P versus NP

- nowa perspektywa

zastosowań
informatyki,
algorytmiki
i informatyki
teoretycznej

Marek Malinowski

P versus NP - nowa perspektywa

Wydanie pierwsze

ISBN 978-83-933596-1-5

Technologie Teleinformatyczne
Płock 2018

... Nie mogła być błędna tylko dlatego, że była prosta ...
Nie było dokładnie wiadomo, do czego prowadzi, jednak było dość oczywiste, że powinna być udostępniona. W zasadzie chciałem opublikować ten pomysł i powiedzieć: Oto zgrabna koncepcja - wyjaśnia na czym polega ten problem, wyjaśnia to, że jego rozwiązanie jest osiągalne i staje się jasno zdefiniowanym problemem badawczym. Teraz niech wkroczą inni i zobaczymy, co jeszcze będziemy mogli znaleźć. ...

[cytat słów Ralpha Merkle - Steven Levy, "Rewolucja kryptografii", WNT, W-wa, 2002, s. 89-90]

PRZEDMOWA

Od autora [manifest]

Istota rodzącej się w latach 50-tych ubiegłego stulecia tytułowej tematyki, której jestem rówieśnikiem, jest ściśle zwią-zana z pojęciem *ZUPEŁNOŚCI*. Gdy kilka lat temu zetknąłem się z teorią złożoności, byłem nie tylko jej rówieśnikiem – byłem w tym względzie zupełnym laikiem, mimo trwałych związków z informatyką od 1971 roku.

Lektura książki Davida Harela „*Algorytmika. Rzecz o istocie informatyki*", będąc źródłem inspiracji do prób znalezienia odpowiedzi na formułowane w niej pytania, zaowocowała szeregiem pomysłów. Przybrały one postać konkretnych modeli i koncepcji rozwiązania kilku problemów charakteryzujących się luką algorytmiczną.

Książka, którą zdecydowałem się zredagować, powstała na podstawie prac prowadzonych poza nurtem życia naukowego. Nie był to jednak świadomy wybór. Koncepcje odwołujące się do prostych, elementarnych modeli, abstrahujące istniejący zaawansowany aparat matematyczny i formalizmy wysokiego poziomu, były zapewne zbyt szokujące, by je traktować poważnie.

Teraz, gdy szkice tych idei mają bardziej dojrzały kształt, jest czas by stały się jednak przedmiotem weryfikacji i oceny ich wartości także poza środowiskiem akademickim. Dlatego zdecydowałem, że oprócz ograniczonego nakładu książki w trady-cyjnej postaci, prace udostępnione zostaną także w postaci publi-kacji elektronicznej. W szczególności wydanie elektroniczne ma

sprzyjać przełamaniu bariery w swobodzie wymiany i udostępniania informacji. W ten sposób nawiązuję w pewnym sensie do idei Timothy Gowersa wyrażanej w nadziei, że wielkie komercyjne wydawnictwa nie będą potrzebne, aby uprawiać naukę.

Wybór takich sposobów prezentacji rezultatów moich prac nie jest wyrazem kontestowania przyjętych w kręgach akademickich norm i zwyczajów. Jest raczej formą ich wzbogacenia o swoisty mechanizm demokracji, który nie należy jednak utożsamiać z ustalaniem prawdy naukowej metodą głosowania. Zamieszczone motto wystarczająco komentuje wybór tej formy demokracji.

Prezentowane wyniki, nie są zapewne wolne od błędów i nieścisłości. Mimo to powinny być zrozumiałe i czytelne. Liczę więc, że znajdą zainteresowanie. Wyrażone oczekiwania wynikają z kilku przesłanek.

Po pierwsze, uważam podobnie jak Ralph Merkle, że zarówno koncepcje jak i same rozwiązania warte są dalszej pracy i „… *nie mogą być błędne tylko dlatego, że są proste*". Otwierając nową perspektywę, mimo destrukcyjnego charakteru, „ … *jest oczywiste, że powinny być udostępnione*".

Po drugie, jeden z dyskutowanych w książce problemów autoryzowali Panowie Adleman, Rivest i Shamir. Zakładam więc, że każdy, dla którego tematyka książki nie jest obca, ma świadomość znaczenia rozwiązania tego i pozostałych dyskutowanych problemów. Pisze o tym Pan Schneier w swojej książce „*Cryptography*". Także Panowie Arora i Barak piszą o tym w przeglądanym przeze mnie drafcie ich książki „*Computational Complexity: A Modern Approach*".

Po trzecie, moje rozwiązania wykorzystują z jednej strony znane już podstawowe wyniki badań w teorii złożoności. Z drugiej strony wykorzystują zupełnie nowe pomysły odwzorowane przez elementarne modele.

Wreszcie po czwarte, traktując tematykę złożoności obliczeniowej jako hobby, chciałbym poznać popełnione błędy, a wśród nich być może błędy kardynalne, jeśli je popełniłem.

Niezależnie od możliwych błędów, wyrażam nadzieję, że udostępnione przez publikację tej książki prace, wnoszą pewien

nowy element w rozwój nauki i w obszary praktycznych zastosowań informatyki.

Także niezależnie od tego jak ta wiara zostanie potraktowana, pozostanie mi satysfakcja z potyczek z piękną, acz trudną algorytmiką, Zaś Wszystkim zwolennikom platonizmu w matematyce, dedykuję krótki esej.

Mam nadzieję, że odebrany zostanie jako przemyślenia ***nie-ZUPEŁNEGO*** laika.

Pojęcie ZUPEŁNOŚCI, *zdefiniowane w teorii złożoności obliczeniowej, jest bytem obiektywnie występującym w realnym świecie, niezależnym od czasu i przestrzeni. Jego istota polega na tym, że „coś nie jest gorsze od czegoś" z jednej strony i „to coś nie jest lepsze od czegoś innego". Do wykazania takich relacji wykorzystywany jest mechanizm redukcji wielomianowej.*

Rok 1736 – Euler formułuje podwaliny teorii grafów, definiuje problem EULER-PATH *oraz* EULER-CYCLE*. Dla obu podaje jedynie algorytmy wykładnicze. Jednocześnie definiuje problem* P *versus* NP *z pierwotnym problemem* NP*-zupełnym, czyli spełnialnością (*SATISFABILITY*). Wykorzystując pojęcie* ZUPEŁNOŚCI *pokazuje istnienie redukcji wielomianowej* EULER-PATH *w obie strony do i z* 3SAT *i tym samym jego przynależność do klasy problemów* NP*-zupełnych.*

Rok 1971 – Cook pokazuje wielomianowe rozwiązanie problemu EULER-PATH *(dany graf G jest grafem Eulera wtedy i tylko wtedy, gdy wszystkie wierzchołki G są stopnia parzystego). Tym samym pokazał relację* $\mathbf{P = NP}$*.*

Zawartość i układ książki

Książka jest zbiorem opracowanych w różnym czasie rezultatów prowadzonych prac. Przedmiotem tych prac był problem spełnialności SAT i jego warianty, wariant problemu faktoryzacji MULT(n) rozkładu iloczynu dwóch liczb pierwszych na czynniki proste oraz problem znajdywania największego wspólnego dzielnika NWD. W szczególności dwa pierwsze problemy (SAT i MULT(n)) rozważane są w modelu maszyny

Turinga oraz równoważnym modelu sieci logicznych. Dla obu problemów i w obu modelach, uzyskane wyniki prowadzą ogólnie do „nieprawdopodobnego" rozwiązania w postaci równości klas ***NC = P = NP***.

Całość książki podzielona jest na dwie części. Część pierw-sza, zatytułowana „*W kręgu spełnialności*" poświęcona jest problemowi spełnialności **SAT**. W części drugiej, zatytułowanej „*W kręgu faktoryzacji*" zostały umieszczone rozdziały poświęcone dyskusji problemów **NWD** i **MULT(n)**. Taki podział powinien ułatwić zaznajomienie się z prezentowanymi treściami, a przy tym czytać je niezależnie.

Klamrą spinającą obie części i poszczególne rozdziały jest siedem ogniw tworzących łańcuch. Pojawia się on jako element graficzny na początku każdego rozdziału. Jednak symbolika łańcucha została użyta przede wszystkim w kontekście poprawności proponowanych rozwiązań problemów. Poprawności rozumianej tak jak przedstawia to J. Gleen Brookshear w swojej książce „Informatyka w ogólnym zarysie", dyskutując problem wędrowca

> *Podróżny posiadający złoty łańcuch z siedmioma ogniwami musi zatrzymać się w osamotnionym hotelu na siedem nocy. Za każdy nocleg musi zapłacić, oddając jedno ogniwo z łańcucha. Jaką najmniejszą liczbę cięć musi wykonać podróżny, aby każdego ranka mógł zapłacić za miniony nocleg, nnie płacąc przy tym z góry za następny?*

i jego rozwiązanie – przecięcie tylko jednego ogniwa.

Na ogólny wynik w postaci równości klas ***NC = P = NP*** składają się rozwiązania przedstawione w poszczególnych rozdziałach. I tak w części pierwszej pokazuję, że:

- dla problemu **3SAT** istnieje wielomianowa jednostajna sieć logiczna, co rozstrzyga kwestię ***P versus NP*** (rozdział 1);
- przy analizie problemów **3SAT** wygodnie jest posłużyć się modelem referencyjnym, który uwzględnia wszystkie możliwe przypadki i warianty problemów **3SAT** oraz, że wprowadzony model referencyjny dla dowolnego *k>1* jest strukturalnie identyczny (rozdział 2);
- dla elementarnych struktur modeli referencyjnych istnieją różnorodne odwzorowania pozwalające konstruować algorytmy

wielomianowe dla **3SAT**, co rozstrzyga równość klas ***P* = *NP*** (rozdział 3);

- wykorzystując równoległość, zarówno dla **2SAT** jak i **3SAT,** można konstruować algorytmy charakteryzujące się czasem polilogarytmicznym, co rozstrzyga równość klas ***NC* = *P*** (rozdział 4).

W części drugiej pokazuję, że:

- problem **NWD** należąc do klasy ***P*** posiada równoległy algorytm *log Time*, co rozstrzyga równość klas ***NC* = *P*** (rozdział 5);
- dla problemu **MULT(n)** istnieje model umożliwiający skonstruowanie równoległego algorytmu *log Time*, co przy założeniu ***NP****-zupełności* problemu **MULT(n)** bezpośrednio rozstrzyga równość klas **NC = NP**, (rozdział 6);
- istnieje redukcja problemu **MULT(n)** do **CIRCUIT SAT**, a tym samym **MULT(n)** należy do klasy problemów ***NP****-zupełnych* oraz, że istnieje wielomianowa jednostajnej sieci logicznej dla **MULT(n)** (rozdział 7).

Każdy rozdział zredagowany został w formie niezależnego opracowania, co praktycznie pozwala czytać je niezależnie. Jednak taki sposób redakcji sprawił, że w części opracowań pojawiają się powtórzenia pewnych treści.

Prezentowane opracowania, każde oddzielnie, zawierają opis koncepcji rozwiązania tytułowego problemu, niekiedy krytyczną analizę stosowanych metodologii oraz polemikę z niektórymi twierdzeniami, a wreszcie rozwiązania dyskutowanych problemów.

Podziękowania

Książka zawarta w niej treść, nie mogłaby zmaterializować się bez pomocy wielu życzliwych przyjaciół, kolegów i rodziny – wszystkim im jestem niezmiernie wdzięczny za okazywane wsparcie.

Dziękuję także wszystkim innym za okazywaną życzliwość i pouczające rozmowy oraz wyrażane w korespondencji sugestie, wątpliwości i opinie. Każda z nich, także te krytyczne, mobilizowała do dalszej pracy.

Szczególnie dziękuję Panom Ryszardowi Kossowskiemu i Andrzejowi Pankowskiemu, osobom, które na kolejnych etapach moich prac okazywali zainteresowanie postępami prac i uzyskiwanymi wynikami oraz niezmiennie motywowali do dalszej pracy.

Chcę specjalnie podziękować Panu Profesorowi Januszowi Zawile-Niedźwieckiemu, który kierując Radą Centrum Informatyzacji Politechniki Warszawskiej gdy byłem jej członkiem, namówił mnie do przygotowania dwóch wykładów, które stały się fragmentami publikowanej książki.

Część pierwsza

W kręgu spełnialności

R o z d z i a ł p i e r w s z y

Dyskusja sieci logicznej dla 3SAT

Marek Malinowski

Technologie Teleinformatyczne

Dyskusję poprzedza przegląd rezultatów badań złożoności obliczeniowej prowadzonych z wykorzystaniem modelu sieci logicznych. Zaprezentowane wyniki upoważniają do stwierdzenia, że dla problemu **3SAT** istnieje jednostajna sieć wielomianowa. W szczególności pokazano: - że problem **3SAT** może być traktowany jako koniunkcja elementarnych problemów **3SAT**; - że dla dowolnego elementarnego problemu **3SAT** istnieje uniwersalna sieć logiczna o stałym rozmiarze (ilości bramek); - i ostatecznie, że dla koniunkcji elementarnych problemów **3SAT** istnieje sieć wielomianowa, i jest to sieć jednostajna.

1

Zrób dobry początek

[z teorii rozwiązywania problemów]

1.1 Wstęp – sformułowanie tytułowej tezy.

Rozstrzyganie charakteru relacji zawierania klas $P \subseteq NP$ bazuje na parametryzacji klas złożoności poprzez model obliczeń (maszyna Turinga), tryb obliczeń (determinizm i niedeterminizm), zasoby (czas i pamięć) i ograniczenia (notacja $O(.)$).

Przyjęty sposób parametryzacji stał się podstawą sformułowania twierdzenia Cooka-Levina. Określa ono pierwotny problem ***NP-zupełny***, a pośrednio wskazuje, że jeśli pokazany zostanie wielomianowy algorytm dla problemu **3SAT** (lub innego problemu ***NP-zupełnego***), to $N = NP$. Z kolei, jeśli udowodnione zostanie wykładnicze dolne ograniczenie, to $P \neq NP$.

W toku prowadzonych badań, określony na wstępie sposób parametryzacji klas złożoności był uzupełniany o inne proste modele, będące odpowiednikiem modelu maszyny Turinga. Wiadomo na przykład, że maszyny Turinga można symulować za pomocą sieci logicznych. Dowód twierdzenia wiążącego złożoność sieci logicznych ze złożonością czasową można znaleźć w książce Michaela Sipsera [2].

Przydatność modelu sieci logicznych w rozstrzyganiu problemu ***P versus NP*** wynika z bliskich związków sieci logicznych z problemem spełnialności **SAT**.

Po pierwsze, problem **SAT** został sformułowany w kategoriach rachunku zdań. Ponieważ formy zdaniowe można wyrazić

w postaci funkcji logicznych, to **SAT** może być definiowany poprzez funkcje logiczne, a te wygodnie jest reprezentować w postaci sieci logicznych.

Przydatność modelu sieci logicznych w rozstrzyganiu problemu ***P*** *versus* ***NP*** wynika z bliskich związków sieci logicznych z problemem spełnialności **SAT**.

Po pierwsze, problem **SAT** został sformułowany w kategoriach rachunku zdań. Ponieważ formy zdaniowe można wyrazić w postaci funkcji logicznych, to **SAT** może być definiowany poprzez funkcje logiczne, a te wygodnie jest reprezentować w postaci sieci logicznych.

Po drugie, dowód twierdzenia Cooka-Levina można alternatywnie przeprowadzić przy pomocy właśnie sieci logicznych [2, s. 401-408]. Wreszcie, sieci logiczne znajdują praktyczną realizację w postaci układów cyfrowych obliczających funkcje logiczne [3].

Wiadomo także, że każdy problem z klasy ***P*** ma sieć wielomianową. Niestety, ponieważ istnieją sieci wielomianowe dla problemów nierozstrzygalnych, stwierdzenie odwrotne nie jest prawdziwe. Wprowadza się więc pojęcie jednostajnych sieci wielomianowych i poprzez stwierdzenie, że jednostajna sieć wielomianowa dla reprezentowanego przez nią pewnego problemu istnieje wtedy i tylko wtedy, gdy problem ten należy do klasy ***P***, wiąże się je z obliczeniami wielomianowymi.

Powyższe stwierdzenie można traktować jako równoważne twierdzeniu Cooka-Levina. Wskazuje ono, że jeśli pokazana zostanie jednostajna sieć wielomianowa dla pewnego problemu ***NP*** - *zupełnego*, to ***P = NP***. Z kolei, jeśli udowodnione zostanie, że problemy ***NP*** - *zupełne* nie mają jedno-stajnych sieci wielomianowych, to ***P*** ≠ ***NP***.

Wbrew pewnym przesłankom (np. twierdzenie Razborowa) przemawiającym za hipotezą, że problemy ***NP*** - *zupełne* nie mają jednostajnych ani niejednostajnych sieci wielomianowych (**Hipoteza B** [1 s. 287], w dalszej części dyskutowany będzie schemat dowodu istnienia jednostajnej sieci wielomianowej dla pierwotnego problemu ***NP*** - *zupełnego* jakim jest **3SAT**.

Na niekorzyść **Hipotezy B** przemawia samo określenie problemu ***NP***, według którego weryfikacja (sprawdzenie) certy-

fikatu rozwiązania problemu odbywa się w deterministycznym czasie wielomianowym. Można więc założyć, że proces weryfikacji będzie zrealizowany przez sieć wielomianową. Ta sytuacja koresponduje z przywoływanymi dalej stwierdzeniami i określeniami dotyczącymi sieci i ich związków z problemami CIRCUIT SAT i CIRCUIT VALUE i definicją sieci jednostajnej [1] (fakt 2.7).

W kolejnych etapach wnioskowania wykorzystane zostaną, przedstawione w następnym punkcie, znane już rezultaty badań w modelu sieci logicznych.

1.2 Funkcje logiczne i sieci logiczne

Podstawowe stwierdzenia odnoszące się do modelu sieci logicznych można przedstawić w postaci faktów, których zdecydowana większość została zaczerpnięta z książki Christosa Papadimitriou [1].

Fakt 2.1: Ekwiwalentność formuł logicznych i funkcji logicznych [1, s. 95].

Każda formuła φ ze zmiennymi $x_1, \ldots, x_n$ wyraża n-argumentową funkcję logiczną f i odwrotnie, dowolna n-argumentowa funkcja logiczna f może być wyrażona jako formuła φ zawierająca zmienne $x_1, \ldots, x_n$.

Fakt 2.2: Reprezentacja funkcji logicznych przez sieci logiczne [1, s. 95-96].

Każda funkcja logiczna może być przedstawiona jako sieć logiczna, zdefiniowaną jako graf $C = (V, E)$, w którym $V = \{1, \ldots, n\}$ są bramkami grafu C, zaś E zbiorem krawędzi (i, j) przy $i < j$ (warunek acykliczności).

Składnia sieci ponadto wiąże z każdą bramką $i \in V$ rodzaj $s(i) \in \{true, false\} \cup \{x_1, \ldots, x_n\} \cup \{\vee, \wedge, \neg\}$ (gdzie $\{true, false\}$ są bramkami stałych, $\{x_1, \ldots, x_n\}$ są bramkami zmiennych oraz $\{\vee, \wedge, \neg\}$ są bramkami OR, AND, NOT odpowiednio) i charakteryzuje je stopniem wejściowym (liczba wchodzących krawędzi) równym 0,

[1] W istocie, zgodnie z definicją, sieć jednostajna ma rozstrzygać „specjalny" rodzaj certyfikatu – słowo, w którym wszystkie zmienne mają ustaloną wartość logiczną *true*.

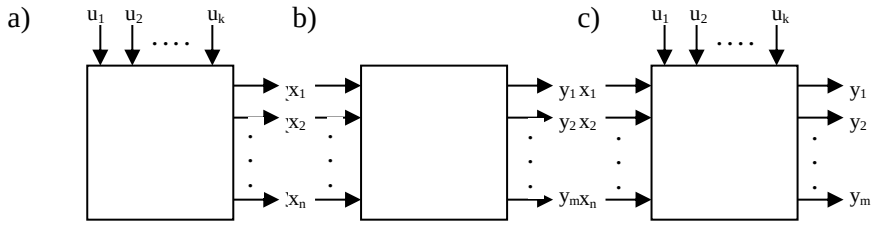
1 lub *2*. Wierzchołek, który nie ma krawędzi wychodzących, nazywany bramką wyjściową sieci oblicza funkcję logiczną.

