UNIWERSYTET GDAŃSKI
Wydział Matematyki, Fizyki i Informatyki

Szymon Rękawek
nr albumu: 206288

Gra Thuego

Praca magisterska na kierunku:
INFORMATYKA
Promotor:
prof. dr hab. T. Dzido

Gdańsk 01.01.2016

Streszczenie

Słowa kluczowe

Thue

Spis treści

Wprowadzenie

Tematem niniejszej pracy jest Gra Thuego. Axel Thue był norweskim matematykiem żyjącym w latach 1863 - 1922, znanym z prac z zakresu kombinatoryki.

Thue pracował nad problemami, powstałymi w wyniku badań nad sekwencjami symboli. Praca Thue [1] opisywała problem, który autor nazwał *nieredukowalne słowa* (*irreducible words*). Poświęca w niej szczególną uwagę dwu i trzy literowym przypadkom. W skrócie wprowadza pojęcie znane obecnie jako *Thue-Morse word* i pokazuje, że nieskończone słowa bez nasunięć (*Overlap-free*) są pochodnymi tej sekwencji. W swoich pracach definiuje kolejną strukturę, a mianowicie infinite *Square-Free word*, oraz przedstawia sposoby generowania nieskończenie długich słów wolnych zarówno od kwadratów jak i nasunięć.

Gra która powstała na podstawie teorii Thuego w skrócie polegała będzie na utworzeniu jak najdłuższego ciągu znaków nad określonym z góry alfabetem. Zależnie od trybu gry, kończyć się ona będzie w momencie gdy pojawi się zdefiniowany na początku rodzaj powtórzenia w tworzonym przez nas, bądź algorytm ciągu. Jednym z trybów gry jest walka komputera przeciwko niemu samemu, po takiej rozgrywce przedstawione zostaną złożoności czasowe oraz wnioski wynikające z obranej przez oponentów taktyki.

Ostatnia część pracy poświęcona jest analizie algorytmów zarówno pod względem czasu ich wykonywania jak i zdolności do przewidywania ruchów przeciwnika.

ROZDZIAŁ 1

Teorie Axela Thue na temat sekwencji symboli

lbaldsaldsal dsadsa asdkdsak sadkdsa dsadsadsak dsak dsa dsajdsajdas

1.1. Definicje

- Alfabet jest skończonym zbiorem symboli lub liter.
- Słowo alfabetu A jest skończoną sekwencją elementów z A.
- Długość słowa ω jest reprezentowana przez $|\omega|$.
- Puste słowo o długości 0 jest reprezentowane przez ε.
- Czynnik (factor) słowa ω jest słowem u, które występuje wewnątrz ω formie $\omega = xuy$, podczas gdy x oraz y również są słowami tego alfabetu.
- Kwadrat (square) jest niepustym słowem w formie uu, gdzie u jest niepuste.
- Square-free, słowo jest wolne od kwadratów, jeśli żaden z jego czynników nie jest kwadratem.
- Nasunięcie (Overlap) jest słowem w formie $xuxux$, gdzie x jest niepuste. Nazwa pojęcia wzięła się z tego, że xux występuje dwa razy w $xuxux$. Pierwszy raz jako prefiks (początkowy czynnik) oraz jako sufiks (końcowy czynnik) i te dwa wystąpienia mają wspólną część - centralne x, a więc *nasuwają* się na siebie.

- Overlap-free - słowo w którym żaden z czynników nie nasuwa się na siebie. W definicji Axela Thue słowo ω w alfabecie długości n jest nieredukowalne jeśli jakiekolwiek dwa wystąpienia tego samego słowa jako czynnik wewnątrz ω są zawsze oddzielone od siebie przez $n-2$ liter. Oznacza to, że nieredukowalne dwuliterowe słowo jest bez nasunięć i nieredukowalne trzyliterowe słowo jest bez kwadratów.
- Morfizm - mapowanie obiektu z jednej matematycznej struktury w inną.

1.2. Thue-Morse word

Rozważmy nieskończone słowo

$$01101001100101101001011001101001...$$

Zostało ono nazwane po Thue, który badał jego właściwości w referacie z 1906 roku, oraz Morsie, który odkrył je na nowo w latach 20 XIX wieku. Słowo Thue-Morse'a występuje również o wiele wcześniej w wiadomościach Prouheta[200] z Francuską Akademią Nauk w 1851 roku. Rzeczywiście Prouhet podał więcej ogólnych konstrukcji, uzyskując nie tylko słowo Thue-Morse'a, ale całą rodzinę słów na większych alfabetach mających inne interesujące właściwości. Słowa te czasami odnoszą się do ogólnych słow Thue-Morse'a lub słów Prouheta.

