione zostały przykładowe prymitywy należące do klasy ARC. Literkami „*t*" i „*h*" zaznaczono odpowiednio ogon i głowę. Symbol „*R*" oznacza promień, natomiast „φ" kąt łuku.



Rysunek 6: Język PDL - przykład prymitywu należącego do klasy ARC

Dopuszczalne jest tworzenie pustych (niewidocznych) lub nieznaczących prymitywów. Utworzenie takich obiektów może być konieczne w celu połączenia rozłącznych części obrazu lub w celu wyrażenia geometrycznej zależności pomiędzy częściami obrazu. Do specyfikacji języka PDL dodano również specjalny pusty prymitywny obiekt (ang. null point). Oznaczono go przy pomocy symbolu λ. Składa się on tylko z jednego punktu, w którym znajduje się zarówno ogon jak i głowa. Rolą tego specjalnego prymitywu jest etykietowanie węzłów w grafie obrazu.

Składnia języka

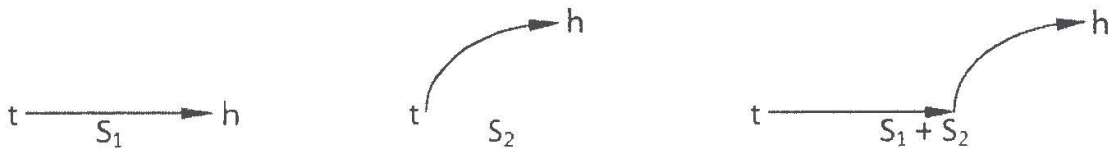
Składnia języka opisuje dozwolone operacje przewidziane przez twórców języka.

Niech S_1 oraz S_2 będą poprawnie zbudowanymi wyrażeniami zgodnie z regułami języka PDL. Dodatkowo niech p będzie prymitywem pewnej klasy S jest poprawnie zbudowanym wyrażeniem języka PDL, jeśli:

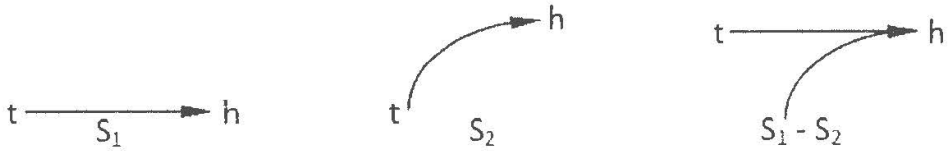
- S jest jednym z prymitywów $(S \rightarrow p)$
- S powstał poprzez konkatenację poprawnie zbudowanych wyrażeń przy użyciu jednego z dozwolonych operatorów $(S \rightarrow S_1 \theta S_2)$
- jedynymi dopuszczalnymi binarnymi operatorami konkatenacji są $\theta \rightarrow \{+, -, *, \times, \sim\}$.
- Ogonem konkatenacji dwóch wyrażeń jest ogon pierwszego wyrażenia $Tail(S_1 \theta S_2) = Tail(S_1)$
- głową konkatenacji dwóch wyrażeń jest głowa drugiego wyrażenia
- $Head(S_1 \theta S_2) = Head(S_2)$

- S jest „odwróceniem" poprawnie zbudowanego wyrażenia $(S \rightarrow \sim S_1)$
- S jest „ukryciem" poprawnie zbudowanego wyrażenia $(S \rightarrow \overline{(S_1)})$
- S jest transformacją poprawnie zbudowanych wyrażeń $(S \rightarrow T(\omega)S_1)$
- operatory $\beta \rightarrow \{\sim, \overline{S}, S_1(\omega)\}$ są operatorami unarnymi
- S jest poprawnie zbudowanym wyrażeniem opatrzonym etykietą $(S \rightarrow S_i^j)$

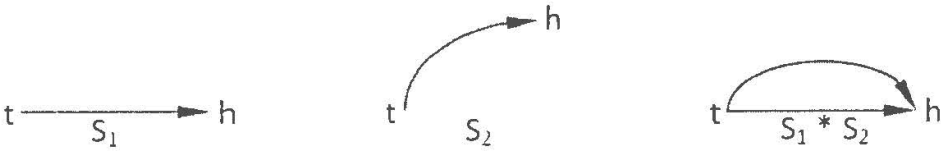
Na rysunkach 7, 8, 9 i 10 zaprezentowano sposób działania binarnych operatorów konkatenacji. Operatory +, -, x są zdefiniowane dla wszystkich kombinacji wyrażeń. w przypadku operatora * mogą istnieć takie kombinacje wyrażeń, dla których wykonanie konkatenacji nie jest możliwe.