Jeśli będziemy rozważać sieci, które mają kilka wyjść, to obliczają one kilka funkcji jednocześnie. W oparciu o tak zdefiniowaną sieć, możliwe jest konstruowanie trzech typów sieci:

- sieć bez zmiennych (rys.1 a), tj. z bramkami $V \in \{true, false\} \cup \{\vee, \wedge, \neg\}$;
- sieć ze zmiennymi i bez stałych (rys. 1 b), tj. z bramkami $V \in \{ x_1, \ldots , x_n\} \cup \{\vee, \wedge, \neg\}$;
- sieć ze zmiennymi i stałymi (rys. 1 c), tj. z bramkami $V \in \{true, false\} \cup \{ x_1, \ldots , x_n\} \cup \{\vee, \wedge, \neg\}$.

Uwaga 2.1: Każda sieć wielowyjściowa daje się sprowadzić w elementarny sposób przy pomocy sieci rozszerzającej C_{ext}, do sieci z jednym wyjściem. Na przykład, dla sieci m wyjściowej, sieć rozszerzająca C_{ext} może być zbudowana z nie więcej niż $(m-1)$ bramek AND w układzie sekwencyjnym lub równoległym.

Dopuszczając sieci wielowyjściowe, każdy z typów sieci można poprzez analogię z układami kombinacyjnymi traktować jako wielobiegunnik. Schematy ideowe wyróżnionych typów sieci przedstawia rysunek 1.1.



Rys. 1.1 Schematy ideowe typów sieci logicznych, gdzie:
X – wektor sygnałów wejściowych (zmienne $x_1, x_2, \ldots , x_n$);
U – wektor stałych $u_i \in \{true, false\}$ $i=1, 2, \ldots , k$;
Y – wektor sygnałów wyjściowych, $y_j \in \{true, false\}$ $j=1, 2, \ldots , m$
jest obliczoną wartością pewnej funkcji (niekoniecznie funkcji wszystkich n zmiennych).

Kolejne stwierdzenia wiążą sieci logiczne z problemami różnych klas złożoności obliczeniowej.

Fakt 2.3: Sieć logiczna jako problem **CIRCUIT VALUE** [1, s.97].

Sieć *C* bez zmiennych (tj. sieć z bramkami $V(i) \in \{true, false\} \cup \{\vee, \wedge, \neg\}$) (schemat ideowy rys. 1.1 a) definiuje problem **CIRCUIT VALUE** $\in$ ***P***.

Fakt 2.4: Sieć logiczna jako problem **CIRCUIT SAT** [1, s. 97].

Sieć *C* ze zmiennymi (rys. 1.1 b, c), kiedy należy stwierdzić czy istnieje takie wartościowanie zmiennych, że bramka wyjściowa zwraca wartość *true*, definiuje problem **CIRCUIT SAT**.

Pokazując [1, s.179] redukcję wielomianową **CIRCUIT SAT** do **3SAT** w pamięci *log(n)*, przyjmuje się, że **CIRCUIT SAT** należy do klasy problemów ***NP-zupełnych***.

Kolejne przytaczane stwierdzenia są wynikiem prac wiążących złożoność obliczeniową ze złożonością sieci logicznych. Złożoność sieci jest określana jako rozmiar sieci wyrażany liczbą bramek w tej sieci.

Fakt 2.5: Sieci wielomianowe [1, s 285-286].

Sieć ma rozmiar wielomianowy, jeśli istnieje rodzina sieci $C=\{C_0, C_1, \dots\}$ dla której prawdą jest po pierwsze, że rozmiar C_n jest równy co najwyżej $p(n)$ dla pewnego ustalonego wielomianu p i po drugie, że dla każdej kombinacji zmiennych $x_i \in \{0,1\}$ $i=1,2, \dots, n$ spełniających formułę reprezentowaną przez tą sieć, wartością sieci jest *true* (w kategoriach języka maszyn Turinga powiedzielibyśmy, że dla słów wejściowych, które maszyna akceptuje – wartością sieci jest *true*).

Fakt 2.6: Każdy problem z klasy ***P*** ma sieć wielomianową [1, s. 286].

Niestety, stwierdzenie odwrotne nie jest prawdziwe. Nie każda sieć wielomianowa jest reprezentacją problemu z klasy ***P***.

Uwaga 2.2: Pokazano, że istnieją problemy nierozstrzy-galne, które mają sieci wielomianowe. Sprawia to, że nie możemy wnioskować równości klas ***P = NP***, mimo że sieć *C* definiująca problem **CIRCUIT SAT** (tj. sieć ze zmiennymi, kiedy należy stwierdzić czy istnieje takie wartościowanie zmiennych, że bramka wyjściowa zwraca wartość *true*) musi mieć rozmiar wielomianowy by redukcja do **3SAT** mogła być przeprowadzona w czasie wielomianowym. A jeśli tak, to **CIRCUIT SAT** należałoby zaliczyć do

klasy ***P***. Z drugiej strony, jeśli **CIRCUIT SAT** redukuje się do **3SAT**, i tym samym w aspekcie *ZUPEŁNOŚCI* nie jest gorszy od **3SAT**, to i **3SAT** należałoby zaliczyć do ***P***.

Związanie sieci wielomianowych z obliczeniami wielomianowymi dokonuje się poprzez wprowadzenie pojęcia jednostajności sieci.

Fakt 2.7: Wielomianowe sieci jednostajne [1, s. 287].

Rodzina sieci $C=\{C_0, C_1, \dots\}$ jest jednostajna, jeśli dla zmiennych $x_i \in \{1\}$ $i=1, 2, \dots, n$ możliwe jest skonstruowanie sieci C_n w pamięci $\log n$.

Tak zdefiniowana jednostajna sieć wielomianowa prowadzi do stwierdzenia, że jednostajna rodzina sieci wielomianowych dla reprezentowanego przez nią problemu istnieje wtedy i tylko wtedy, gdy problem ten należy do klasy ***P***. Zatem, jeśli dla dowolnego problemu ***NP*** - *zupełnego* pokazana zostanie jednostajna sieć wielomianowa, to rozstrzygniecie kwestii ***P*** *versus* ***NP*** przyjmie postać ***P = NP***.

1.3 Jednostajna sieć wielomianowa 3SAT

Zaprezentowane w poprzednim punkcie stwierdzenia upoważniają do rozważenia schematu dowodu istnienia sieci jednostajnych dla **3SAT**.

Na omawiany schemat składają się kolejno następujące elementy: - poszczególne klauzule w zapisie *CNF* problemu **3SAT** stanowią elementarne problemy **3SAT**; - dla każdego elementarnego **3SAT** istnieje uniwersalna sieć o stałym rozmiarze (ilości bramek); - wszystkie uniwersalne sieci reprezentujące elementarne problemy **3SAT** (klauzule) stanowią wielowyjściową sieć wielomianową *C*, która po jej rozszerzeniu o sieć C_{ext} przekształcającą sieć *C* do sieci jednowyjściowej, rozstrzyga spełnialność formuły **3SAT** *CNF*.

Prawdziwość poszczególnych elementów powyższego schematu prowadzi do wniosku o istnieniu jednostajnej sieci wielomianowej dla problemu **3SAT**.

1.3.1 Elementarny 3SAT.

Funkcja logiczna przyporządkowuje wartościom logicznym zmiennych x wartość logiczną zmiennych y i jest opisywana formalnie za pomocą wyrażeń logicznych. Wyrażenia logiczne mogą być zapisywane w wielu różnorodnych i równoważnych sobie postaciach.

Wyrażenie logiczne definiujemy następująco:

- 0, 1, x_i i $\neg x_i$ są wyrażeniami logicznymi oraz jeśli φ_1 i φ_2 są wyrażeniami logicznymi, to $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$ są wyrażeniami logicznymi;
- wyrażenie logiczne może mieć tylko postać opisaną wyżej, ale zamiast x_i mogą wystąpić inne zmienne logiczne.

Znamy wszystkie wyrażenia logiczne opisujące funkcje dwóch zmiennych (jest ich 16). Przy ich pomocy możemy opisać każdą funkcję n zmiennych. Jeśli funkcję i zmiennych oznaczymy przez f^i, to $f^2(f^i(x_1, x_2, \dots, x_i), x_{i+1}) = f^{i+1}$.

Tak więc na przykład $f^3 = f^2(f^2(x_1, x_2), x_3)$, a ponieważ w wyrażeniu mogą wystąpić dowolne zmienne, więc ogólnie można zapisać $f^3 = f^2(f^2(x_r, x_s), x_t)$.

Jeśli φ będzie wyrażeniem logicznym opisującym funkcję n zmiennych w postaci koniunkcji m funkcji trzech zmiennych:

$$\varphi = f_1^3(x_r, x_s, x_t) \wedge f_2^3(x_r, x_s, x_t) \wedge \dots \wedge f_m^3(x_r, x_s, x_t) \qquad (1)$$

gdzie: $x_r, x_s, x_t \in \{x_1, x_2, \dots, x_n\}$ oraz $r \in \{1, 2, \dots n\text{-}2\}$,
$s \in (r+1, r+2, \dots, n\text{-}1\}$ i $t \in \{s+1, s+2, \dots n\}$, tj. $r < s < t$,

to opuszczając oznaczenia argumentów poszczególnych funkcji można zapisać, że φ jest funkcją m zmiennych, których rolę pełnią funkcje f_i^3, $i=1, 2, \dots, m$.

$$\varphi = f^m(f_1^3, f_2^3, \dots, f_m^3) = f^2(f^{m-1}(f_1^3, f_2^3, \dots, f_{m-1}^3), f_m^3)$$

Rozwijając ostatnią postać wyrażenia dalej uzyskamy ostatecznie:

$$\varphi = f^2(f^2(\dots f^2(f^2(f_1^3, f_2^3), f_3^3)\dots, f_{m-1}^3), f_m^3) \qquad (2)$$

Uzyskana postać wyrażenia pokazuje, że można $f_1^3, f_2^3, \dots f_m^3$ traktować niezależnie jako argumenty kolejnych $(m\text{-}1)$ dwu argumentowych funkcji f^2 (wszystkie są funkcjami iloczynu logicznego realizowanymi przez bramki AND), począwszy od najgłębiej zagnieżdżonej.

W kategoriach sieci logicznych, odpowiada to sytuacji, w której $f_1^3, f_2^3, \ldots f_m^3$ stanowią niezależne fragmenty w sieci z *m* wyjściami, tak że *j-te* wyjście rozstrzyga spełnialność *j-tej* funkcji $f_j^3(x_r, x_s, x_t)$.

Fakt 3.1 Elementarny **3SAT**

Jeśli przyjmiemy, że funkcja φ opisana przez wyrażenie *(1)* ma postać normalną iloczynu *CNF*, odpowiadającą definicji problemu **3SAT**, to funkcje f_i^3 *i=1, 2, … , m* reprezentowane są przez klauzule trzech różnych zmiennych, wtedy każda z ośmiu możliwych postaci klauzul trzech zmiennych może być traktowana jako elementarny **3SAT**.

Tabela 1.1 Zestawienie elementarnych alternatyw i koniunkcji funkcji 3 zmiennych.

Wartościowanie zmiennych x_3	x_2	x_1	Elementarne alternatywy (klauzule)	Elementarne koniunkcje (implikanty)
0	*0*	*0*	$x_3 \vee x_2 \vee x_1$	$\bar{x}_3 \wedge \bar{x}_2 \wedge \bar{x}_1$
0	*0*	*1*	$x_3 \vee x_2 \vee \bar{x}_1$	$\bar{x}_3 \wedge \bar{x}_2 \wedge x_1$
0	*1*	*0*	$x_3 \vee \bar{x}_2 \vee x_1$	$\bar{x}_3 \wedge x_2 \wedge \bar{x}_1$
0	*1*	*1*	$x_3 \vee \bar{x}_2 \vee \bar{x}_1$	$\bar{x}_3 \wedge x_2 \wedge x_1$
1	*0*	*0*	$\bar{x}_3 \vee x_2 \vee x_1$	$x_3 \wedge \bar{x}_2 \wedge \bar{x}_1$
1	*0*	*1*	$\bar{x}_3 \vee x_2 \vee \bar{x}_1$	$x_3 \wedge \bar{x}_2 \wedge x_1$
1	*1*	*0*	$\bar{x}_3 \vee \bar{x}_2 \vee x_1$	$x_3 \wedge x_2 \wedge \bar{x}_1$
1	*1*	*1*	$\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1$	$x_3 \wedge x_2 \wedge x_1$

W rachunku zdań, klauzula definiowana jest jako alternatywa literałów określających wartościowanie zmiennych logicznych, zaś koniunkcja literałów określana jest mianem implikantu. Ekwiwalentnymi określeniami tych pojęć w kategoriach funkcji logicznych są elementarna alternatywa i elementarna koniunkcja odpowiednio. Zestawienie możliwych do zdefiniowania ośmiu klauzul (elementarnych alternatyw) i ośmiu implikantów (elementarnych koniunkcji) funkcji trzech zmiennych zawarto w tabeli 1.1.

1.3.2 Uniwersalna sieć logiczna elementarnego 3SAT

Dla ośmiu różnych klauzul trzech zmiennych można zdefiniować 255 funkcji (bez funkcji stałej ***TRUE***) wyrażonych koniunkcją kombinacji klauzul po jednej, po dwie itd. aż do ośmiu z ośmiu i każdej przyporządkować numer. Każda tak

zdefiniowana funkcja będzie zapisana w kanonicznej normalnej postaci koniunkcyjnej (*CNF*).

Wykorzystując dualny sposób definiowania funkcji przy pomocy implikantów otrzymamy 255 funkcji (bez funkcji stałej ***FALSE***) wyrażonych alternatywą kombinacji implikantów po jednym, po dwa itd. aż do ośmiu z ośmiu i każdej przyporządkować numer, to każda tak zdefiniowana funkcja będzie zapisana w kanonicznej normalnej postaci dysjunkcyjnej (*DNF*).

Jeśli konstruowanie sieci logicznej reprezentującej funkcje trzech zmiennych (nawet wtedy, gdy ograniczymy się do rozpatrywania 8 funkcji opisywanych przy pomocy pojedynczych klauzul) realizowane będzie wprost w oparciu o wyrażenia klauzul, to mimo, że nie będą skomplikowane, będą różnić się rozmiarem.

Na przykład, dla funkcji $f^3(x_r, x_s, x_t) = (x_r \vee x_s \vee x_t)$ wystarczą 2 bramki OR, ale dla funkcji $f^3(x_r, x_s, x_t) = (\neg x_r \vee \neg x_s \vee \neg x_t)$, sieć oprócz 2 bramek OR będzie zawierała jeszcze 3 bramki NOT.

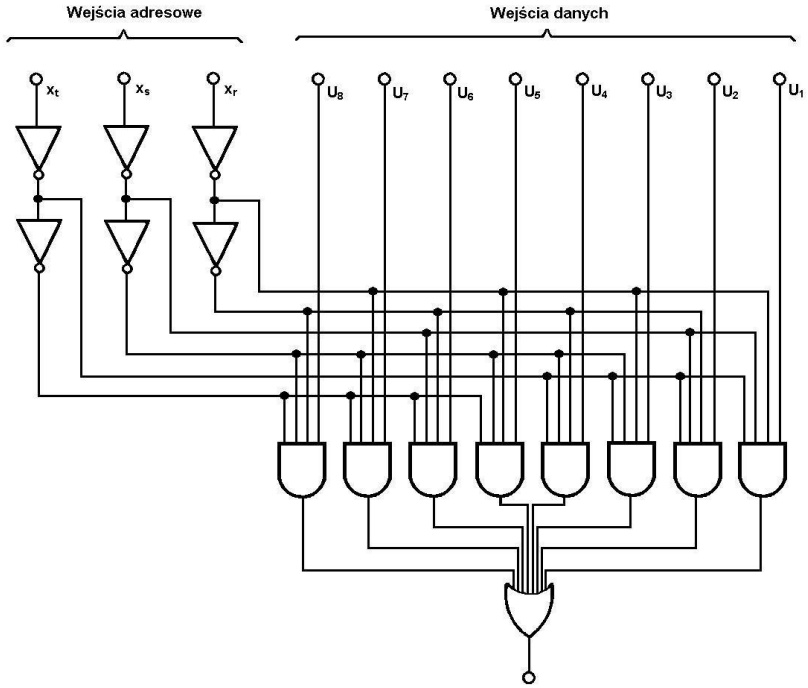
Zróżnicowany rozmiar sieci elementarnego **3SAT** *CNF* nie jest sprzyjającą okolicznością dla zadania konstruowania sieci logicznej dla całego **3SAT** *CNF*. Jednak kosztem zwiększenia rozmiaru sieci logicznej, możliwe jest operowanie uniwersalną siecią logiczną, która może reprezentować dowolną funkcję logiczną trzech zmiennych.

Taką uniwersalna siecią dla przypadku elementarnego **3SAT** *CNF*, jest sieć logiczna reprezentująca funkcję stałą ***TRUE***, a dla przypadku elementarnego **3SAT** *DNF*, sieć reprezentująca funkcję stałą ***FALSE***. W obu przypadkach, podstawą konstruowania sieci jest układ multipleksera 8/1. Rodzaj bramek i sposób ich połączenia w sieci logicznej określa schemat układu prezentowany na rysunku 1.2.

Sygnały wejściowe zmiennych x_r, x_s, x_t , będą podawane na wejścia adresowe układu. Sygnały stałych (wektory $U=[u_1, u_2, \ldots, u_8]$, binarnie interpretowane jako numer funkcji) będą podawane na wejścia danych multipleksera. Sygnał wyjściowy z bramki OR określa tablica stanów (tabela 1.2), z której jednoznacznie wynika, że sygnał na wyjściu układu odpowiada sygnałowi podawanemu

na wejście danych określone przez aktualny stan wejść adresowych.

Rys. 1.2 Schemat układu multipleksera 8/1
Tabela 1.2 Tablica stanów multipleksera 8/1



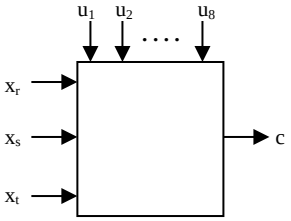
wejścia adresowe	x_r	0	1	0	1	0	1	0	1
	x_s	0	0	1	1	0	0	1	1
	x_t	0	0	0	0	1	1	1	1
wyjście		u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8

Pokazany na rysunku 1.2 schemat multipleksera 8/1, pozwala określić rozmiar odpowiadającej jemu sieci logicznej. Każdą występującą w schemacie czterowejściową bramkę AND można zastąpić trzema dwuwejściowymi bramkami AND. Z kolei ośmiowejściową bramkę OR można zastąpić siedmioma bramkami OR. Ostatecznie rozmiar sieci logicznej odwzorowującej układ multipleksera będzie mieć rozmiar łącznie

48 bramek, w tym 3 bramki wejściowe zmiennych, 8 bramek wejściowych stałych, 24 dwuwejściowych bramek AND, 7 dwuwejściowych bramek OR i 6 bramek NOT.

W oparciu o przeprowadzoną w tym punkcie dyskusję sieci logicznej elementarnego **3SAT**, możemy sformułować stwierdzenie.

Fakt 3.2: Każda funkcja logiczna *f* trzech różnych zmiennych x_r, x_s, x_t, opisana wyrażeniem w jednej z dwóch kanonicznych normalnych postaci koniunkcyjnej (*CNF*) lub dysjunkcyjnej (*DNF*), ma uniwersalną sieć logiczną o stałym rozmiarze *k* bramek. Przyjmuje ona postać wielobiegunnika, przedstawioną na rysunku 3, co odpowiada schematowi ideowemu z rys. 1.1 c.



Rys. 1.3 Schemat ideowy uniwersalnej sieci logicznej funkcji trzech zmiennych.

1.3.3 Konstrukcja sieci 3SAT

Przy rozpatrywaniu wszystkich możliwych 255 funkcji zdefiniowanych przez wyrażenia postaci *CNF*, musielibyśmy operować 255 różnymi wektorami stałych $U=[u_1, u_2, \dots, u_8]$.