Niech $A = \{a, b\}$ będzie dwuliterowym alfabetem. Rozważmy morfizm μ z monoidu $A*$, który definiuje się następująco:

$$\mu(a) = ab, \qquad \mu(b) = ba$$

Dla $n \geq 0$:

$$u_n = \mu^n(a), \qquad v_n = \mu^n(b)$$

Wtedy:

$$\begin{aligned} u_0 &= a & v_0 &= b \\ u_1 &= ab & v_1 &= ba \\ u_2 &= abba & v_2 &= baab \\ u_3 &= abbabaab & v_3 &= baababba \end{aligned}$$

Wzór ogólny:

$$u_{n+1} = u_n v_n, \qquad v_{n+1} = v_n u_n$$

oraz:

$$u_n = \overline{v}_n, \qquad v_n = \overline{u}_n$$

gdzie $\overline{\omega}$ jest uzyskiwane z ω przez zamianę a oraz b. Słowa u_n i v_n są często nazywane *Blokami morsa*. Można łatwo zauważyc że u_{2n} oraz v_{2n} są palindromami oraz to że $u_{2n+1} = \neg v_{2n+1}$, gdzie $\neg w$ jest negacją w. Morfizm μ może być rozszerzony do nieskończonych słow, które mają dwa stałe punkty:

$$t = abbabaabbaababbabaab... = \mu(t)$$
$$t = baababbaabbabaababba... = \mu(\ t)$$

Przedstawione powyżej słowo t jest sekwencją Thue-Morse'a. Jest wiele innych sposobów na stworzenie tego słowa. Niech t_n będzie n-tym symbolem w t, zaczynając od $n = 0$. Wtedy można pokazać, że:

$$t_n = \left\{ \begin{array}{l} a\ if\ d_1(n) \equiv (mod\ 2) \\ a\ if\ d_1(n) \equiv (mod\ 2) \end{array} \right\}$$

gdzie $d_1(n)$ jest liczbą bitów równych 1 w binarnej reprezentacji n.

Dla $n \leq 12$ oraz $n \in \mathbb{N}$ generowane jest następujące słowo:

$$\begin{array}{llll} bin(0) & = 0, & d_1(0) & = 0\ mod\ 2 = 0 \rightarrow a \\ bin(1) & = 1, & d_1(1) & = 1\ mod\ 2 = 1 \rightarrow b \\ bin(2) & = 10, & d_1(2) & = 1\ mod\ 2 = 1 \rightarrow b \\ bin(3) & = 11, & d_1(3) & = 2\ mod\ 2 = 0 \rightarrow a \\ bin(4) & = 100, & d_1(4) & = 1\ mod\ 2 = 1 \rightarrow b \\ bin(5) & = 101, & d_1(5) & = 2\ mod\ 2 = 0 \rightarrow a \\ bin(6) & = 110, & d_1(6) & = 2\ mod\ 2 = 0 \rightarrow a \\ bin(7) & = 111, & d_1(7) & = 3\ mod\ 2 = 1 \rightarrow b \\ bin(8) & = 1000, & d_1(8) & = 1\ mod\ 2 = 1 \rightarrow b \\ bin(9) & = 1001, & d_1(9) & = 2\ mod\ 2 = 0 \rightarrow a \end{array}$$

$$\begin{aligned} bin(10) &= 1010, & d_1(10) &= 2\ mod\ 2 = 0 \rightarrow a \\ bin(11) &= 1011, & d_1(11) &= 3\ mod\ 2 = 1 \rightarrow b \\ bin(12) &= 1100, & d_1(12) &= 2\ mod\ 2 = 0 \rightarrow a \end{aligned}$$

$$t = abbabaabbaaba$$

W konsekwencji istnieje skończony automat obliczający wartości t_n. Automat ten ma dwa stany końcowe 0 oraz 1. Na początku czyta łańcuch znaków $bin(n)$ od lewej do prawej, zaczynając od $n = 0$. Ostateczny stan równy jest 0 lub 1 i definiuje czy t_n jest równe a lub b. W skrócie obliczenie jakie wykonuje automat to $d_1(n)\ modulo\ 2$.