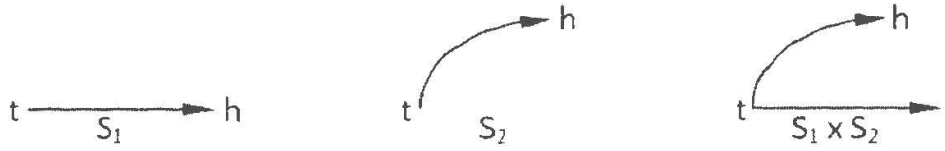
Rysunek 7: Język PDL - operator konkatenacji +



Rysunek 8: Język PDL - operator konkatenacji -



Rysunek 9: Język PDL - operator konkatenacji *



Rysunek 10: Język PDL - operator konkatenacji x

Operator binarny ~ został zdefiniowany przy wykorzystaniu operatora unarnego ~, tj. $S_1 \sim S_2 \equiv S_1 + (\sim S_2)$. Zamienia on miejscami dwa wyróżnione punkty. Na rysunku 11 umieszczono ilustrację działania tego operatora. Natomiast na 12 rysunku przedstawiono zasadę działania operatora „ukrycia". Operator ten powoduje, że obiekt będzie składał się tylko z punktów pustych (ang. *null points*).

Rysunek 11: Język PDL - operator binarny ~

Rysunek 12: Język PDL - operator binarny „ukrycia"

Ostatnim zdefiniowanym operatorem jest operator transformacji. w zapisie $T(\omega)S_1T(\omega)$ oznacza afiniczną transformację, polegającą na skalowaniu, obrocie i przesunięciu. z matematycznego punktu widzenia jest to jednoznaczne przekształcenie płaszczyzny na płaszczyznę.

Przykład

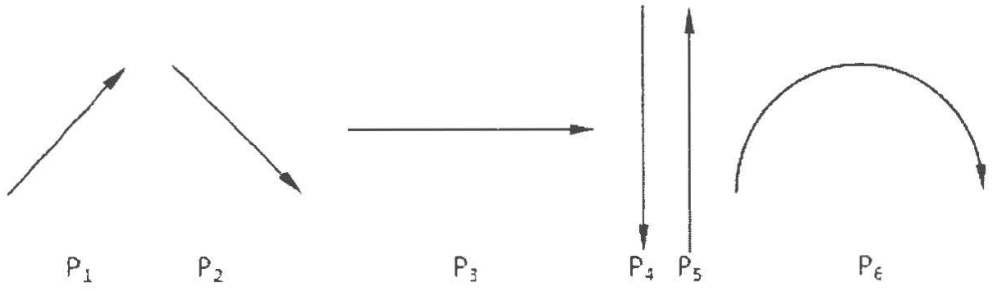
Przy wykorzystaniu prymitywów umieszczonych na rysunku 13 opiszmy schemat domu z zaokrąglonym i z trójkątnym dachem (rysunek 14).

Klasę domów H_1 z okrągłym dachem możemy opisać, jako następującą sekwencję wyrażeń języka PDL:

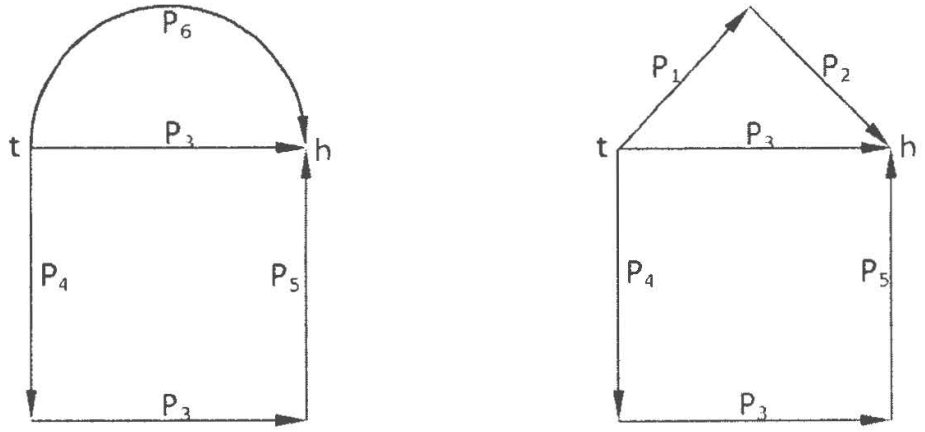
$$H_1 \rightarrow (P_6 * P_3) * ((P_4 + P_3) + P_5)$$