Ponieważ przyjęliśmy, że jako elementarne **3SAT** będziemy traktować funkcje opisywane przez wyrażenia mające postać pojedynczych klauzul, to możemy ograniczyć się do rozpatrywania ośmiu funkcji definiowanych przez osiem różnych klauzul i wtedy przy konstruowaniu sieci operować będziemy ośmioma wektorami stałych *U*, które jednoznacznie są określane przez postaci klauzul.

$$U = \begin{cases} [0,1,1,1,1,1,1,1] & \text{jeśli} \quad x_3 \vee x_2 \vee x_1 \\ [2,0,1,1,1,1,1,1] & \text{jeśli} \quad x_3 \vee x_2 \vee \bar{x_1} \end{cases}$$

[2, 1, 0, 1, 1, 1, 1, 1]	jeśli	$x_3 \vee \bar{x_2} \vee x_1$
[2, 1, 1, 0, 1, 1, 1, 1]	jeśli	$x_3 \vee \bar{x_2} \vee \bar{x_1}$
[2, 1, 1, 1, 0, 1, 1, 1]	jeśli	$\bar{x_3} \vee x_2 \vee x_1$
[2, 1, 1, 1, 1, 0, 1, 1]	jeśli	$\bar{x_3} \vee x_2 \vee \bar{x_1}$
[2, 1, 1, 1, 1, 1, 0, 1]	jeśli	$\bar{x_3} \vee \bar{x_2} \vee x_1$
[2, 1, 1, 1, 1, 1, 1, 0]	jeśli	$\bar{x_3} \vee \bar{x_2} \vee \bar{x_1}$

W poszczególnych wektorach występują stałe *false* na kolejnych jego pozycjach. Dla określenia numeru pozycji, na których występują one w wektorze *U*, wystarczy by negatywne literały w zapisie klauzuli interpretować jako cyfry „1" liczby binarnej. Na przykład $(\bar{x_3} \vee \bar{x_2} \vee x_1) \equiv 110_2 = 6_{10}$.

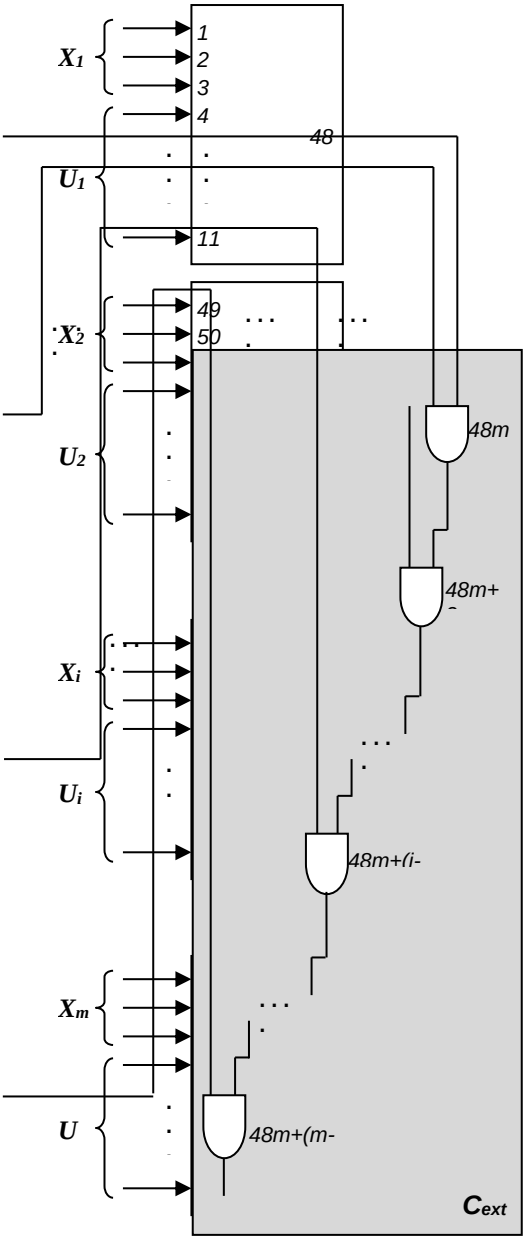
Uwaga 3.1: Należy wyraźnie zaznaczyć, że na określenie postaci wektorów *U* ma wprost wpływ jedynie układ negatywnych literałów w zapisie klauzuli i nie ma wpływu to z jaką kombinacją trzech zmiennych mamy do czynienia. Mogą to być dowolne kombinacje trzech zmiennych dopuszczane przez formułę *(1)* definiującą **3SAT** *CNF*.

Uwaga 3.2: Jeśli na wszystkie wejścia stałych będą podane sygnały *true* (u_i=*1)*, to rozpatrywany schemat multipleksera realizuje funkcję ***TRUE***. Z kolei, jeśli na wszystkie wejścia stałych będą podane sygnały *false* (u_i=*0)*, to multiplekser realizuje funkcję ***FALSE***.

Wobec spostrzeżeń poczynionych w *uwagach 3.1 i 3.2* przyjmiemy, że sieć realizująca funkcję ***TRUE*** dla trzech zmiennych, będzie podstawą konstruowania sieci logicznej całego **3SAT** *CNF*. Ponadto, zapisując wyrażenie *(2)* w postaci operatorowej, stosując operator prefiksowy AND, uzyskamy postać jednoznacznie określającą sposób konstruowania sieci rozszerzającej C_{ext}.

$$\mathbf{3SAT}\ CNF = AND(AND(\ ...\ AND(AND(f_1^3, f_2^3), f_3^3)..., f_{m-1}^3), f_m^3)$$

W efekcie, konstrukcja sieci **3SAT** *CNF* sprowadzi się do *m-krotnego* powielenia sieci elementarnego **3SAT** reprezentowanego przez sieć realizującą funkcję ***TRUE*** i ustalenia stałej *false* na stosownej pozycji wektora U_i *(i=1, 2, … , m)*.



Rys. 1.4 Schemat ideowy sieci logicznej *3SAT CNF*

Zakładając identyczny sposób numerowania bramek wejść adresowych i wejść danych w każdym reprezentującym element-tarny **3SAT** segmencie sieci oraz wykorzystując oznaczenie wielobiegunnika (rys. 1.3), otrzymaną sieć można przedstawić w postaci schematu ideowego (rys. 1.4).

Rozmiar tak skonstruowanej sieci względem *m* ilości klauzul w zapisie **3SAT** *CNF* jest liniowy i wynosi *(49·m-1)* bramek.

Dla określenia rozmiaru sieci względem *n* ilości zmiennych, trzeba uwzględnić przypadek **3SAT** *CNF*, w którym mogą występie klauzule wszystkich kombinacji po *3* z *n* zmiennych. Takich kombinacji jest *(n-2)·(n-1)·n/6*, a więc ich ilość wyraża się wielomianem trzeciego stopnia.

Uwzględniając fakt, że dla każdej kombinacji po *3* z *n* zmiennych istnieje *8* różnych klauzul, więc w najgorszym przypadku ilość klauzul w formule **3SAT** *CNF* będzie wynosić co najwyżej $4/3 \cdot n \cdot (n-1) \cdot (n-2)$.

Uwaga 3.3: W istocie najgorszy przypadek można ograniczyć do sytuacji, w której w formule **3SAT** *CNF* występuje maksymalnie *7* klauzul tej samej kombinacji zmiennych. Możemy tak przyjąć, bo wystarczy by tylko dla jednej kombinacji trzech zmiennych w zapisie *CNF* wystąpiło *8* klauzul, to cała formuła nie jest spełnialna (*CNF* ośmiu klauzul definiuje funkcję stałą ***FALSE***).

Ostatecznie rozmiar sieci logicznej **3SAT** *CNF* skonstruowanej w oparciu o układ multipleksera 8/1 nie jest większy niż $49 \cdot 4/3 \cdot n \cdot (n-1) \cdot (n-2)$ bramek.

Zatem sieć ma rozmiar wielomianowy. Dodatkowo jest to sieć, którą łatwo wykorzystać jako sieć jednostajną. W tym celu wystarczy na wszystkie wejścia adresowe multiplekserów X_i $(i=1,2, \ldots, m)$ podać stałe *true*. Pamiętamy, że na wejścia adresowe multiplekserów podawane są zmienne $x_r, x_s, x_t \in \{x_1, x_2, \ldots, x_n\}$ dla $r \in \{1, 2, \ldots n-2\}$, $s \in (r+1, r+2, \ldots, n-1\}$ i $t \in \{s+1, s+2, \ldots n\}$, tj. $r < s < t$, występujące w poszczególnych klauzulach, zaś podanie wartościowań $x_i = true$ jest wymogiem definicji sieci jednostajnej. Dla wektora wartościowań zmiennych $[x_1, x_2, \ldots, x_n]$ określającego inny dowolny sposób wartościowania tych zmiennych, sieć pozostaje siecią rozstrzygającą.

Pozostaje uzasadnić, że konstrukcję sieci można wykonać w pamięci *log n*. Proces powielania układu elementarnego **3SAT** (multiplekser 8/1) wymaga jednej zmiennej do pamiętania *m* ilości klauzul i jednego licznika do indeksacji kolejnych sieci elementarnych **3SAT** oraz ich wejść. Potrzebny jest także licznik do obliczania numeru wejścia stałych, na które należy podać stałą *false*. Połączenia odpowiednich bramek wymagają tylko bezpośrednich operacji na indeksach i stąd są łatwe do wykonania w pamięci logarytmicznej.

1.4 Podsumowanie

Sformułowana we wstępie teza o istnieniu jednostajnych sieci wielomianowych dla problemu **3SAT** *CNF* prowadzi wprost do stwierdzenia równości klas problemów ***P*** i ***NP***.

Wykazanie prawdziwości stawianej tezy stanowi zasadniczą część prezentowanej pracy. Pokazano, że wykorzystując dualizm funkcji logicznych, możliwe jest traktowanie koniunkcji klauzul występujących w zapisie **3SAT** *CNF* jako koniunkcji elementarnych funkcji trzech zmiennych. Pokazano, że wykorzystując elementy teorii układów kombinacyjnych, możliwe jest zdefiniowanie uniwersalnej sieci logicznej dla dowolnej elementarnej funkcji logicznej trzech zmiennych. Taka uniwersalna sieć charakteryzuje się stałym rozmiarem wyrażonym przez *48* bramek logicznych AND, OR i NOT. W szczególności dotyczy to każdej z *8* możliwych postaci klauzul trzech zmiennych. Pokazano, że rolę takiej uniwersalnej sieci logicznej może spełniać sieć skonstruowana w oparciu o układ multipleksera 8/1.

Ostatecznie pokazano, że sieć logiczna pełnego **3SAT** *CNF*, w najgorszym przypadku charakteryzuje się rozmiarem wyrażanym przez $O(n^3)$. Ponadto pokazano, że taką sieć można skonstruować w pamięci logarytmicznej oraz, że może to być sieć jednostajna.

Wybór problemu **3SAT** dla pokazania istnienia jednostajnej sieci wielomianowej nie jest przypadkowy[2]. Przede wszystkim problem spełnialności ***SAT*** jest problemem pierwotnym dla rozważań w kategoriach *ZUPEŁPOŚCI*. Wszystkie wyniki w odniesieniu do niego, dotyczą więc zarówno do problemów ***NP*** *- zupełnych* (problem **3SAT**) jak również problemów ***P*** *- zupełnych* (problem **2SAT**) Poza tym **3SAT** został wybrany dlatego, by w ocenie i interpretacji wyniku pracy uniknąć ***NP*** *– trudności*, podobnej do tej z jaką mieliśmy przy ocenie i weryfikacji rozwiązania problemu **KNAPSACK**.

Poprawności opisanej i dyskutowanej konstrukcji sieci (i całej rodziny sieci) nie można podważyć, chyba że: - po pierwsze zakwestionowana zostanie definicja rodziny sieci jednostajnej, w tym wymóg by każda sieć z rodziny sieci była siecią skonstruowaną w pamięci $\log n$ i by była rozstrzygającą dla słów wejściowych, takich że $x_i = 1$ dla każdego $i = 1, \ldots, n$, gdzie n jest tak

[2] W kolejnych pracach przygotowywanych do publikacji pokazano konstrukcję jednostajnej sieci wielomianowej dla problemu **MULT**(n)

jak w [1, s.285][3] długością słowa określającego sposób wartościowania zmiennych występujących w formule **3SAT** odwzorowywanej przez sieć logiczną, a nie tak jak w [1, s.43][4] długością słowa opisującego problem w kategoriach maszyny Turinga; - po drugie zakwestionowane zostaną równoważność modelu maszyny Turinga z modelem sieci logicznych i wskazany zostanie błąd w przywołanym we wstępie twierdzeniu wiążącym złożoność sieci logicznych ze złożonością czasową [2, s. 401-405],

W istocie opisana konstrukcja sieci logicznej C_n jest niczym innym jak redukcją wielomianową formuły **3SAT** *CNF* $\varphi (x_1, x_2, \ldots , x_n)$ do problemu **CIRCUIT VALUE**, realizowaną podobnie jak pokazana w [1, s. 184-186] redukcja dowolnego języka $L \in P$ do **CIRCUIT VALUE**, sprowadzająca się do powielania identycznych elementarnych sieci i następnie łączenia odpowiednich bramek wejściowych i wyjściowych.

Pomijając to, że strukturalnie sieć wielomianowa dla **3SAT** nie różni się od sieci dla **2SAT** (w sieci **2SAT**, miejsce multiplekserów 8/1 w sieci **3SAT** zajmują multipleksery 4/1), warto zwrócić uwagę na jeszcze jeden aspekt faktu stwierdzenia możliwości skonstruowania jednostajnej sieci wielomianowej dla **3SAT**.

Otóż, to że dla **3SAT** można skonstruować sieć logiczną o wielomianowej ilości bramek, otwiera drogę do „odważnych" prób skonstruowania sprzętowej implementacji algorytmu rozwiązania **2SAT** i **3SAT** oraz do „jeszcze bardziej odważnej" próby opracowania algorytmów w polilogarytmicznym czasie równoległym przy całkowitej pracy wielomianowej. Uwieńczenie tych „jeszcze bardziej odważnych" prób sukcesem rozstrzygałoby jeszcze jedną interesującą kwestię ***NC*** *(klasa Nick'a) versus* ***P***.

Pokazanie istnienia sieci jednostajnej dla **3SAT** *CNF* jest równoważne dla stwierdzenia, że dla **3SAT** *CNF* istnieje algorytm wielomianowy. Skonstruowanie takiego algorytmu, niezależnie

---

[3] „ … wiemy, że sieć logiczna o *n* zmiennych wejściowych może obliczać dowolną funkcję logiczną *n* zmiennych. Równoważnie możemy myśleć, że sieć akceptuje pewne słowa długości *n* z $\{ 0, 1 \}^*$ i odrzuca pozostałe. W tej sytuacji słowa $x = x_1 \ldots x_n \in \{ 0, 1 \}^*$ traktujemy jako wartościowanie zmiennych wejściowych sieci, … „

[4] „By rozwiązać taki problem używając maszyny Turinga, musimy najpierw zdecydować, w jaki sposób będziemy zapisywać (reprezentować) przykład za pomocą słowa."

od jego praktycznego znaczenia, może być potraktowane jako element weryfikacji przedstawionym wyników.

BIBLIOGRAFIA

[1] Papadimitriou Christos, Złożoność obliczeniowa, Warszawa, WNT, 2007

[2] Sipser Michael, Wprowadzenie do teorii obliczeń, Warszawa, WNT, 2009

[3] Traczyk Wiesław, Układy cyfrowe. Podstawy teoretyczne i metody syntezy, Warszawa, WNT, 1986

R o z d z i a ł d r u g i

Model referencyjny i algorytmy wielomianowe kSAT

Marek Malinowski

Technologie Teleinformatyczne

We wstępie sformułowano szereg pytań określających kwestie warte pogłębionej analizy w odniesieniu do problemu spełnialności **SAT**. Przedstawiono także przesłanki przemawiające za ukierunkowaniem prac w stronę badania złożoności strukturalnej **KSAT**.
W kolejnych punktach opisano, dualny względem klasycznego zapisu *SAT CNF*, sposób definiowania problemów **kSAT** przy pomocy koniunkcji funkcji logicznych. Przedyskutowano nowy sposób parametryzacji, uwzględniający powiązania występujące między zmiennymi.
Zasadniczą część pracy stanowi opis modelu referencyjnego **kSAT** i jego elementarnych struktur. Pokazano, że zarówno model jak i jego elementy charakteryzują się podobieństwem strukturalnym i samoistnie definiują warianty problemów **kSAT**, w tym nieokreślony przypadek najgorszy.

2

Znajdź punkt odniesienia

[z teorii rozwiązywania problemów]

2.1 Wstęp – geneza, cel i zakres pracy

Sformułowana w oparciu o twierdzenie Cook'a-Levina kwestia ***P*** *versus* ***NP*** pozostaje wciąż nie rozstrzygnięta. Ustalenie relacji między zdefiniowanymi w oparciu o modele maszyn Turinga klasami problemów ***P*** i ***NP*** stanowi kluczowe zagadnienie teorii złożoności obliczeniowej. Rozstrzygnięcie tej kwestii ma nie tylko teoretyczne, ale przede wszystkim bardzo praktyczne znaczenie, szczególnie w odniesieniu do kryptografii cyfrowej [5, s. 169], [6, s. 241], [7, s. 463-464].

Z twierdzenia Cook'a-Levina wynika, że jeśli dla jakiegokolwiek problemu zakwalifikowanego do klasy ***NP*** - *zupełnych* (wiemy że musi to być problem *silnie* ***NP****-zupełny*) zostanie opracowany algorytm wielomianowy to ***P*** *v* ***NP*** ma rozstrzygnięcie postaci ***P*** **=** ***NP***. Z drugiej strony, rozstrzygnięcie postaci ***P*** **≠** ***NP*** wymaga dowodu, że nie istnieją algorytmy wielomianowe dla któregokolwiek problemu *silnie* ***NP****-zupełnego*.

Według sondażu z roku 2002 przeprowadzonego przez Williama Gasarch'a [2], zdecydowana większość badaczy przewidywała rozstrzygnięcie w postaci ***P*** **≠** ***NP***. Taki pogląd znajduje odzwierciedlenie w wynikach prowadzonych badań. Większość prac koncentruje się na próbach rozstrzygnięcia w postaci ***P*** **≠** ***NP***, wykorzystując do tego wykraczające poza podstawowy model maszyny Turinga różnorodne rozszerzenia w postaci modeli maszyn alternujących, wyroczni, izomorfizmu, algorytmów losowych czy algorytmów aproksymacyjnych.

Niestety, znaczna część uzyskiwanych rezultatów pozostaje równoważną postacią twierdzenia Cook'a-Levina. Warto jednak zauważyć, że w przypadku niektórych twierdzeń pojawiają się elementy konstruktywnego dowodu równości ***P = NP***. Jako przykład można wskazać opis algorytmu wielomianowego dla **SAT** [4, s. 355] lub algorytm dla problemu funkcyjnego **FSAT** [4. s. 246].

Przeprowadzony przegląd rezultatów badań problemu **SAT** skłania do postawienia szeregu pytań w tym:

- czy dostatecznie uwzględniany jest przejawiający się w wielu płaszczyznach dualizm problemu;
- jakie są przyczyny braku redukcji **SAT** do **2SAT**, o którym wiadomo, że w wersji decyzyjnej należy do klasy ***P***;
- czy dostatecznie uwzględniono obserwowane w formułach **SAT** zróżnicowane powiązań między zmiennymi i kolejność ich występowania;
- czy stosowany system klasyfikowania problemów **kSAT** pozwala określić przypadki najgorsze;
- dlaczego brakuje pogłębionych analiz możliwości rozwiązania połączenia dwóch i więcej wielomianowych wariantów problemu **SAT**, np. **HORNSAT** i **2SAT**, mimo że takie połączenie **MOŻE** [1], ale nie **MUSI** doprowadzić do powstania problemu ***NP**-zupełnego*.

W szczególności, kwestia zasygnalizowana w ostatnim pytaniu, może być potraktowana jako weryfikacja wyniku stwierdzającego istnienie jednostajnej wielomianowej sieci logicznej **3SAT**, co rozstrzygało równość klas ***P*** i ***NP***.

W dyskusji pozostałych kwestii, punktem wyjścia było przyjęcie poglądu, że istotnym ograniczeniem w badaniu problemu **SAT**i jego wariantów jest jego postrzeganie głównie przez pryzmat rachunku zdań i pozostawanie w kręgu zapisu *SAT CNF* koniunkcji klauzul [2].