1.3. Square-free word

Łatwo można zauważyc, że jedynymi słowami bez kwadratów w alfabecie $A = \{a, b\}$ są: a, b, ab, ba, aba, bab. Istnieje jednak dowolnie długi ciąg znaków wolny od kwadratów dla słów nad alfabetem trzyliterowym. By stworzyć dowolne słowo wolne od kwadratów Thue wymyślił następujący algorytm.
Mając alfabet $A = \{a, b, c\}$ należy zastąpić każde wystąpienie litery a przez $abac$, b przez $babc$ oraz c przez $bcac$, jeśli jest poprzedzone przez a lub $acbc$, jeśli jest poprzedzone przez b. Zaczynając od litery a otrzymujemy nieskończone słowo które nie zawiera kwadratów.

$$abacbabcabacbcacbabcabacbabcacbcabacbabc...$$

W 1912 roku Axel Thue wymyślił inny sposób na generowanie nieskończonego słowa bez kwadratów na trzech literach z użyciem następującego morfizmu.

- $a \rightarrow abcab$
- $b \rightarrow acabcb$
- $c \rightarrow acbcacb$

Po raz kolejny zastępujemy każde wystąpienie z naszych liter przez zdefiniowane sekwencje. Jest to dość skomplikowana struktura, zsumowana długość łańcuchów wynosi 18. A Carpi [7] dowiódł, że morfizm na alfabecie składającym się z trzech liter tworzący słowa wolne od kwadratów musi mieć długość równą co najmniej 18.

1.4. Zasady gry

W grze dostępnych jest kilka trybów zarówno dla jednego oraz dwóch graczy jak i komputerowa symulacja, czyli gra doskonała komputera przeciwko niemu samemu.

Zasady trybu **Longest Square-Free word**, dla dwóch graczy są następujące. Na początku gry gracze ustalają moc zbioru kolorów, który musi mieć co najmniej 3 elementy oraz otrzymują swoje role, jeden z nich staje się architektem, drugi malarzem. Rola architekta polega na wybieraniu odpowiedniego indeksu w ciągu tworzonym przez graczy, pod którym powstanie nowy element. Malarz natomiast określa kolor wstawianego elementu. Gra kończy się w momencie, gdy w ciągu tworzonym przez graczy pojawia się **kwadrat**.

Indeks i podawany przez architekta nie może być mniejszy od zera oraz większy niż n, gdzie n jest równe liczbie elementów w ciągu. Grę rozpoczyna architekt, podany przez niego indeks w pierwszym ruchu musi wynosić 0, ponieważ $n = 0$. Gracze wykonują swoje ruchy na przemian. Malarz otrzymuje punkt za każdy pomalowany element, który nie tworzy **kwadratu** wewnątrz ciągu. By wyłonić zwycięzcę potrzebne są dwie rundy. Każdy z graczy musi się sprawdzić się zarówno jako malarz i architekt. Wygrywa osoba, która zdobyła więcej punktów jako malarz.

Tryb ten dostępny jest również dla jednego gracza, rolę przeciwnika otrzymuje wtedy komputer, który działa według algorytmu przewidującego określoną przez poziom trudności liczbę ruchów do przodu. Algorytm może pełnić zarówno rolę budowniczego jak i malarza.

Bliźniaczym trybem gry, opierającym się na tych samych zasadach z

niewielką różnicą jest **Longest Overlap-Free word**. Różnica polega na tym, że malarz w tworzonym ciągu musi unikać **nasunięcia** oraz moc zbioru kolorów musi mieć co najmniej 2 elementy.

Podobnie jak w **Longest Square-Free word** jest możliwość gry przeciwko algorytmowi, który jest w stanie przewidywać określoną ilość ruchów do przodu.

ROZDZIAŁ 2

Algorytmy obsługujące grę

2.1. Algorytm wyszukiwania powtórzeń kwadratowych

```
        private List<Integer> findSquare() {
                List<Integer> squareSeq = null;
                int maxSeqSize = sequence.size()/2;
                int minSeqSize = 1;
                for(int subSeqSize=minSeqSize;
                        subSeqSize<=maxSeqSize;
                        subSeqSize++) {
                        squareSeq = compareSubSeq(subSeqSize);
                        if(squareSeq != null) {
                                return squareSeq;
                        }
                }
                return null;
        }
```

ROZDZIAŁ 3

Narzędzia i standardy pokrewne

Systemy SGML, ze względu na mnogość funkcji jakie spełniają i ich kompleksowe podejście do oznakowywania i przetwarzania dokumentów tekstowych, są bardzo skomplikowane. Możemy wyróżnić dwa podejścia do budowy takich systemów. Z jednej strony, buduje się systemy zindywidualizowane, oparte o specyficzne narzędzia tworzone w takich językach, jak: C, C++, Perl czy Python. Edytory strukturalne, filtry do transformacji formatów czy parsery i biblioteki przydatne do konstrukcji dalszych narzędzi, tworzone są według potrzeb określonych, pojedynczych systemów.