Natomiast klasę domów H2 z trójkątnym dachem możemy opisać, jako:

$$H_2 \rightarrow ((P_1 + P_2) * P_3) * ((P_4 + P_5) + P_5)$$



Rysunek 13: Język PDL – prymitywy wykorzystane do opisania domu

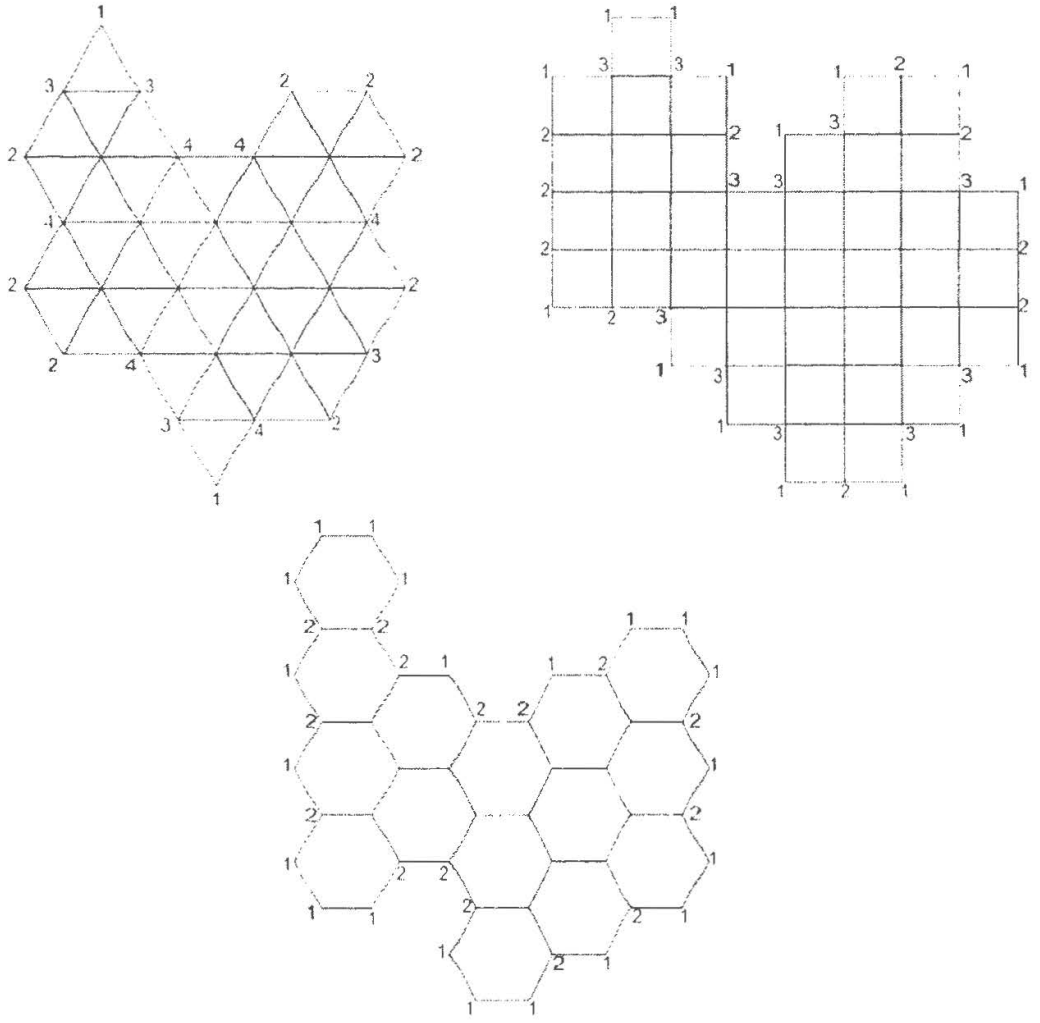


Rysunek 14: Język PDL – schemat domu z zaokrąglonym i z trójkątnym dachem

4. Wierzchołkowy kod łańcuchowy

Wierzchołkowy kod łańcuchowy (ang. VCC – *vertical chain code*) został utworzony przez E. Bribiesca i zaprezentowany w [13]. Jednym z podstawowych założeń, jakie zostały przyjęte przez autora było przekonanie,

że każdy obiekt można przedstawić w postaci siatki regularnych kształtów. Zaproponował użycie trzech rodzajów siatek: trójkątnej, kwadratowej i sześciokątnej (rysunek 15). Elementami kodu są wartości odpowiadające liczbie komórek obiektu spotykających się w danym węźle konturu. w zależności od wybranej siatki zakres wartości elementów kodu jest różny. w przypadku siatki trójkątnej dozwolone wartości należą do przedziału od 1 do 5. Dla siatki kwadratowej przedział ten wynosi od 1 do 3. Natomiast dla siatki sześciokątnej możliwymi wartościami są tylko 1 i 2.