---

[1] W swojej książce Papadimitriou [4, s. 224] formułuje zadanie 9.5.6: „Pokaż, że szczególny przypadek problemu **SAT**, w którym każda klauzula jest *albo hornowska, albo zawiera dokładnie dwa literały*, jest ***NP**-zupełny*. (Innymi słowy, połączenie dwóch wielomianowych wariantów problemu **SAT** może doprowadzić do powstania problemu ***NP**-zupełnego*).

[2] „Na ograniczenie dopuszczalnych postaci formuły (koniunktywnej postaci normalnej) pozwalamy sobie z dwóch powodów. Po pierwsze, wiemy, ze każdą

Takie ujęcie i stosowana terminologia w pewnym stopniu ogranicza możliwości definiowania nowych konstrukcji, nowych pojęć i operacji (w szczególności postać klauzulowa zapisu uniemożliwia zdefiniowanie wprost przy ich użyciu logicznej funkcji stałej ***TRUE***).

Praktyka wskazuje, że w przypadku wielu problemów właśnie wprowadzenie nowych konstrukcji i operacji przy konstruowaniu algorytmów umożliwiło zlikwidowanie charakte-ryzującej je wcześniej luki algorytmicznej.

Najczęściej ulepszanie algorytmów polegało na wprowadzeniu nowych elementów strukturalnych i ich uporządkowaniu wg określonego kryterium w regularnej strukturze, jaką jest np. tablica (przykład problemu powłoki wypukłej [3, s. 159-162]). Sama operacja uporządkowania jest w takich strukturach w najgorszym wypadku wielomianowa, zaś inne takie jak wyszukiwanie w strukturach uporządkowanych wykonywane są w czasie logarytmicznym. Poczynione spostrzeżenia stały się przesłanką podjęcia próby podobnego potrak-towania problemu **kSAT**.

W opracowaniu zawarto niezbędne konstrukcje i modele przydatne w dyskusji algorytmów dla **3SAT**. Przede wszystkim skoncentrowano się na takim ujęciu **3SAT**, które umożliwia jego postrzeganie w kategoriach struktury tablicy trójkątnej górnej i/lub dolnej.

Wszystkie operacje dla tak postrzeganych struktur, w tym operacje porządkowania, wyszukiwania, zamiany i przeglądania sekwencyjnego (także z nawrotami) elementów tablic, w strukturach tablicowych są operacjami wielomianowymi. Stanowi to mocne przesłanki na rzecz istnienia sekwencyjnych algorytmów wielomianowych dla **3SAT**. Obszerne omówienie algorytmów sortowania różnych struktur danych można znaleźć w książce Niklausa Wirth'a *„Agorytmy + struktury danych = programy"* [8].

Dla charakteryzowania elementów strukturalnych we wprowadzonych konstrukcjach użyto aparatu pojęciowego – rząd i rozpiętość schematu - stosowanego w algorytmach genetycznych

formułę można tak przedstawić. Po drugie, wygląda na to, że ta szczególna postać spełnialności oddaje złożoność całego problemu" [4, s. 93].

[1]. Uzyskano dzięki temu możliwość wygodnego i prostego parametryzowania rozpatrywanych przypadków.

Dodatkowo, wykorzystując wzajemną ekwiwalentność formuł rachunku zdań i funkcji logicznych, sięgnięto po metody i mechanizmy wypracowane dla problemów syntezy układów kombinacyjnych[3].

W opisie wprowadzonych i dyskutowanych konstrukcji został użyty elementarny aparat formalny. Jego użycie powoduje, że ujęcie tematu w znacznym stopniu odbiega od zwykle stosowanych w publikacjach formalizmów i wykorzystywanych abstrakcji.

Właśnie te względy sprawiają, że omówieniu terminologii poświęcono odrębny punkt opracowania. Opisano w nim także sposób interpretowania funkcji logicznych dwóch i trzech zmiennych w ujęciu dyskutowanych treści.

2.2 Terminologia – dualizm funkcji

Problem spełnialności **SAT** został sformułowany w formalizmie rachunku zdań, ale każdą formułę rachunku zdań można traktować jako wyrażenie pewnej funkcji logicznej. Tym samym pojęcia rachunku zdań mają swoje ekwiwalentne określenia w formalizmie funkcji logicznych. W odniesieniu do pewnych pojęć stosowane jest wspólne nazewnictwo.

Zmienne występujące w zapisie formuł **SAT** oznaczane są zależnie od ich wartościowania symbolem x_i (wartość logiczna *true* – odpowiednik literału pozytywnego) lub $\bar{x}_i$ (wartość logiczna *false* – odpowiednik literału negatywnego), gdzie dolny indeks oznacza numer zmiennej. Przyjęty sposób oznaczania zmiennych

[3] Przypomnijmy, układem kombinacyjnym nazywamy automat bez pamięci zrealizowany w postaci układu elektronicznego. Funkcjonowanie układu kombinacyjnego określają elementy składowe i struktura połączeń między nimi. Ogólnie, układ kombinacyjny można przedstawić jako urządzenie o *n* wejściach i *m* wyjściach. Algorytm funkcjonowania można określić wykorzystując aparat algebry logiki, zapisując zależności między sygnałami wejściowymi w postaci funkcji logicznych, przy czym wykorzystywane są w tym celu normalne kanoniczne postaci alternatywne lub koniunkcyjne.

uwalnia z konieczności posługiwania się dalej pojęciem literałów. W odróżnieniu od stałych logicznych *true* i *false*, funkcje stałe oznaczane będą ***TRUE*** i ***FALSE***.

W rachunku zdań, klauzula definiowana jest jako alternatywa zmiennych, zaś koniunkcja zmiennych określana jest mianem implikantu. Ekwiwalentnymi określeniami tych pojęć w kategoriach funkcji logicznych są terminy - elementarna alternatywa i elementarna koniunkcja odpowiednio.

Pojęcie funkcji stosowane w opisie oznacza reprezentację funkcji logicznej w postaci dysjunkcyjnej kanonicznej postaci normalnej (*DNF)*, tj. w postaci alternatywy elementarnych koniunkcji zmiennych. Postać *DNF* funkcji logicznej jest dualną postacią względem postaci koniunkcyjnej kanonicznej postaci normalnej (*CNF)* funkcji, tj. koniunkcji elementarnych alternatyw zmiennych. Dla funkcji trzech zmiennych logicznych takich funkcji jest 256, zaś dla funkcji dwóch zmiennych jest ich 16.

Klauzula *Horna* oznacza alternatywę zmiennych, z których co najwyżej jedna jest pozytywna.

Klauzula *coHorna* oznacza alternatywę zmiennych, z których co najwyżej jedna jest negatywna.

W przypadku funkcji logicznej trzech zmiennych, jej zmienne mogą być wartościowane na osiem sposobów. Każdy z możliwych sposobów wartościowania zmiennych można opisać poprzez stosowną elementarną alternatywę lub odpowiadającą jej elementarną koniunkcję (dla formy zdaniowej trzech zmiennych będą to odpowiednio klauzule i implikanty) – ich zestawienie zawarto w tabeli 2.1.

W zestawieniu występują 4 klauzule *Horna* i 4 klauzule *coHorna*. Zbiory klauzul *Horna* i *coHorna* w tym przypadku (i także w przypadku funkcji więcej niż trzech zmiennych) są rozłączne.

Tabela 2.1 Zestawienie elementarnych alternatyw (klauzul) i elementarnych koniunkcji (implikantów) dla funkcji 3 zmiennych.

Wartościowanie zmiennych			Elementarne alternatywy (klauzule)	Elementarne koniunkcje (implikanty)
x_3	x_2	x_1		
0	0	0	$x_3 \vee x_2 \vee x_1$	$\bar{x}_3 \wedge \bar{x}_2 \wedge \bar{x}_1$
0	0	1	$x_3 \vee x_2 \vee \bar{x}_1$	$\bar{x}_3 \wedge \bar{x}_2 \wedge x_1$
0	1	0	$x_3 \vee \bar{x}_2 \vee x_1$	$\bar{x}_3 \wedge x_2 \wedge \bar{x}_1$
0	1	1	$x_3 \vee \bar{x}_2 \vee \bar{x}_1$	$\bar{x}_3 \wedge x_2 \wedge x_1$
1	0	0	$\bar{x}_3 \vee x_2 \vee x_1$	$x_3 \wedge \bar{x}_2 \wedge \bar{x}_1$
1	0	1	$\bar{x}_3 \vee x_2 \vee \bar{x}_1$	$x_3 \wedge \bar{x}_2 \wedge x_1$
1	1	0	$\bar{x}_3 \vee \bar{x}_2 \vee x_1$	$x_3 \wedge x_2 \wedge \bar{x}_1$
1	1	1	$\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1$	$x_3 \wedge x_2 \wedge x_1$

Dla 8 różnych alternatyw (klauzul) można zdefiniować 255 funkcji (bez funkcji stałej ***TRUE***) i każda z nich będzie zapisana w kanonicznej postaci koniunkcyjnej (*CNF*). Podobnie dla 8 różnych elementarnych koniunkcji (implikantów) można zdefiniować 255 funkcji (bez funkcji stałej ***FALSE***) i wtedy każda z nich będzie zapisana w kanonicznej postaci dysjunkcyjnej (*DNF*). W obu przypadkach zapisu, funkcje można ponure-rować. Przyjęty sposób numerowania funkcji trzech zmiennych przedstawia tabela 2.2 (każdą z 256 możliwych kombinacji wartości funkcji interpretujemy jako liczbę binarną).

Tabela 2.2 Numeracja funkcji trzech zmiennych.

x_3	x_2	x_1	f_0	f_1	f_2	...	f_{15}	f_{16}	f_{17}	...	f_{30}	f_{31}	f_{32}	...	f_{254}	f_{255}
0	0	0	0	1	0	...	1	0	1	...	0	1	0	...	0	1
0	0	1	0	0	1	...	1	0	0	...	1	1	0	...	1	1
0	1	0	0	0	0	...	1	0	0	...	1	1	0	...	1	1
0	1	1	0	0	0	...	1	0	0	...	1	1	0	...	1	1
1	0	0	0	0	0	...	0	1	1	...	0	1	1	...	1	1
1	0	1	0	0	0	...	0	0	0	...	0	0	0	...	1	1
1	1	0	0	0	0	...	0	0	0	...	0	0	0	...	1	1
1	1	1	0	0	0	...	0	0	0	...	0	0	0	...	1	1

W przypadku funkcji logicznej dwóch zmiennych operujemy czterema elementarnymi alternatywami i czterema elementarnymi koniunkcjami – ich zestawienie zawarto w tabeli 2.3.

Tabela 2.3 Zestawienie elementarnych alternatyw (klauzul) i elementarnych koniunkcji (implikantów) dla funkcji 2 zmiennych.

Wartościowanie zmiennych		Elementarne alternatywy (klauzule)	Elementarne koniunkcje (implikanty)
x_2	x_1		
0	*0*	$x_2 \vee x_1$	$\bar{x}_2 \wedge \bar{x}_1$
0	*1*	$x_2 \vee \bar{x}_1$	$\bar{x}_2 \wedge x_1$
1	*0*	$\bar{x}_2 \vee x_1$	$x_2 \wedge \bar{x}_1$
1	*1*	$\bar{x}_2 \vee \bar{x}_1$	$x_2 \wedge x_1$

W zestawieniu występują 3 klauzule *Horna* i 3 klauzule *coHorna* (np. $x_2 \vee \bar{x}_1$ można traktować zarówno jako klauzulę *Horna* jak i *coHorna*). Zbiory klauzul *Horna* i *coHorna* w tym przypadku nie są rozłączne (mają część wspólną).

Dla 4 różnych elementarnych alternatyw (klauzul) można zdefiniować 15 funkcji (bez funkcji stałej ***TRUE***) i wtedy każda funkcja będzie zapisana w koniunkcyjnej kanonicznej postaci normalnej (*CNF*). Podobnie dla 4 różnych elementarnych koniunkcji (implikantów) można zdefiniować 15 funkcji (bez funkcji stałej ***FALSE***) i wtedy każda funkcja będzie zapisana w dysjunkcyjnej kanonicznej postaci normalnej (*DNF*). Podobnie jak dla funkcji trzech zmiennych, w obu przypadkach zapisu, funkcje można ponumerować. Przyjęty sposób numerowania funkcji dwóch zmiennych przedstawia tabela 2.4.

Tabela 2.4 Numeracja funkcji dwóch zmiennych.

x_2	x_1	h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}
0	*0*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*
0	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*
1	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*
1	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*

2.3 Model referencyjny

2.3.1 Podstawowe pojęcia i oznaczenia

Przy ocenie złożoności obliczeniowej dowolnego problemu, podstawą takiej oceny jest najczęściej analiza najgorszego przypadku. Aktualny status problemu **SAT** charakteryzuje się wieloma znaczącymi wynikami. Odnoszą się one jednak do wyodrębnionych szczególnych przypadków **SAT**. Odczuwalny jest brak uogólnionego modelu, pozwalającego na prowadzenie

analizy złożoności różnych wariantów **SAT**, łącznie dla wszystkich możliwych przypadków, w tym wyodrębnionych przypadków szczególnych.

Podstawą klasyfikacji problemów **SAT** i jego wariantów jest zapis *SAT CNF*. W kontekście prowadzonych rozważań, należy wyraźnie zaznaczyć, że zapis *SAT CNF* jest koniunkcją klauzul. Formuła *kSAT CNF* oznacza koniunkcję klauzul *k* zmiennych, przy czym w formule mogą wystąpić różne klauzule tych samych zmiennych.

W przypadku *3SAT CNF* mogą to być klauzule spośród czterech różnych klauzul *Horna*, bądź spośród czterech klauzul *coHorna*, co w problemie *3SAT CNF* wyczerpuje zbiór możliwych klauzul (patrz tabela 2.1). Dla tej samej kombinacji zmiennych w *3SAT CNF* mogą wystąpić bądź tylko pojedyncze klauzule (spośród 8 możliwych), bądź dwie, trzy aż do siedmiu różnych klauzul (pomijamy przypadek wszystkich ośmiu możliwych klauzul, bo wtedy reprezentują one funkcję stałą ***FALSE***).

Funkcje trzech zmiennych są oznaczane symbolem *f*, zaś funkcje dwóch zmiennych symbolem *h*. Wszędzie tam, gdzie będzie to konieczne dla wskazania, o którą funkcję spośród możliwych chodzi, używany będzie dolny indeks. Pominięcie dolnego indeksu oznacza, że może to być dowolna funkcja ze zbioru wszystkich możliwych.

Na potrzeby dalszych rozważań zdefiniujemy pojęcie formuły modelu referencyjnego oraz stosowane oznaczenia.

Formuła modelu referencyjnego – oznaczana *FullkSAT* - problemu **kSAT** dla *n* ponumerowanych zmiennych, gdzie *k* jest ilością zmiennych w klauzulach jego postaci *CNF*, konstruowana jest z uwzględnieniem wszystkich możliwych kombinacji $\binom{n}{k}$ zmiennych. Każda z możliwych kombinacji zmiennych rozpatrywana jest w kategoriach funkcji logicznych *k* zmiennych.

Symbole *m* i *M* oznaczają odpowiednio ilość klauzul problemu **kSAT** w postaci *CNF* oraz ilość funkcji w formule modelu referencyjnego *FullkSAT*.

Funkcja diagonalna d_i jest dowolną funkcją kolejnych *k* zmiennych problemu **kSAT**, począwszy od *i-tej* zmiennej, np. dla **3SAT** $d_i \equiv f(x_i, x_{i+1}, x_{i+2})$.

Rząd funkcji oznacza ilość elementarnych koniunkcji w jej postaci *DNF (*dysjunkcyjna kanoniczna postać normalna).

Konstruowane według ustalonych kryteriów podformuły modeli referencyjnych dla **3SAT** oraz **2SAT**, są oznaczane symbolami Ω i Θ odpowiednio. Dla rozróżnienia podformuł konstruowanych według innych kryteriów niż przyjęte dla podformuł Ω i Θ, będą używane oznaczenia symbolami φ i F *oraz* ψ i G odpowiednio dla **3SAT** oraz **2SAT.**

2.3.2 Dyskusja parametryzacji problemów kSAT

Wykorzystywane do scharakteryzowania przedstawionego w postaci zapisu kSAT CNF problemu, parametry w postaci m - ilości klauzul, czy stosunek m/n - ilości klauzul do ilości zmiennych, pozwalają jedynie względnie porównywać ze sobą poszczególne przypadki problemów. Wydaje się, że są one niewystarczające do określenia najgorszego przypadku, czy chociażby wskazania zbioru, w którym się on zawiera.

Także bazująca na strukturze klauzul klasyfikacja problemów **KSAT** jest obarczona mankamentami. W zasadzie wyodrębnia tylko szczególne przypadki, na przykład **NEASAT** (kiedy żądamy, by w żadnej klauzuli wszystkie trzy zmienne nie były wartościowane jednakowo – ani *true*, ani *false*), **ONE-IN-TREE SAT** (kiedy wymagamy, by dokładnie jeden literał w każdej klauzuli był prawdziwy, zamiast przynajmniej jednego) czy **3SAT** dla wyrażeń, w których każda zmienna może pojawić się najwyżej trzy razy, a każdy literał najwyżej dwa razy.

Do takiego sformułowania upoważniają przykłady dwóch problemów **3SAT** dyskutowane poniżej. Pokazują one, że ilość klauzul w formule nie determinuje złożoności problemu.

Przykład 2.1 Pierwszy przykład można skonstruować w taki sposób, że dla dwóch występujących w nim kombinacji 4 zmiennych, w formule *3SAT CNF*, każda taka kombinacja zmiennych będzie opisana przez siedem różnych klauzul.

$$\begin{aligned} \textit{3SAT CNF} = {} & (x_3 \vee x_2 \vee x_1) \wedge (x_3 \vee x_2 \vee \overline{x}_1) \wedge (x_3 \vee \overline{x}_2 \vee x_1) \wedge (x_3 \vee \overline{x}_2 \vee \overline{x}_1) \wedge \\ & (\overline{x}_3 \vee x_2 \vee x_1) \wedge (\overline{x}_3 \vee x_2 \vee \overline{x}_1) \wedge (\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_1) \wedge \\ & (x_4 \vee x_3 \vee x_1) \wedge (x_4 \vee x_3 \vee \overline{x}_1) \wedge (x_4 \vee \overline{x}_3 \vee x_1) \wedge (x_4 \vee \overline{x}_3 \vee \overline{x}_1) \wedge \\ & (\overline{x}_4 \vee \overline{x}_3 \vee x_1) \wedge (\overline{x}_4 \vee \overline{x}_3 \vee x_1) \wedge (\overline{x}_4 \vee \overline{x}_3 \vee \overline{x}_1) \end{aligned}$$

Mamy tu sytuację, w której w zapisie formuły *3SAT CNF* dla danego zbioru kombinacji *n* zmiennych wystąpi maksymalna m_{max} ilość klauzul. Rozszerzenie formuły o jakąkolwiek nową, różną klauzulę, ale dla występującej w formule kombinacji zmiennych (np. klauzuli $(x_3 \vee x_2 \vee x_1)$ lub klauzuli $(x_4 \vee x_3 \vee x_1)$), spowoduje wystąpienie ośmiu różnych klauzul tej samej kombinacji zmiennych i tym samym problem stanie się trywialny. Formuła będzie niespełniana, bo osiem różnych klauzul dla tych samych zmiennych definiuje funkcję stałą ***FALSE***. Także bez rozszerzenia formuły o kolejną klauzulę problem jest prosty. Wykorzystując fakt, że dualnym wobec kanonicznej koniunkcyjnej postaci normalnej funkcji (a tą jest koniunkcja elementarnych alternatyw) jest kanoniczna dysjunkcyjna postać normalna funkcji (alternatywa elementarnych koniunkcji zmiennych), każdą występującą
w *3SAT CNF* koniunkcję siedmiu klauzul tych samych zmiennych (klauzulę traktujemy jako elementarną alternatywę zmiennych) zastępujemy jedną elementarną koniunkcją tych zmiennych.

$$3SAT\ CNF = (x_3 \wedge x_2 \wedge x_1) \wedge (\overline{x}_4 \wedge x_3 \wedge x_1) = \overline{x}_4 \wedge x_3 \wedge x_2 \wedge x_1$$

Zabieg ten sprawia, że formuła *3SAT CNF* przyjmuje postać koniunkcji zmiennych i jeśli nie wystąpią w niej jednocześnie dwa przeciwne sobie wartościowania tej samej zmiennej (w innej kombinacji zmiennych), to reprezentuje ona wektor rozwiązania (wartościowania zmiennych spełniających formułę).