Z drugiej strony, twórcy oprogramowania postanowili pójść krok dalej i połączyć te różne narzędzia w jedną całość. Tą całość miał stanowić DSSSL lub jego XML-owy odpowiednik – standard XSL. Ze względu na oferowane możliwości można twierdzić, że tworzenie i używanie narzędzi implementujących standard DSSSL/XSL, jest najwłaściwszym podejściem. Przemawiają za tym różne argumenty, ale najważniejszym z nich jest to, że mamy tu możliwość stworzenia niezależnego od platformy programowej i narzędziowej zbioru szablonów – przepisów jak przetwarzać dokumenty SGML.

3.1. Przetwarzanie dokumentów SGML – standard DSSSL

DSSSL (*Document Style Semantics and Specification Language*) – to międzynarodowy standard ściśle związany ze standardem SGML. Standard ten, można podzielić na następujące części:

- język transformacji (*transformation language*). To definicja języka słu-

żącego do transformacji dokumentu oznaczonego znacznikami zgodnie z pewnym DTD na dokument oznaczony zgodnie z innym DTD.

- język stylu (*style language*) opisujący sposób formatowania dokumentów SGML.
- język zapytań (*query language*) służy do identyfikowania poszczególnych fragmentów dokumentu SGML.

Opisane główne części składowe standardu DSSSL dają obraz tego, jak wiele aspektów przetwarzania zostało zdefiniowanych i jak skomplikowany jest to problem. Jest to głównym powodem tego, że mimo upływu kilku lat od zdefiniowania standardu nie powstały ani komercyjne ani wolnodostępne aplikacje wspierające go w całości. Istnieją natomiast *nieliczne* narzędzia realizujące DSSSL w ograniczonym zakresie, głównie w części definiującej język stylu, który odpowiada za opatrzenie dokumentu czysto strukturalnego w informacje formatujące. Daje to możliwość publikacji dokumentów SGML zarówno w postaci elektronicznej, hipertekstowej czy też drukowanej.

3.2. Przetwarzanie dokumentów XML – standard XSL

Tak jak XML jest *uproszczoną* wersją standardu SGML, tak XSL jest uproszczonym odpowiednikiem standardu DSSSL. W szczególności, wyróżnić można w tym standardzie następujące części składowe:

- język transformacji (XSLT) To definicja języka służącego do transformacji dokumentu.
- język zapytań (XPath) służy do identyfikowania poszczególnych fragmentów dokumentu.
- język stylu deefiniujący sposób formatowania dokumentów XML.

ROZDZIAŁ 4

Przegląd dostępnych narzędzi

W celu wykorzystania standardu SGML do przetwarzania dokumentów, niezbędne jest zebranie odpowiedniego zestawu narzędzi. Narzędzi do przetwarzania dokumentów SGML jest wiele. Są to zarówno całe systemy zintegrowane, jak i poszczególne programy, biblioteki czy skrypty wspomagające.

4.1. Narzędzia do przeglądania dokumentów SGML

Do tej kategorii oprogramowania zaliczamy przeglądarki dokumentów SGML oraz serwery sieciowe wspomagające standard SGML, przy czym rozwiązań wspierających standard XML jest już w chwili obecnej dużo więcej i są dużo powszechniejsze.

Jeżeli chodzi o przeglądarki to zarówno Internet Explorer jak i Netscape umożliwiają bezpośrednie wyświetlenie dokumentów XML; ponieważ jednak nie wspierają w całości standardu XML, prowadzi to ciągle do wielu problemów[1].

[1] Z innych mniej popularnych rozwiązań można wymienić takie aplikacje, jak: HyBrick SGML Browser firmy Fujitsu Limited, Panorama Publisher firmy InterLeaf Inc, DynaText firmy Inso Corporation czy darmowy QWeb. W przypadku serwerów zwykle dokonują one transformacji „w locie" żądanych dokumentów na format HTML (rzadziej bezpośrednio wyświetlają dokumenty XML). Ta kategoria oprogramowania ma, z punktu widzenia projektu, znaczenie drugorzędne.