Rysunek 15: VCC - elementy kodu VCC na siatce trójkątnej, kwadratowej i sześciokątnej

Przykładowe reprezentacje obiektu wykorzystujące różne rodzaje siatek przedstawiono na rysunku 15. Założono również, że łańcuch powstały z opisu brzegu figury ma być zamkniętym konturem

Łańcuch VCC powstaje poprzez konkatenację wartości elementów kodu. Podobnie, jak ma to miejsce w innych algorytmach wywodzących się z kodu Freemana, zarówno kierunek tworzenia łańcucha jak i wybór punktu, od którego rozpoczniemy generowanie łańcucha ma znaczenie.

Ponieważ elementy kodu VCC posiadają wartość, łańcuch utworzony z tych elementów można traktować jak liczbę naturalną. w zależności od wyboru punktu startu otrzymujemy inną liczbę. E. Bribiesca proponuje wybrać punkt startu w taki sposób, aby odpowiadająca łańcuchowi liczba naturalna miała najmniejszą możliwą wartość. Autor nie dokonuje wyboru, w którym kierunku (zgodnie czy przeciwnie do ruchu wskazówek zegara) ma być tworzony łańcuch, dokonany wybór nie wpływa na dalszą analizę.

Po uwzględnieniu propozycji E. Bribiesca i po ustaleniu domyślnego kierunku tworzenia łańcuchów VCC, dla obiekt z rysunku 15 na siatce kwadratowej zostanie przypisany kod:

11312221231313121313122132121313321 3

Podsumowanie

Zaproponowano wiele różnych ciągowych struktur symbolicznych. Zaprezentowane w tej pracy przedstawiają najszersze ujęcie tematu i prezentują najpowszechniej stosowane rozwiązania. Zarówno możliwości jak i oczekiwania stawiane obecnie przed metodami symbolicznymi są ogromne. Opublikowano wiele prac prezentujących wykorzystanie struktur symbolicznych w systemach rozpoznawania kształtów oraz w medycznych systemach wspomagających podejmowanie decyzji.

Możliwość generowania praktycznie nieskończonego zbioru oczekiwanych rezultatów jest zarazem ogromną zaletą metod symbolicznych jak i jednym z najpoważniejszych problemów. Złożoność obliczeniowa związana rozmiarem analizowanego zbioru rezultatów jak i z samym procesem przetwarzaniem struktur symbolicznych może wpływać na rezygnację z użycia struktur symbolicznych. Jednakże ustawiczny rozwój mocy obliczeniowej współczesnych urządzeń elektronicznych sprawia, że prawdopodobieństwo powstawania kolejnych systemów opartych o struktury symboliczne stale wzrasta.

Literatura

[1] Tadeusiewicz, R., Ogiela, M. R. (2004) : *Medical Image Understanding Technology*, Springer Berlin/Heidelberg.

[2] Freeman, H. (1961): On the encoding of arbitrary geometric configurations, *IRE Trans. Electron. Computers* EC-10, 260-268.

[3] Freeman, H. (1974): Computer Processing of Line-Drawing Images, *ACM Computing Surveys* (CSUR), 57-97.

[4] Bons, J. H., Kegel, A. (1977): *On the Digital Processing and Transmission of Handwriting and Sketching*, Proceedings of EUROCON 77.

[5] Hwang, Y. T., Wang, Y. C., Wang, S. S. (2001): An efficient shape coding scheme and its codec design, *IEEE Workshop on Signal Processing Systems*, 225-232.

[6] Liu, Y. K., Zalik, B. (2005): An efficient chain code with Huffman coding, *Pattern recognition*, 553-557.

[7] Kass, M., Witkin, W., Terzopoulos, D. (1988): Snakes: Active Contour Models, *International Journal of Computer Vision*.

[8] Grzeszczuk, R. P., Levin, D. N. (1997): "Brownian strings": Segmenting images with stochastically deformable contours, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1100-1114.

[9] Grzeszczuk, R., Eddins, S., DeFanti, T. (1992): *A Region Fill Algorithm Based on Crack Contour Boundaries*, Univ. of Illinois Technical Report UIC-EECS-92-6.

[10] Miller, W. F., Shaw, A. C. (1967): *A Picture Calculus*, Stanford University SLAC-PUB-358.

[11] Miller, W. F., Shaw, A. C. (1968): *Linguistic methods in picture processing: a survey*, AFIPS Joint Computer Conferences, 279-290.

[12] Shaw, A. C. (1968): *The formal description and parsing of pictures*, Stanford Linear Accelerator Center.

[13] Bribiesca, E. (1999): *A new chain code, Pattern Recognition*, 235-251.