Przykład 2.2 Drugi przykład, jako przeciwwagę pierwszego, można skonstruować w taki sposób, że dla występujących w nim kombinacji (w przykładach wszystkie kombinacje z 4 po 3) zmiennych w formule *3SAT CNF* klauzul, każda taka kombinacja zmiennych opisana jest przez dokładnie jedną klauzulę. W tej sytuacji, w formule *3SAT CNF* dla danego zbioru kombinacji zmiennych wystąpi minimalna m_{min} ilość klauzul, a poprzez fakt, że wśród tych klauzul mogą wystąpić klauzule *Horna* i/lub *coHorna*, będziemy mieli trzy przypadki:

- wszystkie klauzule są klauzulami *Horna* (**HORN3SAT**):

$$3SAT\ CNF = (\overline{x}_3 \wedge \overline{x}_2 \wedge x_1) \wedge (\overline{x}_4 \wedge \overline{x}_3 \wedge \overline{x}_1) \wedge (\overline{x}_4 \wedge x_2 \wedge \overline{x}_1) \wedge (x_4 \wedge \overline{x}_3 \wedge \overline{x}_2)$$

- wszystkie klauzule są klauzulami *coHorna* (**coHORN3SAT**):

$$3SAT\ CNF = (\overline{x}_3 \vee x_2 \vee x_1) \wedge (x_4 \vee x_3 \vee x_1) \wedge (\overline{x}_4 \vee x_2 \vee x_1) \wedge (x_4 \vee x_3 \vee \overline{x}_2)$$

- część klauzul jest klauzulami *Horna*, a pozostałe klauzulami *coHorna* (mieszany – nazwijmy go **MIXHORN3SAT**):

$$3SAT\ CNF = (\bar{x}_3 \wedge \bar{x}_2 \wedge x_1) \wedge (\bar{x}_4 \wedge \bar{x}_3 \wedge \bar{x}_1) \wedge (\bar{x}_4 \vee x_2 \vee x_1) \wedge (x_4 \vee x_3 \vee \bar{x}_2).$$

Dwa pierwsze przypadki są proste. Wiemy, że **HORN3SAT** jest rozwiązywalny w czasie wielomianowym. Więcej, można elementarnie pokazać, że **HORN3SAT** jest zawsze spełnialny (wszystkie zmienne wartościowane negatywnie – wartość logiczna *false*).

Podobnie jest z **coHORN3SAT** – jest zawsze spełnialny (wszystkie zmienne wartościowane pozytywnie – wartość logiczna *true*) i podobnie jak **HORN3SAT** jest rozwiązywany w czasie wielomianowym. Wystarczy zauważyć, że przez podstawienie $x_i = \bar{y}_i$ oraz $\bar{x}_i = y_i$, klauzule *coHorna* stają się klauzulami *Horna*. Na przykład **coHORN3SAT** określony przez formułę $3SAT\ CNF = (\bar{x}_3 \vee x_2 \vee x_1) \wedge (x_4 \vee x_3 \vee x_1)$ po podstawieniach przyjmie postać wyrażenia $(y_3 \vee y_2 \vee y_1) \wedge (y_4 \vee y_3 \vee \bar{y}_1)$, które dla nowych zmiennych może być rozpatrywany jako problem **HORN3SAT**. Po uzyskaniu wektora rozwiązań, wystarczy każdą występującą w nim wartość y_i zanegować.

O ile w pierwszym i drugim przypadku rozpatrywanego przykładu (**HORN3SAT** i **coHORN3SAT**), dla ich scharakteryzowania wielkości m oraz m/n, mimo iż są względne, mogą być wystarczające dla oceny trudności problemu, o tyle w trzecim przypadku (**MIXHORN3SAT**) użycie tych parametrów nie wnosi nic istotnego dla takiej oceny. Jeśli w formule *3SAT CNF* w miejsce klauzuli określonych zmiennych pojawi się inna klauzula tych samych zmiennych niż ta, która występuje w formule, parametry m i m/n nie ulegną zmianie (poprzez te parametry nie zauważona zostanie zmiana problemu).

Ponadto pozostając przy pierwszym przypadku, rozszerzenie formuły o dodatkową klauzulę *Horna* tych samych zmiennych co już występująca w *3SAT CNF* klauzula, powoduje, że problem pozostaje problemem **HORN3SAT**. Jeśli natomiast byłaby to klauzula *coHorn*, problem przestaje być problemem **HORN3SAT**. Z kolei, w przypadku **HORN3SAT** i **coHORN3SAT**, usunięcie z for-muły dowolnej klauzuli nie gwarantuje obniżenia rzędu problemu (zmniejszenia ilości zmiennych). Możliwa jest przecież sytuacja, że w części, bądź we wszystkich występujących w formule *3SAT CNF* klauzulach jest ta sama zmienna.

Rozwiązanie trzeciego przypadku byłoby proste, gdyby zmienne występujące w klauzulach *Horna* nie występowały w klauzulach *coHorna*. Wtedy w istocie mielibyśmy do czynienia z dwoma niezależnymi problemami, z których każdy potrafimy rozwiązać. Trzeba jednak założyć, że zmienne występujące w klauzulach *Horna* będą występowały także w klauzulach *coHorna*.

Jednak, nawet w sytuacji braku niezależności **HORN3SAT** i **coHORN3SAT** nie jesteśmy bezradni. Ponieważ oba problemy są zawsze spełniane, to można próbować wykorzystać, wskazany już wcześniej [4, s. 246] algorytm rozwiązania problemu funkcyjnego **FSAT**. Po znalezieniu wektorów wartościowań spełniających odrębnie formuły dla **HORN3SAT** i **coHORN3SAT**, możemy następnie je odpowiednio złożyć. W tym kontekście znaczenia nabiera pytanie dotyczące pogłębionych analiz możliwości rozwiązania połączenia dwóch wielomianowych wariantów problemu **SAT**.

Dyskusję obu przykładów można pogłębić, rozpatrując szereg innych możliwych działań (rozszerzenie o pewne nowe klauzule czy usuwanie pewnych klauzul) w stosunku do pierwotnych problemów. Mimo tego, że cała powyższa dyskusja może wydawać się oczywista i trywialna, może być wykorzystana do sformułowania kilku wniosków i postulatów.

Przede wszystkim dla prowadzenia usystematyzowanych analiz wariantów problemu **KSAT** potrzebny jest pewien „stabilny" bezwzględny model odniesienia, ponieważ:

- dowolna operacja dodania, usunięcia lub zmiany chociażby jednego członu w formule *3SAT CNF* może radykalnie zmienić jego status, a parametr *m* określający ilość klauzul pozwala tylko względnie porównywać problemy między sobą;
- w zapisie *3SAT CNF* można wykorzystywać dualizm reprezentowania funkcji *k* zmiennych, bo w pewnych sytuacjach (przykład pierwszy), bardziej wygodnym sposobem opisania problemu **KSAT** jest operowanie koniunkcyjną normalną postacią kanoniczną *CNF* (koniunkcja elementarnych alternatyw – klauzul), innym razem wygodniejszą staje się dysjunkcyjna normalna postać kanoniczna *DNF* (alternatywa elementarnych koniunkcji – implikantów);
- obie kanoniczne postaci normalne funkcji – dysjunkcyjna i koniunkcyjna – „atomizują" zapis problemu;

- porzucenie „atomizującego" zapisu na rzecz bardziej ogólnego, stwarza szansę spojrzenia na problem w skali „makro".

Tak sformułowane wnioski, pozwalają określić podstawowe postulowane własności modelu odniesienia (dalej modelu referencyjnego). Przede wszystkim:

(i) powinien obejmować wszystkie możliwe warianty problemów **KSAT**, zarówno wcześniej wyodrębnione warianty szczególne, jak i nie określony przypadek najgorszy;
(ii) ilość członów wyrażenia opisującego model powinna być stała;
(iii) wszystkie człony powinny być strukturalnie jednorodne.

Wzorcem takiego modelu referencyjnego może być problem **K-MAXGSAT** zdefiniowany w [4, s. 319], jako bardziej ogólny od problemu **MAXSAT**. Problem **K-MAXGSAT** polega na znalezieniu wartościowania spełniającego jak największą liczbę formuł z danego zbioru formuł logicznych $\Phi = \{ \varphi_1, \varphi_2, \ldots \varphi_m\}$ mających wspólnie n zmiennych. Dopuszcza się by formuły ze zbioru mogły być dowolną formułą logiczną zawierającą najwyżej k spośród n zmiennych dla ustalonego $k > 0$.

Przytoczona definicja problemu **K-MAXGSAT** spełnia (ii) i (iii) postulaty sformułowane wobec własności modelu referencyjnego **KSAT**. Ponieważ dopuszczalne są dowolne formuły, to ich rolę mogą spełniać funkcje logiczne k zmiennych, przy czym niekoniecznie muszą to być formuły dokładnie k zmiennych spośród n zmiennych. To rozluźnienie reguł można wykorzystać w sposób umożliwiający dla określonego **KSAT** uzyskanie w modelu referencyjnym stałej ilości funkcji. Tak więc pozostaje dalej zadbać, by spełniany był postulat (i).

2.3.3 Funkcje w zapisie formuł *kSAT CNF*

W formułach *kSAT CNF* mamy do czynienia zawsze z koniunkcjami klauzul k zmiennych (dla *3SAT CNF* oraz *2SAT CNF* będą to *trzy* oraz *dwie* zmienne odpowiednio). Każdą koniunkcję klauzul tych samych zmiennych można postrzegać w kategoriach funkcji logicznych. Każda z funkcji logicznych k zmiennych, może być jednoznacznie identyfikowana związanym z nią numerem (dla *trzech* i *dwóch* zmiennych - patrz tabele

2.2 i 2.4). Znając przyjęty sposób numerowania tych funkcji, bardzo łatwo będzie można przedstawić każdą z nich w jednej z dual-nych postaci – koniunkcyjnej kanonicznej normalnej postaci (koniunkcja elementarnych alternatyw), bądź w dysjunkcyjnej kanonicznej normalnej postaci (alternatywa elementarnych koniunkcji).

Model referencyjny można rozpatrywać w kategoriach klauzul, ale kierując się omówionymi wcześniej przesłankami, rozpatrywany będzie w kategoriach funkcji. Nie tracąc nic z ogólności, dalej ograniczymy się do rozpatrywania problemów **3SAT** i **2SAT**.

Ponadto, ponieważ parametr *m* ilość klauzul w formułach *3SAT CNF* oraz *2SAT CNF* nie jest w stanie charakteryzować powiązań występujących między zmiennymi, do oceny struktury tych powiązań, wprowadzimy nowe parametry.

Dla zapewnienia stałej, niezmiennej ilości funkcji, model referencyjny problemów **3SAT** i **2SAT** dla *n* ponumerowanych zmiennych występujących w klauzulach jego postaci *3SAT CNF* oraz *2SAT CNF*, konstruowany będzie z uwzględnieniem wszystkich możliwych kombinacji $\binom{n}{3}$ zmiennych dla **3SAT** i $\binom{n}{2}$ dla **2SAT**. Tak więc w obu modelach referencyjnych będzie wyrażać się wielomianami i będzie równa $M = n\cdot(n-1)\cdot(n-2)/6$ dla **3SAT** oraz $M = n\cdot(n-1)/2$ dla **2SAT** i zawsze nie będzie większa od ilości klauzul[4].

Każda funkcja w modelu referencyjnym problemu **3SAT**, prócz numeru i rzędu (ilości elementarnych koniunkcji), będzie charakteryzowana uporządkowaną trójką *(r, s, t)* numerów zmiennych oraz rozpiętościami δ_1 i δ_2 numerów tych zmiennych.

Rozpiętości δ_1 i δ_2 są dla każdej uporządkowanej trójki numerów *(r, s, t)* zmiennych, różnicami numerów drugiej i pierwszej zmiennej oraz numerów trzeciej i drugiej zmiennej odpowiednio. Rozpiętość traktowana jest jako miara powiązań między zmiennymi funkcji. Dodatkowo δ_c rozpiętość skrajnych zmiennych funkcji jest określona przez sumę δ_1 i δ_2. Na funkcje nie

[4] Ilość *m* klauzul w modelu referencyjnym **3SAT** byłaby zmienna - co najmniej równa *M* (jeśli były by to pojedyncze klauzule dla każdej kombinacji zmiennych), a maksymalnie $m_{max} = 7\cdot M$ (jeśli dla każdej możliwej kombinacji zmiennych w formule *CNF* występowałoby po siedem różnych klauzul – pomijamy przypadek ośmiu różnych klauzul, bo wtedy definiują one funkcję stałą ***FALSE***).

są nakładane żadne dodatkowe ograniczenia (co do jej numeru i pośrednio rzędu).

Podobnie jak w modelu referencyjnym problemu **3SAT**, każda funkcja w modelu referencyjnym problemu **2SAT**, prócz numeru i rzędu, będzie charakteryzowana uporządkowaną dwójką *(r, s)* numerów zmiennych oraz rozpiętością δ numerów tych zmiennych. Także w tym przypadku na funkcje nie są nakładane żadne dodatkowe ograniczenia.

W przypadku problemów **KSAT** dla $k>3$, charakterystyka modeli referencyjnych będzie wyglądała podobnie. Każda funkcja $\binom{n}{k}$ kombinacji zmiennych problemu **KSAT**, może być sparametryzowana przez uporządkowany *k-elementowy* zbiór numerów jej zmiennych oraz przez *k-1* rozpiętości $\delta_1, \delta_2, \ldots \delta_{k-1}$.

W ten sposób, model referencyjny staje się modelem uogólnionym. Cechuje go stały (zależny tylko od *n* ilości zmiennych) parametr charakteryzujący wielomianowy rozmiar problemu. Obejmuje on wszystkie możliwe przypadki problemu **KSAT** dla $k\geq 1$, zarówno wcześniej wyodrębnione przypadki szczególne jak i nie określony wprost przypadek najgorszy. Jest tak, bo jeśli na przykład w konkretnym przypadku formuły *3SAT CNF* nie występują w niej klauzule pewnych kombinacji zmiennych, to w formule modelu referencyjnym ***Full3SAT*** można to odnotować przez użycie funkcji stałej ***TRUE*** (funkcja stała ***TRUE*** trzech zmiennych jest alternatywą ośmiu elementarnych koniunkcji i według przyjętej numeracji jest oznaczana f_{255}). Możliwość użycia funkcji stałej ***TRUE*** powoduje, że **KAŻDY** dowolny problem **3SAT** może być rozpatrywany w kategoriach modelu referencyjnego.

Przy przyjętych założeniach, model referencyjny staje się wygodnym obiektem dla prowadzenia różnorodnych analiz. Traktując model referencyjny jako zbiór elementów – funkcji o ustalonych własnościach (numer, rząd, rozpiętości zmiennych), analizy modelu mogą być prowadzone pod kątem np. ustalenia charakterystycznych struktur jego elementów (funkcji), czy też określenia wpływu pewnych operacji na elementy (funkcje) modelu.

Wyjawienie charakterystycznych struktur elementów (funkcji), przekładać się będzie na możliwość wyodrębnienia podformuł w

zapisie *3SAT CNF*, i dalej ewentualnego podziału problemu na podproblemy.

2.3.4 Strukturyzacja formuł w modelu referencyjnym FullkSAT

Wprowadzone w modelu referencyjnym problemu **3SAT** parametry charakteryzujące funkcje, pozwalają wykonać operację uporządkowania ich według zróżnicowanych kryteriów.

I tak, przyjmując parametry – numer pierwszej zmiennej funkcji w modelu referencyjnym, rozpiętości δ_1 oraz δ_2 za klucze uporządkowania funkcji modelu referencyjnego, otrzymamy formułę *Full3SAT* w postaci:

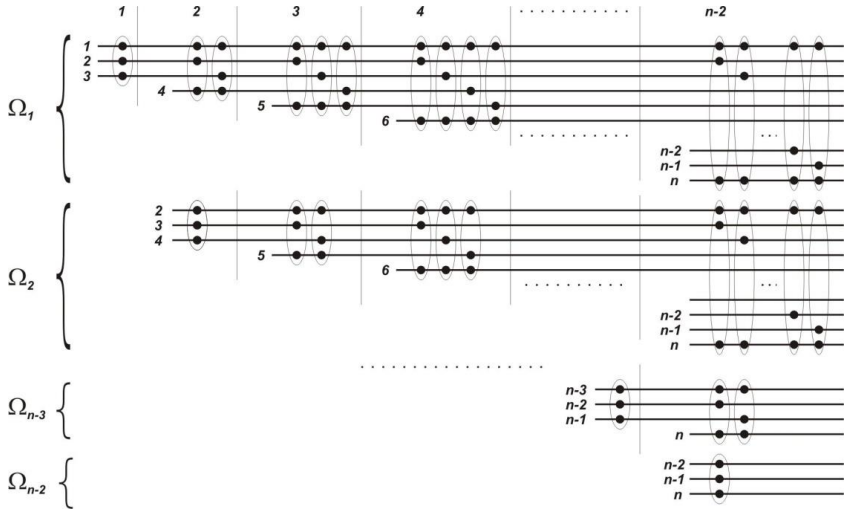
$$
\begin{aligned}
Full3SAT = \quad & f(x_1, x_2, x_3) \\
& \wedge f(x_1, x_2, x_4) \wedge f(x_1, x_3, x_4) \\
& \wedge f(x_1, x_2, x_5) \wedge f(x_1, x_3, x_5) \wedge f(x_1, x_4, x_5) \\
& \quad \cdots\cdots\cdots\cdots\cdots \\
& \wedge f(x_1, x_2, x_n) \wedge f(x_1, x_3, x_n) \wedge \ldots \wedge f(x_1, x_{n-1}, x_n) \qquad \} \Omega_1 \\
& \wedge f(x_2, x_3, x_4) \\
& \wedge f(x_2, x_3, x_5) \wedge f(x_2, x_4, x_5) \\
& \wedge f(x_2, x_3, x_6) \wedge f(x_2, x_4, x_6) \wedge f(x_2, x_5, x_6) \\
& \quad \cdots\cdots\cdots\cdots\cdots \\
& \wedge f(x_2, x_3, x_n) \wedge f(x_2, x_4, x_n) \wedge \ldots \wedge f(x_1, x_{n-1}, x_n) \qquad \} \Omega_2 \\
& \quad \cdots\cdots\cdots\cdots\cdots \\
& \quad \cdots\cdots\cdots\cdots\cdots \\
& \wedge f(x_i, x_{i+1}, x_{i+2}) \\
& \wedge f(x_i, x_{i+1}, x_{i+3}) \wedge f(x_i, x_{i+2}, x_{i+3}) \\
& \wedge f(x_i, x_{i+1}, x_{i+4}) \wedge f(x_i, x_{i+2}, x_{i+4}) \wedge f(x_i, x_{i+3}, x_{i+4}) \\
& \quad \cdots\cdots\cdots\cdots\cdots \\
& \wedge f(x_i, x_{i+1}, x_n) \wedge f(x_i, x_{i+2}, x_n) \wedge \ldots \wedge f(x_i, x_{n-1}, x_n) \qquad \} \Omega_i \\
& \quad \cdots\cdots\cdots\cdots\cdots \\
& \wedge f(x_{n-3}, x_{n-2}, x_{n-1}) \\
& \wedge f(x_{n-3}, x_{n-2}, x_n) \wedge f(x_{n-3}, x_{n-1}, x_n) \qquad \} \Omega_{n-3} \\
& \wedge f(x_{n-2}, x_{n-1}, x_n) \qquad \} \Omega_{n-2}
\end{aligned}
$$

Analizując powyższe wyrażenie, widać, że uporządkowanie funkcji w formule modelu referencyjnego *Full3SAT* wg przyjętego kryterium – numer pierwszej zmiennej, rozpiętość δ_1 i rozpiętość δ_2 - daje możliwość wyodrębnienia podformuł Ω_i, gdzie $i=1, 2, \ldots, n-2$ oznacza numer pierwszej zmiennej we wszystkich funkcjach *i*-

tej podformuły (takich podformuł będzie maksymalnie *n-2*) i będzie znana ich ilość M_i w każdej z tych podformuł (będzie ich maksymalnie *(n-i)(n-i-1)/2*). W szczególności podformuła $\boldsymbol{\Omega}_{n-2}$ będzie określona przez jedną funkcję ostatnich trzech zmiennych. Tak więc formuła modelu referencyjnego **3SAT** może być zapisana w postaci:

$$Full3SAT = \boldsymbol{\Omega}_1 \wedge \boldsymbol{\Omega}_2 \wedge \ldots \boldsymbol{\Omega}_{i-1} \wedge \boldsymbol{\Omega}_i \wedge \boldsymbol{\Omega}_{i+1} \wedge \ldots \wedge \boldsymbol{\Omega}_{n-3} \wedge \boldsymbol{\Omega}_{n-2}$$

Każda z podformuł $\boldsymbol{\Omega}_i$, może być zobrazowana graficznie, co ilustruje rysunek 2.1.