4.2. Parsery SGML

Program `nsgmls` (z pakietu `SP` Jamesa Clarka) jest doskonałym parserem dokumentów SGML, dostępnym publicznie. Parser `nsgmls` jest dostępny w postaci źródłowej oraz w postaci programów wykonywalnych przygotowanych na platformę MS Windows, Linux/Unix i inne. Oprócz analizy poprawności dokumentu parser ten umożliwia również konwersję danych do formatu ESIS, który wykorzystywany jest jako dane wejściowe przez wiele narzędzi do przetwarzania i formatowania dokumentów SGML. Dodatkowymi, bardzo przydatnymi elementami pakietu `SP` są: program `sgmlnorm` do normalizacji, program `sx` służący do konwersji dokumentu SGML na XML oraz biblioteki programistyczne, przydatne przy tworzeniu specjalistycznych aplikacji służących do przetwarzania dokumentów SGML.

W przypadku dokumentów XML publicznie dostępnych, parserów jest w chwili obecnej kilkadziesiąt. Do popularniejszych zaliczyć można Microsoft Java XML Parser firmy Microsoft, LT XML firmy Language Technology Group, Exapt oraz XP (James Clark)

4.3. Wykorzystanie języków skryptowych

4.4. Wykorzystanie szablonów XSL

Stosując wersję XML typu DocBook można wykorzystać szablony stylów przygotowane w standardzie XSL (autor N. Walsh). W chwili obecnej są dostępne narzędzia umożliwiające przetworzenie dokumentów XML do postaci drukowanej (Adobe PDF) oraz hipertekstowej (HTML).

Podobnie jak w przypadku szablonów DSSSL, szablony stylów XSL są sparametryzowane i udokumentowane i dzięki temu łatwe w adaptacji. Do zamiany dokumentu XML na postać prezentacyjną można wykorzystać jeden z dostępnych publicznie procesorów XSLT (por. tabela 4.1).

XSL:FO jest skomplikowanym językiem o dużych możliwościach, zawierającym ponad 50 różnych „obiektów formatujących”, począwszy od

Nazwa	Autor	Adres URL
sablotron	Ginger Alliance	http://www.gingerall.com
Xt	J. Clark	http://www.jclark.com
4XSLT	FourThought	http://www.fourthought.com
Saxon	Michael Kay	http://users.iclway.co.uk/mhkay/saxon
Xalan	Apache XML Project	http://xml.apache.org

Tabela 4.1. Publicznie dostępne procesory XLST

Źródło: Opracowanie własne

najprostszych, takich jak prostokątne bloki tekstu poprzez wyliczenia, tabele i odsyłacze. Obiekty te można formatować wykorzystując przeszło 200 różnych właściwości (*properties*), takich jak: kroje, odmiany i wielkości pisma, odstępy, kolory itp. W tym dokumencie przedstawione jest absolutne miniumum informacji na temat standardu XSL:FO.

Cały dokument XSL:FO zawarty jest wewnątrz elementu `fo:root`. Element ten zawiera (w podanej niżej kolejności):

- dokładnie jeden element `fo:layout-master-set` zawierający szablony określające wygląd poszczególnych stron oraz sekwencji stron (te ostatnie są opcjonalne, ale typowo są definiowane);
- zero lub więcej elementów `fo:declarations`;
- jeden lub więcej elementów `fo:page-sequance` zawierających treść formatowanego dokumentu wraz z opisem jego sformatowania i podziału na strony.

Zakończenie

Możliwości, jakie stoją przed archiwum prac magisterskich opartych na XML-u, są ograniczone jedynie czasem, jaki należy poświęcić na pełną implementację systemu. Nie ma przeszkód technologicznych do stworzenia co najmniej równie doskonałego repozytorium, jak ma to miejsce w przypadku ETD. Jeżeli chcemy w pełni uczestniczyć w rozwoju nowej ery informacji, musimy szczególną uwagę przykładać do odpowiedniej klasyfikacji i archiwizacji danych. Sądzę, że język XML znacznie to upraszcza.

DODATEK A

Tytuł załącznika jeden

Treść załącznika jeden.

DODATEK B

Tytuł załącznika dwa

Treść załącznika dwa.

Spis tabel

Spis rysunków

Oświadczenie

Ja, niżej podpisany(a) oświadczam, iż przedłożona praca dyplomowa została wykonana przeze mnie samodzielnie, nie narusza praw autorskich, interesów prawnych i materialnych innych osób.

.....	.....
data	podpis