Rys. 2.1 Graficzny obraz formuły $Full3SAT = \boldsymbol{\Omega}_1 \wedge \boldsymbol{\Omega}_2 \wedge \ldots \wedge \boldsymbol{\Omega}_i \wedge \ldots \wedge \boldsymbol{\Omega}_{n-2}$

Na rysunku, numerowane linie poziome służą do odwzorowania odpowiednich zmiennych. Dla oznaczenia funkcji posłużono się owalnym kształtem. Rozmieszczone wewnątrz owalnego kształtu punkty wskazują na zmienne występujące w danej funkcji. Dodatkowo pionowymi liniami rozdzielono pewne charakterystyczne grupy funkcji występujące w poszczególnych podformułach.

Oznaczając przez $\boldsymbol{F}_{i,j}$ dla $i = 1, \ldots, n\text{-}2$ oraz $j = 1, \ldots, n\text{-}2$ podformuły tworzone przez koniunkcje funkcje $\boldsymbol{f(x_i, x_s, x_{j+2})}$, gdzie $i < s < j+2$, tj. na przykład $\boldsymbol{F}_{1,3} = f(x_1, x_2, x_5) \wedge f(x_1, x_3, x_5) \wedge f(x_1, x_4, x_5)$, to każda z formuł $\boldsymbol{\Omega}_i$ może być przedstawiona w postaci:

$$\boldsymbol{\Omega}_i = \boldsymbol{F}_{i,i} \wedge \boldsymbol{F}_{i,i+1} \wedge \ldots \wedge \boldsymbol{F}_{i,j} \wedge \ldots \wedge \boldsymbol{F}_{i,n-2}$$

i wykorzystując wprowadzone oznaczenia, formułę *Full3SAT* można zapisać następująco:

$$\begin{aligned} Full3SAT = \; & \boldsymbol{F}_{1,1} \wedge \boldsymbol{F}_{1,2} \wedge \boldsymbol{F}_{1,3} \wedge \ldots \wedge \boldsymbol{F}_{1,j} \wedge \ldots \wedge \boldsymbol{F}_{1,n-2} \\ & \wedge \boldsymbol{F}_{2,2} \wedge \boldsymbol{F}_{2,3} \wedge \ldots \wedge \boldsymbol{F}_{2,j} \wedge \ldots \wedge \boldsymbol{F}_{2,n-2} \\ & \wedge \boldsymbol{F}_{3,3} \wedge \ldots \wedge \boldsymbol{F}_{3,j} \wedge \ldots \wedge \boldsymbol{F}_{3,n-2} \\ & \ldots\ldots\ldots\ldots\ldots\ldots \\ & \wedge \boldsymbol{F}_{n-3,n-3} \wedge \boldsymbol{F}_{n-3,n-2} \\ & \wedge \boldsymbol{F}_{n-2,n-2} \end{aligned}$$

Ostatnia postać zapisu formuły *Full3SAT* pozwala postrzegać cały problem w kategoriach struktury mającej postać tablicy trójkątnej górnej.

Grupując inaczej podformuły $\boldsymbol{F}_{i,j}$, formułę *Full3SAT* można zapisać w postaci:

$$Full3SAT = \boldsymbol{E}_1 \wedge \boldsymbol{E}_2 \wedge \ldots \wedge \boldsymbol{E}_j \wedge \ldots \wedge \boldsymbol{E}_{n-3} \wedge \boldsymbol{E}_{n-2}$$

gdzie $\boldsymbol{E}_j$ dla $j = 1, 2, \ldots, n\text{-}2$ jest koniunkcją podformuł $\boldsymbol{F}_{i,j}$ dla $i = 1, \ldots, j$. tj. na przykład $\boldsymbol{E}_j = \boldsymbol{F}_{1,j} \wedge \boldsymbol{F}_{2,j} \wedge \ldots \wedge \boldsymbol{F}_{j,j}$. Wtedy zapis formuły *Full3SAT* pozwala postrzegać cały problem w kategoriach struktury przyjmującej postać tablicy trójkątnej dolnej.

$$\begin{aligned} Full3SAT = \; & \boldsymbol{F}_{1,1} \\ & \wedge \boldsymbol{F}_{1,2} \wedge \boldsymbol{F}_{2,2} \\ & \wedge \boldsymbol{F}_{1,3} \wedge \boldsymbol{F}_{2,3} \wedge \boldsymbol{F}_{3,3} \\ & \ldots\ldots\ldots\ldots \\ & \wedge \boldsymbol{F}_{1,j} \wedge \boldsymbol{F}_{2,j} \wedge \boldsymbol{F}_{3,j} \ldots \wedge \boldsymbol{F}_{j,j} \\ & \ldots\ldots\ldots\ldots \\ & \wedge \boldsymbol{F}_{1,n-3} \wedge \boldsymbol{F}_{2,n-3} \wedge \boldsymbol{F}_{3,n-3} \wedge \ldots \wedge \boldsymbol{F}_{n-4,n-3} \wedge \boldsymbol{F}_{n-3,n-3} \\ & \wedge \boldsymbol{F}_{1,n-2} \wedge \boldsymbol{F}_{2,n-2} \wedge \boldsymbol{F}_{3,n-2} \wedge \ldots \wedge \boldsymbol{F}_{n-4,n-2} \wedge \boldsymbol{F}_{n-3,n-2} \wedge \boldsymbol{F}_{n-2,n-2} \end{aligned}$$

W obu odwzorowaniach, trójkątnej tablicy górnej i dolnej, elementami są wyrażenia $\boldsymbol{F}_{i,j}$ utworzone przez koniunkcję funkcji składających się na zaznaczone w obrazie graficznym grupy (rys. 2.1).

W przypadku problemu **2SAT**, po uporządkowaniu formuły *Full2SAT* wg kluczy określonych przez numer pierwszej zmiennej oraz całkowitą rozpiętość, otrzymamy zapis w postaci:

$$\begin{aligned} Full2SAT = \; & \boldsymbol{h}(x_1, x_2) \wedge \boldsymbol{h}(x_1, x_3) \wedge \ldots \wedge \boldsymbol{h}(x_1, x_{n-1}) \wedge \boldsymbol{h}(x_1, x_n) \\ & \wedge \boldsymbol{h}(x_2, x_3) \wedge \boldsymbol{h}(x_2, x_4) \wedge \ldots \wedge \boldsymbol{h}(x_2, x_{n-1}) \wedge \boldsymbol{h}(x_2, x_n) \\ & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \end{aligned}$$

$\wedge\ \boldsymbol{h}(x_i, x_{i+1}) \wedge\ \ldots \wedge \boldsymbol{h}(x_i, x_{n-1}) \wedge \boldsymbol{h}(x_i, x_n)$

……………………………………………..

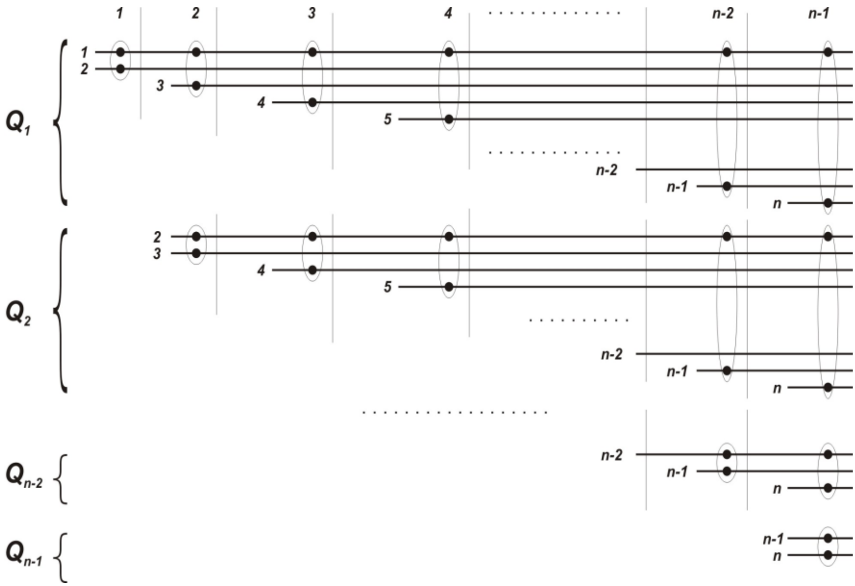
$\wedge\ \boldsymbol{h}(x_{n-2}, x_{n-1}) \wedge \boldsymbol{h}(x_{n-2}, x_n)$

$\wedge\ \boldsymbol{h}(x_{n-1}, x_n)$

który po wprowadzeniu oznaczenia $\boldsymbol{\Theta}_i = \boldsymbol{h}(x_i, x_{i+1}) \wedge \boldsymbol{h}(x_i, x_{i+2}) \wedge \ldots \wedge \boldsymbol{h}(x_i, x_{n-1}) \wedge \boldsymbol{h}(x_i, x_n)$ dla $i = 1, 2, \ldots, n-1$ można przedstawić w postaci:

$$Full2SAT = \boldsymbol{\Theta}_1 \wedge \boldsymbol{\Theta}_2 \wedge \ldots \wedge \boldsymbol{\Theta}_{i-1} \wedge \boldsymbol{\Theta}_i \wedge \boldsymbol{\Theta}_{i+1} \wedge \ldots \wedge \boldsymbol{\Theta}_{n-2} \wedge \boldsymbol{\Theta}_{n-1}$$

Podobnie jak to uczyniono dla formuły *Full3SAT* i jego podformuł $\boldsymbol{\Omega}_i$, dla formuły *Full2SAT*, każda z podformuł $\boldsymbol{\Theta}_i$ może być zobrazowana graficznie, co ilustruje rysunek 2.2.



Rys. 2.2 Graficzny obraz formuły *Full2SAT*=$\Theta_1 \wedge \Theta_2 \wedge \ldots \wedge \Theta_i \wedge \ldots \wedge \Theta_{n-2} \wedge \Theta_{n-1}$

Także *Full2SAT* może być postrzegany w kategoriach struktury tablicy trójkątnej górnej lub dolnej. Elementami tablic w tym przypadku będą proste wyrażenia - funkcje dwóch zmiennych.

Warto w tym miejscu zauważyć, że liczebność funkcji tworzących podformułę $\boldsymbol{\Omega}_1$ w *Full3SAT* jest równa sumie wszystkich funkcji modelu referencyjnego *Full2SAT* , i dalej –

liczebność funkcji tworzących Ω_i jest równa sumie funkcji tworzących formuły $\Theta_i \wedge \Theta_{i+1} \wedge \ldots \wedge \Theta_{n-2} \wedge \Theta_{n-1}$ modelu referencyjnego *Full2SAT*.

Operacja uporządkowania formuł *Full3SAT* i *Full2SAT* jest operacją wykonywaną w czasie wielomianowym. Samoistnie nie umożliwia konstruowania algorytmu wielomianowego – pozwala jednak na opracowanie różnorodnych strategii przy konstruowaniu takiego (takich) algorytmu.

Z analizy pełnego wyrażenia *Full3SAT* opisanego przez koniunkcję funkcji modelu referencyjnego oraz z analizy uproszczonego wyrażenia *Full3SAT* opisanego przez koniunkcję podformuł Ω_i wynika, że każda podformuła Ω_i jest problemem **3SAT** i charakteryzuje się pewną specyficzną strukturą. Analogicznie jest w przypadku wyrażenia *Full2SAT* i podformuł Θ_i. Każda z podformuł Θ_i jest problemem **2SAT** i charakteryzuje się specyficzną strukturą.

Te specyficzne struktury będziemy nazywać schematem wierszowym. Dodatkowo, w obrazie graficznym formuł *Full3SAT* oraz *Full2SAT* (rys. 2.1 i 2.2) można wskazać jeszcze dwie inne specyficzne struktury. Jedną z nich, w przypadku zapisu formuły *Full3SAT* = $E_1 \wedge E_2 \wedge \ldots \wedge E_j \wedge \ldots \wedge E_{n-3} \wedge E_{n-2}$ jest reprezentacja podformuł E_j. Nie definiując ich w tym miejscu, zarezerwujemy dla nich określenia – schemat kolumnowy i schemat diagonalny. Przyjmujemy także założenie, że wyodrębnienie tych specyficznych struktur będzie możliwe dla wszystkich **KSAT**.

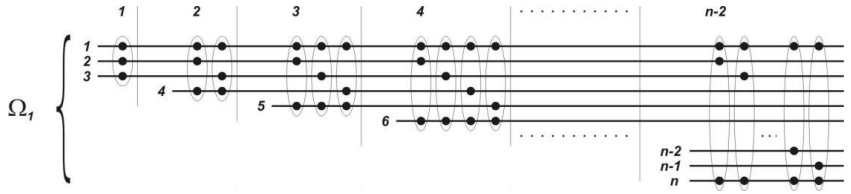
2.4 Struktury elementarne modeli referencyjnych

Konkretny przypadek problemu **3SAT** może przybierać postać istotnie różniącą się od postaci modelu referencyjnego *Full3SAT*. Dla ułatwienia dyskusji posłużymy się sygnalizowanymi już pojęciami schematu wierszowego, kolumnowego i diagonalnego.

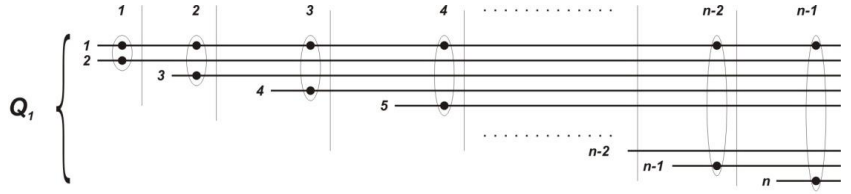
Schemat wierszowy problemu **3SAT** odwzorowuje podformuły Ω_i modelu referencyjnego *Full3SAT*. Charakteryzuje się wystąpieniem co najwyżej jednej funkcji trzech zmiennych o kolejnych numerach. Pozostałe funkcje są funkcjami wszystkich takich kombinacji trzech spośród pozostałych *n-i+1* zmiennych,

w których pierwszym elementem kombinacji jest zmienna o numerze *i*, ponadto funkcje te są pogrupowane tak by każda grupa zawierała funkcje mające pierwszą i trzecią zmienną o identycznych numerach (lub inaczej, w każdej z *n*-2 grup sytuowane są funkcje różniące się numerem tylko drugiej zmiennej[5]). W ten sposób, w wyrażeniach $F_{i,j}$ opisujących koniunkcję funkcji w grupach, wszystkie drugie zmienne występujące w funkcjach są względem siebie niezależne.

Graficzny obraz schematu wierszowego Ω_i na przykładzie $\Omega_1 = F_{1,1} \wedge F_{1,2} \wedge F_{1,3} \wedge \ldots \wedge F_{1,j} \wedge \ldots \wedge F_{1,n-2}$ ilustruje rysunek 2.3 i analogicznie dla **2SAT**, graficzny obraz schematu wierszowego Θ_i na przykładzie $\Theta_1 = h(x_1, x_2) \wedge h(x_1, x_3) \wedge \ldots \wedge h(x_1, x_{n-1}) \wedge h(x_1, x_n)$ ilustruje rysunek 2.4.



Rys. 2.3 Bazowy schemat wierszowy 3SAT.



Rys. 2.4 Bazowy schemat wierszowy 2SAT.

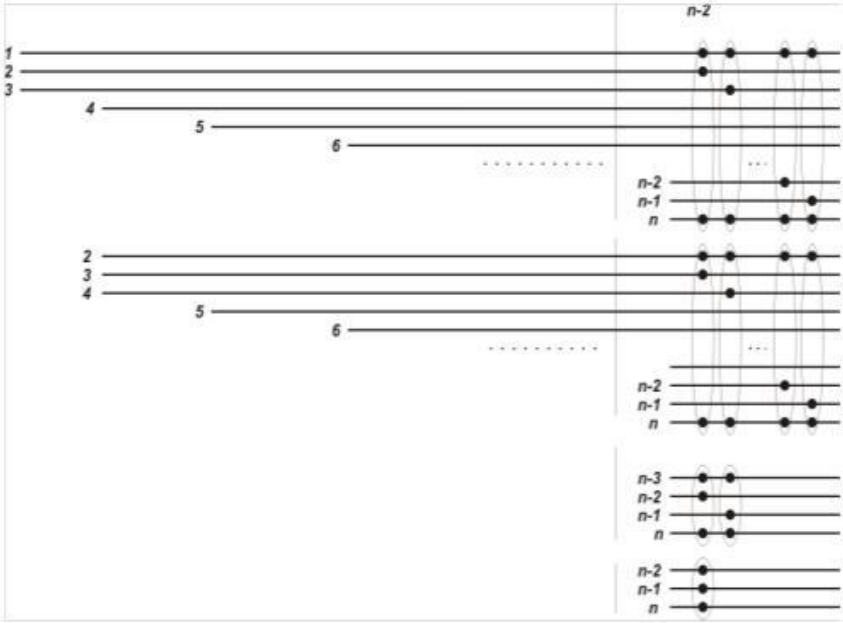
W problemie **2SAT**, w wyodrębnionych grupach schematu wierszowego Θ_i występują pojedyncze funkcje dwóch zmiennych, z których pierwsza o numerze *i* jest wspólna,

[5] Mówimy tu o numerze zmiennej w sensie uporządkowanej trójki numerów zmiennych występujących w opisie funkcji trzech dowolnych zmiennych z pośród wszystkich *n* zmiennych opisujących problem **3SAT**.

a wszystkie pozostałe z pośród *n-i+1* są względem siebie niezależne.

Schemat kolumnowy problemu **3SAT** ma postać rozłącznych grup funkcji i charakteryzuje się wystąpieniem co najwyżej jednej funkcji trzech zmiennych o kolejnych numerach. Pozostałe funkcje w każdej z możliwych *(n-2)* grup są funkcjami wszystkich takich kombinacji trzech spośród *n* zmiennych, w których trzecią zmienną jest zmienna o numerze *n*. Wyrażenia opisujące schematy kolumnowe **3SAT** były już wcześniej zdefiniowane w formule *Full3SAT* w postaci $Full3SAT = E_1 \wedge E_2 \wedge \ldots \wedge E_j \wedge \ldots \wedge E_{n-3} \wedge E_{n-2}$.

Graficzny obraz schematu kolumnowego E_j na przykładzie E_{n-2} ilustruje rysunek 2.5 ($E_{n-2} = F_{1,n-2} \wedge F_{2,n-2} \wedge F_{3,n-2} \wedge \ldots \wedge F_{n-4,n-2} \wedge F_{n-3,n-2} \wedge F_{n-2,n-2}$).

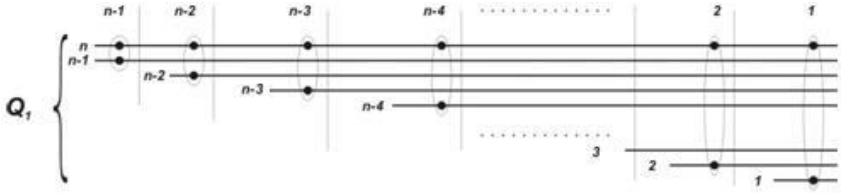


Rys. 2.5 Bazowy schemat kolumnowy **3SAT**.

W przypadku problemu *Full2SAT* *j-ty* schemat kolumnowy odwzorowuje formułę będącą koniunkcją *j* funkcji, w których w odróżnieniu od schematu wierszowego, we wszystkich funkcjach jako druga zmienna występuje x_j, natomiast jako pierwsze zmienne występują wszystkie pozostałe *n-j+1* zmienne.

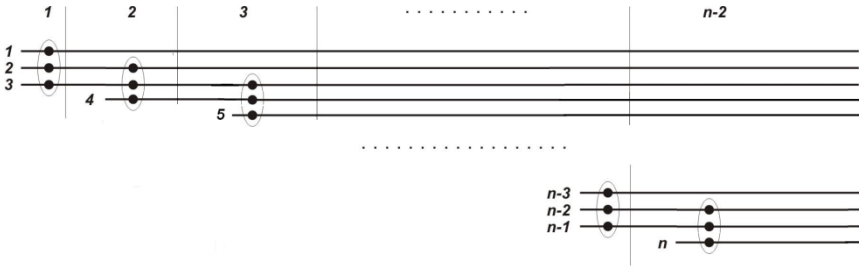
Przykładem formuły odwzorowywanej przez *n-1* schemat kolumnowy modelu referencyjnego *Full2SAT* jest wyrażenie w postaci: $\boldsymbol{h}\,(x_1\,,\,x_n\,) \wedge \boldsymbol{h}\,(x_2\,,\,x_n\,) \wedge \;\ldots \wedge \boldsymbol{h}\,(x_i\,,\,x_n\,) \wedge \ldots \wedge \boldsymbol{h}\,(x_{n-2}\,,\,x_n\,) \wedge \boldsymbol{h}\,(x_{n-1}\,,\,x_n\,)$.

Liczebność funkcji składających się na schemat kolumnowy *Full2SAT* jest identyczna jak w przypadku schematu wierszowego. Zarówno to jak i uwagi dotyczące struktury funkcji składających się na schemat wierszowy i kolumnowy sugerują, że schematy te są względem siebie transponowane. Można przyjąć więc założenie, że podobnie będzie w przypadku modelu referencyjnego *Full3SAT* i dalej ogólnie *FullkSAT*.



Rys. 2.6 Bazowy schemat kolumnowy **2SAT**.

Schemat diagonalny modelu referencyjnego *Full3SAT* charakteryzuje się wystąpieniem wyłącznie funkcji trzech zmiennych o kolejnych numerach. Obraz graficzny tak zdefiniowanego schematu diagonalnego prezentuje rysunek 2.7.



Rys. 2.7 Bazowy schemat diagonalny **3SAT**.

Na przykład pełen schemat diagonalny *Full3SAT* będzie określony przez wyrażenie w postaci $Full3SAT = d_1(x_1, x_2, x_3) \wedge d_2(x_2, x_3, x_4) \wedge \ldots \wedge d_i(x_i, x_{i+1}, x_{i+2}) \wedge \ldots \wedge d_{n-2}(x_{n-2}, x_{n-1}, x_n)$.

W przypadku modelu referencyjnego *Full2SAT*, w schemacie diagonalnym będą występowały wyłącznie funkcje dwóch zmiennych o kolejnych numerach. Pełny schemat diagonalny

określany jest wtedy przez wyrażenie postaci $Full2SAT = d_1(x_1, x_2) \wedge d_2(x_2, x_3) \wedge \ldots \wedge d_i(x_i, x_{i+1}) \wedge \ldots \wedge d_{n-2}(x_{n-2}, x_{n-1}) \wedge d_{n-1}(x_{n-1}, x_n)$.

2.5 Perspektywa algorytmów wielomianowych kSAT

Wcześniejsza dyskusja parametryzacji oraz analiza strukturalna formuł problemów **KSAT** wniosła nowe, niekiedy uwypukliła oczywiste spostrzeżenia, w szczególności:

- że w przypadku logicznych funkcji trzech zmiennych możemy operować wzajemnie komplementarnymi czterema parami klauzul (klauzule *Horna* i *coHorna*).
- że problemy **KSAT** mogą być postrzegane w kategoriach tablic trójkątnych,
- że można operować w formułach wyrażeniami, w których zmienne nie są zależne wprost od innych i można te wyra-żenia odpowiednio uporządkować.

Wszystkie wymienione spostrzeżenia sprzyjają lepszemu rozumieniu problemu. W powiązaniu i z wykorzystaniem znanych już rezultatów złożoności obliczeniowej, a także innych dziedzin, mogą stanowić elementy nowych koncepcji konstrukcji algorytmów wielomianowych **KSAT**.

Znajdowanie algorytmów ma bliski związek z teorią rozwiązywania problemów. Tak jak dla procesu rozwiązywania problemów, tak i dla opracowania algorytmów brakuje ogólnej metodologii postępowania. Dysponujemy jedynie co najwyżej zbiorem luźnych reguł, zasad i rekomendacji przydatnych w czterech wyodrębnionych fazach procesu rozwiązywania problemów i opracowania algorytmów: (i) zrozumienie problemu; (ii) plan rozwiązania problemu; (iii) realizacja planu rozwiązania; (iv) ocena poprawności rozwiązania.

Występujące w prezentowanym schemacie fazy są ze sobą ściśle powiązane. Ogólnie cały proces może mieć charakter iteracyjny i niekoniecznie prowadzony zawsze zgodnie z przedstawioną sekwencją. Pojawiające się trudności na określonym etapie, wymagają niekiedy powrotów do wcześniejszych faz – jednak, dominującą pozostaje niezmiennie faza zrozumienia problemu.

Opracowanie algorytmów mogą ułatwić wypracowane wzorce metod algorytmicznych, takie jak na przykład stopniowe uściślanie problemu (dekompozycja) czy poszukiwanie analogii. Oprócz tego, bardzo istotną rolę w konstruowaniu algorytmów odgrywają struktury danych. Zwraca na to uwagę Niklaus Wirth w przedmowie swojej książki *„Algorytmy + struktury danych = programy"*, pisząc *„… Wszystkie decyzje dotyczące struktury danych mogą być podjęte jedynie na podstawie znajomości algorytmów zastosowanych do danych i, vice versa, znajomość i wybór algorytmów zależą często ściśle od struktury danych. …"*.

Przytoczony cytat można traktować jako wzmocnienie zawartych we wstępie uwag dotyczących roli odpowiednich struktur w konstruowaniu i ulepszaniu algorytmów. W dyskutowanym modelu referencyjnym **KSAT**, opisy poszczególnych wariantów problemu i ich graficzna reprezentacja jednoznacznie pokazują, że mogą być one odwzorowane na struktury tablicowe jedno-, dwu- i trójwymiarowe, a te w prosty sposób poddają się linearyzacji.

Dodatkowo, kluczowe operacje na strukturach tablicowych mogą być realizowane na wiele sposobów o zróżnicowanej wydajności. Wiemy jednak, że samoistnie są problemami zamkniętymi bez luki algorytmicznej.

Tabela 2.5 Złożoność wybranych algorytmów sortowania

	Nazwa	**Rozmiar** (liczba procesorów)	**Czas** (najgorszy przypadek)	**Iloczyn** (rozmiar x czas)
SEKWENCYJNE	**Sortowanie bąbelkowe**	**1**	$O(N^2)$	$O(N^2)$
	Sortowanie przez scalanie	**1**	$O(N \log N)$	$O(N \log N)$
RÓWNOLEGLE	**Zrównoleglone sortowanie przez scalanie**	$O(N)$	$O(N)$	$O(N^2)$
	„Optymalna" sieć sortująca	$O(N)$	$O(\log N)$	$O(N \log N)$

W szczególności, kluczowa w modelu referencyjnym, operacja sortowania funkcji (patrz tabela 2.5) może być wykonana sekwencyjnie dowolnym ze znanych algorytmów wielomia-

nowych, a równolegle przy wielomianowej ilości procesorów w czasie nie gorszym niż polilogarytmiczny[6].

W ten sposób dysponujemy dobrym punktem wyjścia do dyskusji relacji zawierania klas $\boldsymbol{NC \subseteq NP}$, a nawet szerzej $\boldsymbol{NC \subseteq P \subseteq NP}$. ***NC*** – klasa Nick'a obejmuje problemy dla których istnieją algorytmy równoległe, realizowane przez wielomianową liczbę procesorów w czasie co najwyżej polilogarytmicznym, tj. o złożoności $O(n^k \cdot log^p (n))$.

Istnieje jeszcze bardziej znaczący powód, dla którego takie rozważania mają sens. Otóż bez wnikliwej analizy, pewne przypadki problemów **2SAT** (np. o elementarnej strukturze wierszowej $\boldsymbol{\Theta_1}$ – patrz rys. 2.4) można zakwalifikować do klasy ***NC***. Jak zauważyliśmy wcześniej, uporządkowanie tego typu struk-tury można uzyskać równolegle w czasie logarytmicznym, a zakładając oddzielnie wartościowanie zmiennej $x_1 = 0$ oraz $x_1 = 1$, wartościowania spełniające dla pozostałych zmiennych, jeśli istnieją, są niezależne i ich ustalenie równolegle uzyskujemy w czasie $O(1)$.

Zdefiniowanie modelu referencyjnego *FullkSAT* w kategoriach funkcji logicznych, określa jeszcze jeden kierunek nowej perspektywy. Wzajemna ekwiwalentność rachunku zdań i funkcji logicznych uzasadnia uwzględnianie metod i mechanizmów wypracowanych dla problemów syntezy układów kombinacyjnych i podjęcie prób implementacji sprzętowej konstruowanych algorytmów rozwiązywania problemów spełnialności **SAT**.

Zarówno we wstępie, jak i w toku ogólnych rozważań dotyczących parametryzacji problemów **KSAT**, wskazano na niedoskonałość dotychczasowych sposobów klasyfikacji problemów **3SAT** w szczególności. W oparciu o analizę sposobu definiowania funkcji logicznych dwóch i trzech zmiennych (patrz tabela 2.1 i 2.3) oraz wprowadzając nowe pojęcie klauzuli *coHorna*, problemy **3SAT** można klasyfikować jako **HORN3SAT**, **coHORN3SAT** i **MIXHORN3SAT** (jako koniunkcja **HORN3SAT** i **coHORN3SAT**).

[6] Dla linearnej *N-elementowej* tablicy w czasie logarytmicznym, ale w przypadku *Full2SAT* liczebność elementów jest rzędu n^2, a dla *Full3SAT* – rzędu n^3 (n jest ilością zmiennych).

Problemy **HORN3SAT** i jako pokazano to w elementarny sposób także **coHORN3SAT**, mają rozwiązania w czasie wielomianowym. Tak więc wyzwaniem pozostaje problem **MIXHORN3SAT**. Szkic rozwiązania został zarysowany przy okazji dyskusji przykładu 2.1. W tym miejscu warto jednak zwrócić uwagę, że koniunkcja **HORN3SAT** i **coHORN3SAT** nie jest spełniana, jeśli dla którejkolwiek zmiennej x_i wartościowania w wektorach rozwiązań będą jednoznaczne, bo jako wartościowania zmiennych spełniające „swój" problem, będąc jednoznacznymi, muszą mieć wartości przeciwne.

Można więc sformułować warunek konieczny spełnialności problemu **MIXHORN3SAT** następująco: jeśli w jednym z wektorów rozwiązań **HORN-** lub **coHORN3SAT** wartościowanie spełniające zmiennej x_i jest jednoznacznie określone, w drugim wektorze rozwiązań (odpowiednio dla **coHORN-** lub **HORN3SAT**) musi być dopuszczalne dowolne (*true* i *false*) wartościowanie zmiennej x_i jako spełniające.

Pokazane podobieństwo strukturalne modeli referencyjnych problemów *Full2SAT* i *Full3SAT* wytycza jeszcze jeden kierunek dalszych rozważań. Zasadnym wydaje się zbadanie możliwych przekształceń wzajemnych struktur tych problemów, możliwych operacji oraz ich wpływu na postać całej struktury i jej elementy (funkcje).

Wszystkie poczynione wyżej uwagi sprawiają, że wiedza o istocie problemu spełnialności dzięki zdefiniowaniu modelu referencyjnego znacząco się poszerzyła. Cząstkowe wyniki zaprezentowane w tym opracowaniu, nie rozstrzygając wprost równości klas ***NC = P = NP***, przybliżają jednak do takiego rozstrzygnięcia w modelu maszyny Turinga.

BIBLIOGRAFIA

[1] Goldberg David E., Algorytmy genetyczne i ich zastoso-wania, Warszawa, WNT, 1998

[2] Gasarch William, http://www.cs.umd.edu/~gasarch/papers/poll.pdf

[3] Harel David, Rzecz o istocie informatyki. Algorytmika, Warszawa, WNT, 1992

[4] Papadimitriou Christos, Złożoność obliczeniowa, Warszawa, WNT, 2007

[5] Penrose Roger, Nowy umysł cesarza, Warszawa, PWN, 1996

[6] Schneier Bruce, Kryptografia dla praktyków. Protokoły, algorytmy i programy źródłowe w języku C, Warszawa, WNT, 1995

[7] Sipser Michael, Wprowadzenie do teorii obliczeń, Warszawa, WNT, 2009

[8] Wirth Niklaus, Algorytmy + struktury danych = programy, Warszawa, WNT, 1980

R o z d z i a ł t r z e c i

Badanie struktur problemów *kSAT* i problemów pokrewnych

Marek Malinowski

Technologie Teleinformatyczne

Podstawą prowadzonych rozważań są modele referencyjne problemów KSAT. Cechuje je opis w kategoriach funkcji logicznych i dla k>1 podobieństwo elementarnych struktur wierszowych, kolumnowych i diagonalnych.
W kolejnych punktach pokazano, że opis problemów KSAT w kate-goriach funkcji logicznych pozwala zdefiniować pewne typy prze-kształceń wzajemnych struktur elementarnych 2SAT i 3SAT, a prze-analizowane związki pomiędzy funkcjami logicznymi 2- i 3-zmiennych pozwalają wprowadzić nową klasyfikację problemów 3SAT. Wreszcie w końcowych punktach pokazano, że dla każdej struktury elementarnej 3SAT można skonstruować algorytm wielomianowy, a istnienie takich algorytmów rozstrzyga kwestię *P versus NP*.

3

Poszukaj analogii

[z teorii rozwiązywania problemów]

3.1 Wstęp

W szeregu prac dyskutując rodzaje spełnialności, zwraca się uwagę na linię podziału rozgraniczającą warianty tego problemu. Jedne należą do ***P***, inne pozostają ***NP****-zupełne*. Wskazuje się także, że uogólnienie problemu należącego do ***P*** może spowodować jego zmianę statusu. Jest tak w przypadku problemu *kSAT CNF* dla $k > 1$, w którym wszystkie klauzule mają po k literałów. Warianty dla $k=1$ (*1SAT*) i $k=2$ (*2SAT*) oraz problemy skonstruowane wyłącznie przy pomocy *klauzul Horna* (*HornSAT*) należą do ***P***. Z drugiej strony, uogólnienie **2SAT** do **Max2SAT** (jaka jest największa liczba klauzul, które można spełnić), czyni tak uogólniony problem ***NP****-zupełnym*.

Problem **KSAT** dla $k \geq 3$ należy także do ***NP****-zupełnych*, przy czym znana jest redukcja formuły *kSAT CNF* do *3SAT CNF*. Ogólnie, jeśli klauzula zawiera k literałów $(x_1 \vee x_2 \vee ... \vee x_k)$, to możemy wprowadzając nowe zmienne zastąpić ją przez koniunkcję k-2 klauzul $(x_1 \vee x_2 \vee y_1)(\overline{y_1} \vee x_3 \vee y_2)(\overline{y_2} \vee x_4 \vee \overline{y_3}) \ \ldots \ (\overline{y_{k-3}} \vee x_{k-1} \vee x_k)$, z których każda zawiera 3 literały [2 s.316]. Na przykład klauzulę $(x_1 \vee x_2 \vee x_3 \vee x_4)$ można zastąpić równoważną formułą złożoną z dwóch klauzul $(x_1 \vee x_2 \vee y_1)\ (\overline{y_1} \vee x_3 \vee x_4)$. Niestety tego rodzaju redukcja nie prowadzi do **2SAT**.

Fakt, że znany sposób redukcji **KSAT** nie zmienia jego statusu, nie musi jednak oznaczać braku innego sposobu postępowania, które prowadzi do takiej zmiany.

W podsumowaniu dyskusji modelu referencyjnego **KSAT** (rozdział II), wskazano perspektywy konstrukcji algorytmów wielomianowych. Stwierdzono, że model referencyjny stanowi stabilny, bezwzględny model odniesienia dla prowadzenia usystematyzowanych analiz różnorodnych wariantów **KSAT**.

Przede wszystkim, pozwala rozpatrywać problemy **KSAT** na wyższym poziomie ogólności, tj. w kategoriach funkcji logicznych zamiast klauzul. Tym samym, tak postrzegany problem spełnialności może być analizowany nie tylko w formalizmie rachunku zdań, ale także z użyciem innych postaci algebry Boole'a – algebry zbiorów, algebry sygnałów i algebry wektorów binarnych.

Ponadto, wobec podobieństwa strukturalnego modeli referencyjnych **KSAT**, zasadnym jest zbadanie możliwych przekształceń wzajemnych struktur tych modeli, możliwych operacji oraz ich wpływu na postać całej struktury i jej elementy.

Te dwa aspekty – rozpatrywanie **KSAT** w kategoriach funkcji oraz podobieństwo strukturalne modeli referencyjnych **KSAT** - stanowią treść dyskusji w kolejnych punktach. Ponieważ podstawą prowadzonych rozważań jest model referencyjny, opisany w poprzednim rozdziale, używane będą terminologia i oznaczenia tam zdefiniowane.

W szczególności dotyczy to pojęcia klauzuli *coHorna* i rzędu funkcji oraz oznaczeń h i f funkcji dwóch i trzech zmiennych odpowiednio, sposobu numeracji funkcji, a także oznaczeń podformuł Ω i Θ dla *Full2-* i *Full3SAT* odpowiednio.

3.2 Przekształcenia wzajemne *2SAT* i *3SAT*

Wzajemna ekwiwalentność rachunku zdań i funkcji logicznych została już raz wykorzystana w dyskusji jednostajnej wielomianowej sieci problemu **KSAT** (rozdział I). W tym miejscu, w ujęciu funkcji logicznych analizowane będą możliwe odwzorowania funkcji trzech zmiennych przez funkcje dwóch zmiennych. Wiadomo, że przy ich pomocy funkcji dwóch zmiennych możemy opisać każdą funkcję n zmiennych. Jednak postać zapisu *CNF* sprawia, że spośród 16 funkcji dwóch zmiennych musimy ograniczyć się do funkcji **AND**. Podane dalej

przykłady pokazują, że przynajmniej w odniesieniu do części funkcji trzech zmiennych jest to możliwe.

Zastosowanie takich odwzorowań pozwala klasyfikować problemy **3SAT** w innym niż dotychczas ujęciu. Analizowana będzie także użyteczność takich odwzorowań w przekształceniach modelu referencyjnego.

3.2.1 Dyskusja funkcji logicznych dwóch i trzech zmiennych w *kSAT*

Funkcja logiczna przyporządkowuje wartościom logicznym zmiennych x wartość logiczną zmiennych y i jest opisywana formalnie za pomocą wyrażeń logicznych. Wyrażenia logiczne mogą być zapisywane w wielu różnorodnych i równoważnych sobie postaciach.

Wyrażenie logiczne definiujemy następująco:

- *0, 1,* x_i i $\neg x_i$ są wyrażeniami logicznymi oraz jeśli φ_1 i φ_2 są wyrażeniami logicznymi, to $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$ są wyrażeniami logicznymi;
- wyrażenie logiczne może mieć tylko postać opisaną wyżej, ale zamiast zmiennych logicznych x_i mogą wystąpić inne zmienne logiczne, np. y_j wartości logiczne funkcji.

Wykorzystując fakt, że jako argumenty funkcji logicznej, oprócz zmiennych x, mogą wystąpić także inne funkcje, to oznaczając przez f^i funkcję i zmiennych możemy zapisać $f^{i+1} = f^2(f^i(x_1, x_2, \ldots, x_i), x_{i+1})$, w którym f^{i+1} jest wyrażeniem funkcji $i+1$ zmiennych [3 s.22].

W odniesieniu do funkcji $f^3(x_r, x_s, x_t)$ trzech zmiennych można więc operować na przykład wyrażeniami postaci:

$f^3(f^2(x_r, x_s), f^1(x_t))$ lub $f^3(f^2(x_r, x_t), f^1(x_s))$ lub $f^3(f^2(x_s, x_t), f^1(x_r))$ (1)

$f^3(f^2(x_r, x_s), f^2(x_r, x_t))$ lub $f^3(f^2(x_r, x_s), f^2(x_s, x_t))$ lub $f^3(f^2(x_r, x_t), f^2(x_s, x_t))$ (2)

$f^3(f^2(x_r, x_s), f^2(x_r, x_t), f^2(x_s, x_t))$ (3)

ale także $f^3(f^3(x_r, x_s, x_t), f^2(x_r, x_t))$.

Jeśli φ będzie wyrażeniem *3SAT CNF* opisującym funkcję n zmiennych w postaci koniunkcji m funkcji trzech zmiennych:

$$\varphi = f_1^3(x_r, x_s, x_t) \wedge f_2^3(x_r, x_s, x_t) \wedge \ \ldots \ \wedge f_m^3(x_r, x_s, x_t)$$

gdzie: $x_r, x_s, x_t \in \{x_1, x_2, \ldots, x_n\}$ oraz $r \in \{1, 2, \ldots n\text{-}2\}$, $s \in (r+1, r+2, \ldots, n\text{-}1\}$ i $t \in \{s+1, s+2, \ldots n\}$, tj. $r < s < t$,

i jeśli występujące w wyrażeniu φ funkcje $f^3(x_r, x_s, x_t)$ będzie można wyrazić w szczególności w postaciach *(1)*, *(2)* lub *(3)* z operatorem prefiksowym AND (patrz uwaga we wstępie), to wyrażenie *3SAT CNF* przyjmie postać *2SAT CNF*.

Poniższe przykłady pokazują, że przynajmniej w odniesieniu do części funkcji $f^3(x_r, x_s, x_t)$ jest to możliwe. Przykładowe funkcje określone są w postaci zapisu *DNF*, a elementarne koniunkcje zapisane są bez użycia jawnie operatora ∧ (zapis $\overline{x}_1\,\overline{x}_2\,x_3$ oznacza koniunkcję trzech zmiennych, z których dwie pierwsze są zanegowane).

Przykład 3.1 Odwzorowania funkcji $f^3(x_r, x_s, x_t)$ przez funkcje dwóch zmiennych, określone według zależności oznaczonych *(2)*.

Funkcja $f_{50}(x_1, x_2, x_3) = \overline{x}_1\,\overline{x}_2\,x_3 \vee x_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,x_3$ może być wyrażona przez koniunkcję $h_3(x_1, x_2) \wedge h_{14}(x_1, x_3)$ bo:

$$h_3(x_1, x_2) \wedge h_{14}(x_1, x_3) = (\overline{x}_1\,\overline{x}_2 \vee x_1\,\overline{x}_2) \wedge (x_1\,\overline{x}_3 \vee \overline{x}_1\,x_3 \vee x_1\,x_3) = \\ = \overline{x}_1\,\overline{x}_2\,x_3 \vee x_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,x_3$$

Funkcja $f_{51}(x_1, x_2, x_3) = \overline{x}_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,\overline{x}_3 \vee \overline{x}_1\,\overline{x}_2\,x_3 \vee x_1\,\overline{x}_2\,x_3$ może być wyrażona przez koniunkcję $h_3(x_1, x_2) \wedge h_5(x_2, x_3)$ bo:

$$h_3(x_1, x_2) \wedge h_5(x_2, x_3) = (\overline{x}_1\,\overline{x}_2 \vee x_1\,\overline{x}_2) \wedge (\overline{x}_2\,\overline{x}_3 \vee \overline{x}_2\,x_3) = \\ = \overline{x}_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,\overline{x}_3 \vee \overline{x}_1\,\overline{x}_2\,x_3 \vee x_1\,\overline{x}_2\,x_3$$

Funkcja $f_{35}(x_1, x_2, x_3) = \overline{x}_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,x_3$ może być wyrażona przez koniunkcję $h_{11}(x_1, x_3) \wedge h_5(x_2, x_3)$, bo:

$$h_{11}(x_1, x_3) \wedge h_5(x_2, x_3) = (\overline{x}_1\,\overline{x}_3 \vee x_1\,\overline{x}_3 \vee x_1\,x_3) \wedge (\overline{x}_2\,\overline{x}_3 \vee \overline{x}_2\,x_3) = \\ = \overline{x}_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,x_3$$

Funkcja $f_{34}(x_1, x_2, x_3) = x_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,x_3$ może być wyrażona przez koniunkcję $h_2(x_1, x_2) \wedge h_5(x_2, x_3)$, ale także $h_6(x_1, x_2) \wedge h_{10}(x_1, x_3)$ oraz $h_{10}(x_1, x_3) \wedge h_5(x_2, x_3)$ bo:

$h_2(x_1, x_2) \wedge h_5(x_2, x_3) = (x_1\,\overline{x}_2) \wedge (\overline{x}_2\,\overline{x}_3 \vee \overline{x}_2\,x_3) = x_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,x_3$
$h_6(x_1, x_2) \wedge h_{10}(x_1, x_3) = (\overline{x}_1\,x_2 \vee x_1\,\overline{x}_2) \wedge (x_1\,x_3 \vee x_1\,\overline{x}_3) = x_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,x_3$
$h_{10}(x_1, x_3) \wedge h_5(x_2, x_3) = (x_1\,x_3 \vee x_1\,\overline{x}_3) \wedge (\overline{x}_2\,\overline{x}_3 \vee \overline{x}_2\,x_3) = x_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,x_3$

Przykład 3.2 Odwzorowania funkcji $f^3(x_r, x_s, x_t)$ przez funkcje dwóch zmiennych, określone według zależności oznaczonych *(3)*.

Funkcja $f_{23}(x_1, x_2, x_3) = x_1 \overline{x}_2 \overline{x}_3 \vee \overline{x}_1 x_2 \overline{x}_3 \vee x_1 \overline{x}_2 \overline{x}_3 \vee \overline{x}_1 \overline{x}_2 \overline{x}_3$ może być wyrażona przez koniunkcję $h_7(x_1, x_2) \wedge h_7(x_2, x_3) \wedge h_7(x_1, x_3)$ bo:

$h_7(x_1, x_2) \wedge h_7(x_2, x_3) \wedge h_7(x_1, x_3) = (\overline{x}_1 \overline{x}_2 \vee \overline{x}_1 x_2 \vee x_1 \overline{x}_2) \wedge (\overline{x}_2 \overline{x}_3 \vee \overline{x}_2 x_3 \vee x_2 \overline{x}_3) \wedge (\overline{x}_1 \overline{x}_3 \vee \overline{x}_1 x_3 \vee x_1 \overline{x}_3) = x_1 \overline{x_2}\, \overline{x_3} \vee \overline{x_1}\, x_2\, \overline{x_3} \vee x_1\, x_2\, \overline{x_3} \vee x_1\, \overline{x_2}\, \overline{x_3}$

Rozpatrując po 225 przypadków możliwych odwzorowań w postaci $h_{1\text{-}15}(x_r x_s) \wedge h_{1\text{-}15}(x_s, x_t)$ (tabela 3.1), $h_{1\text{-}15}(x_r, x_s) \wedge h_{1\text{-}15}(x_r, x_t)$ (tabela 3.2) oraz $h_{1\text{-}15}(x_r, x_t) \wedge h_{1\text{-}15}(x_s, x_t)$ (tabela 3.3), otrzymujemy dla każdego z tych odwzorowań numer odwzorowanej funkcji f trzech zmiennych. Numery $r \in \{1, 2, \ldots n\text{-}2\}$, $s \in \{r+1, r+2, \ldots, n\text{-}1\}$ i $t \in \{s+1, s+2, \ldots n\}$, stanowią uporządkowaną trójkę zmiennych w funkcjach f trzech zmiennych.

Tabela 3.1 Odwzorowania postaci $h_{1\text{-}15}(x_r, x_s) \wedge h_{1\text{-}15}(x_s, x_t)$.

$h_{1\text{-}15}(x_r, x_s)$ \ $h_{1\text{-}15}(x_s, x_t)$	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}
h_1	1	0	1	16	17	16	17	0	1	0	1	16	17	16	17
h_2	2	0	2	32	34	32	34	0	2	0	2	32	34	32	34
h_3	3	0	3	48	51	48	51	0	3	0	3	48	51	48	51
h_4	0	4	4	0	0	4	4	64	64	68	68	64	64	68	68
h_5	1	4	5	16	17	20	21	64	65	68	69	80	81	84	85
h_6	2	4	6	32	34	36	38	64	66	68	70	96	98	100	102
h_7	3	4	7	48	51	52	55	64	67	68	71	112	115	116	119
h_8	0	8	8	0	0	8	8	128	128	136	136	128	128	136	136
h_9	1	8	9	16	17	24	25	128	129	136	137	144	145	152	153
h_{10}	2	8	10	32	34	40	42	128	130	136	138	160	162	168	170
h_{11}	3	8	11	48	51	56	59	128	131	136	139	176	179	184	187
h_{12}	0	12	12	0	0	12	12	192	192	204	204	192	192	204	204
h_{13}	1	12	13	16	17	28	29	192	193	204	205	208	209	220	221
h_{14}	2	12	14	32	34	44	46	192	194	204	206	224	226	236	238
h_{15}	3	12	15	48	51	60	63	192	195	204	207	240	243	252	255

Analizując wszystkie funkcje trzech zmiennych można stwierdzić, że dla odwzorowań w postaci $h_{1\text{-}15}(x_r, x_s) \wedge h_{1\text{-}15}(x_s, x_t)$ (tabela 3.1), $h_{1\text{-}15}(x_r, x_s) \wedge h_{1\text{-}15}(x_r, x_t)$ (tabela 3.2) oraz $h_{1\text{-}15}(x_r, x_t) \wedge h_{1\text{-}15}(x_s, x_t)$ (tabela 3.3), wśród 254 (z pominięciem funkcji stałych ***TRUE*** i ***FALSE***) funkcji trzech zmiennych istnieją takie, dla których można wskazać przynajmniej jedno odwzorowanie poprzez koniunkcję funkcji dwóch zmiennych.

Tabela 3.2 Odwzorowania postaci $h_{1\text{-}15}(x_r,x_s) \wedge h_{1\text{-}15}(x_r,x_t)$.

$h_{1\text{-}15}(x_r,x_s)$ \ $h_{1\text{-}15}(x_r,x_t)$	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}
h_1	1	0	1	16	17	16	17	0	1	0	1	16	17	16	17
h_2	0	2	2	0	0	2	2	32	32	34	34	32	32	34	34
h_3	1	2	3	16	17	18	19	32	33	34	35	48	49	50	51
h_4	4	0	4	64	68	64	68	0	4	0	4	64	68	64	68
h_5	5	0	5	80	85	80	85	0	5	0	5	80	85	80	85
h_6	4	2	6	64	68	66	70	32	36	34	38	96	100	98	102
h_7	5	2	7	80	85	82	87	32	37	34	39	112	117	114	119
h_8	0	8	8	0	0	8	8	128	128	136	136	128	128	136	136
h_9	1	8	9	16	17	24	25	128	129	136	137	144	145	152	153
h_{10}	0	10	10	0	0	10	10	160	160	170	170	160	160	170	170
h_{11}	1	10	11	16	17	26	27	160	161	170	171	176	177	186	187
h_{12}	4	8	12	64	68	72	76	128	132	136	140	192	196	200	204
h_{13}	5	8	13	80	85	88	93	128	133	136	141	208	213	216	221
h_{14}	4	10	14	64	68	74	78	160	164	170	174	224	228	234	238
h_{15}	5	10	15	80	85	90	95	160	165	170	175	240	245	250	255

Tabela 3.3 Odwzorowania postaci $h_{1\text{-}15}(x_r,x_t) \wedge h_{1\text{-}15}(x_s,x_t)$.

$h_{1\text{-}15}(x_r,x_t)$ \ $h_{1\text{-}15}(x_s,x_t)$	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}
h_1	1	4	5	0	1	4	5	0	1	4	5	0	1	4	5
h_2	2	8	10	0	2	8	10	0	2	8	10	0	2	8	10
h_3	3	12	15	0	3	12	15	0	3	12	15	0	3	12	15
h_4	0	0	0	16	16	16	16	64	64	64	64	80	80	80	80
h_5	1	4	5	16	17	20	21	64	65	68	69	80	81	84	85
h_6	2	8	10	16	18	24	26	64	66	72	74	80	82	88	90
h_7	3	12	15	16	19	28	31	64	67	76	79	80	83	92	95
h_8	0	0	0	32	32	32	32	128	128	128	128	160	160	160	160
h_9	1	4	5	32	33	36	37	128	129	132	133	160	161	164	165
h_{10}	2	8	10	32	34	40	42	128	130	136	138	160	162	168	170
h_{11}	3	12	15	32	35	44	47	128	131	140	143	160	163	172	175
h_{12}	0	0	0	48	48	48	48	192	192	192	192	240	240	240	240
h_{13}	1	4	5	48	49	52	53	192	193	196	197	240	241	244	245
h_{14}	2	8	10	48	50	56	58	192	194	200	202	240	242	248	250
h_{15}	3	12	15	48	51	60	63	192	195	204	207	240	243	252	255

W oparciu o tabele 3.1 - 3.3 można skonstruować tablicę odwzorowania funkcji trzech zmiennych na koniunkcje dwóch funkcji dwóch zmiennych. Przykładowe możliwe odwzorowania zawarto w tabeli 3.4.

Tabela 3.4 Przykładowe odwzorowania funkcji 3 zmiennych przez koniunkcje funkcji dwóch zmiennych, gdzie ***(k, m)*** oznacza koniunkcję funkcji $h_k \wedge h_m$ odpowiednich dwóch zmiennych.

Funkcja 3 zmiennych	Przykładowe odwzorowania przez koniunkcje funkcji		
	$h_k(x_r x_s) \wedge h_m(x_r x_t)$	$h_k(x_r x_s) \wedge h_m(x_s x_t)$	$h_k(x_r x_t) \wedge h_m(x_s x_t)$
$f_1(x_r,x_s,x_t)$	(1,1); (1,3); (1,9)	(1,1); (1,3); (1,9)	(1,1); (1,5); (1,9)
$f_2(x_r,x_s,x_t)$	(2,2); (2,3); (2,6);	(2,1); (2,3); (2,9)	(2, 5)
. . . . .	. . .	. . .	. . .
$f_{17}(x_r,x_s,x_t)$	(1,5); (1,7); (1,13);	(1, 5); (5, 5)	(5, 5)
$f_{18}(x_r,x_s,x_t)$	(3,3)	nie istnieje	(6, 5)
$f_{19}(x_r,x_s,x_t)$	(3,7)	nie istnieje	(7, 5)
$f_{20}(x_r,x_s,x_t)$	nie istnieje	(5, 6)	(5, 6)
$f_{21}(x_r,x_s,x_t)$	nie istnieje	(5, 7)	(5, 7)
$f_{22}(x_r,x_s,x_t)$	nie istnieje	nie istnieje	nie istnieje
$f_{23}(x_r,x_s,x_t)$	nie istnieje	nie istnieje	nie istnieje
. . . . .	. . .	. . .	. . .
$f_{32}(x_r,x_s,x_t)$	(2,8); (2,9); (2,12);	(6, 4); (10, 4)	(9, 4); (10; 4)
$f_{33}(x_r,x_s,x_t)$	(3,9)	nie istnieje	(9, 5)
$f_{34}(x_r,x_s,x_t)$	(2,10); (2,11); (2,14);	(6, 5); (10, 5)	(10, 5))
$f_{35}(x_r,x_s,x_t)$	(3,11)	nie istnieje	(11, 5)
. . . . .		. . .	. . .
$f_{48}(x_r,x_s,x_t)$	(3,12)	(7, 4); (11,4); (15, 4)	(12, 4); (12, 5); (13, 4)
$f_{49}(x_r,x_s,x_t)$	(3,13)	nie istnieje	(13, 5)
$f_{50}(x_r,x_s,x_t)$	(3,14)	nie istnieje	(14, 5)
. . . . .	. . .	. . .	. . .
$f_{254}(x_r,x_s,x_t)$	nie istnieje	nie istnieje	nie istnieje

Z zestawienia wynika, że nie wszystkie funkcje trzech zmiennych dają się odwzorować przez koniunkcje dwóch funkcji dwóch zmiennych przynajmniej w jednej z postaci $h_{1\text{-}15}(x_r,x_s) \wedge h_{1\text{-}15}(x_s, x_t)$, $h_{1\text{-}15}(x_r, x_s) \wedge h_{1\text{-}15}(x_r, x_t)$ oraz $h_{1\text{-}15}(x_r, x_t) \wedge h_{1\text{-}15}(x_s, x_t)$. Listę funkcji odwzorowanych w ten sposób można poszerzyć o funkcje odwzorowywane w postaci koniunkcji trzech funkcji dwóch zmiennych, tj. $h_{1\text{-}15}(x_r, x_s) \wedge h_{1\text{-}15}(x_r, x_t) \wedge h_{1\text{-}15}(x_s, x_t)$, odpowiadającej wyrażeniu (3). Przy pomocy takiego odwzorowania możliwe jest określenie ośmiu funkcji:

$$f_{23}(x_r, x_s, x_t) = h_7(x_r, x_s) \wedge h_7(x_s, x_t) \wedge h_7(x_r, x_t)$$
$$f_{43}(x_r, x_s, x_t) = h_{11}(x_r, x_s) \wedge h_7(x_s, x_t) \wedge h_{11}(x_r, x_t)$$
$$f_{77}(x_r, x_s, x_t) = h_{13}(x_r, x_s) \wedge h_{11}(x_s, x_t) \wedge h_7(x_r, x_t)$$
$$f_{113}(x_r, x_s, x_t) = h_7(x_r, x_s) \wedge h_{13}(x_s, x_t) \wedge h_{13}(x_r, x_t)$$
$$f_{142}(x_r, x_s, x_t) = h_{14}(x_r, x_s) \wedge h_{11}(x_s, x_t) \wedge h_{11}(x_r, x_t)$$
$$f_{178}(x_r, x_s, x_t) = h_{11}(x_r, x_s) \wedge h_{13}(x_s, x_t) \wedge h_{14}(x_r, x_t)$$
$$f_{212}(x_r, x_s, x_t) = h_{13}(x_r, x_s) \wedge h_{14}(x_s, x_t) \wedge h_{13}(x_r, x_t)$$
$$f_{232}(x_r, x_s, x_t) = h_{14}(x_r, x_s) \wedge h_{14}(x_s, x_t) \wedge h_{14}(x_r, x_t)$$

Spośród 254 funkcji trzech zmiennych (pomijamy funkcje stałe ***TRUE*** i ***FALSE***), te które dają się odwzorować w postaci koniunkcji dwóch lub trzech funkcji dwóch zmiennych będziemy nazywać przywiedlnymi.

Wszystkie pozostałe funkcje, z pominięciem funkcji definiowanych przez *klauzule Horna* (funkcje f_{127}, f_{191}, f_{223}, f_{247}) i *coHorna* (funkcje f_{239}, f_{251}, f_{253}, f_{254}), mogą być wyrażone jako koniunkcja jednej odpowiednio dobranej funkcji przywiedlnej lub kilku takich funkcji przywiedlnych przez funkcję *Horna* i/lub *coHorna*. Na przykład, nieprzywiedlna funkcja f_{113} może być wyrażona przez koniunkcję funkcji $f_{117} \wedge f_{251}$ oraz koniunkcję funkcji $f_{115} \wedge f_{253}$, ale także przez koniunkcję funkcji $f_{245} \wedge f_{127} \wedge f_{251}$, gdzie f_{117}, f_{115} i f_{245} są przywiedlne, a f_{251}, f_{253} są funkcjami *coHorna*, natomiast f_{127} jest funkcją *Horna*.

W efekcie, uwzględniając analizowane związki między funkcjami dwóch i trzech zmiennych, możemy wyodrębnić następujące kategorie funkcji trzech zmiennych:

- funkcje przywiedlne wprost (przez koniunkcje funkcji dwóch zmiennych) lub
- funkcje przywiedlne pośrednio (przez koniunkcje funkcji przywidlnych wprost oraz funkcji *Horna* i/lub *coHorna*)
- funkcje *Horna* i *coHorna* (odpowiedniki *klauzul Horna i coHorna*).

Poczynione spostrzeżenie o możliwości wyrażenia grupy funkcji nieprzywiedlnych przez koniunkcje funkcji przywiedlnych i/lub funkcji *Horna* oraz *coHorna* pozwala rozróżniać następujące postaci formuł i tym samym odpowiednio klasyfikować **3SAT**:

- koniunkcja wyłącznie funkcji przywiedlnych i wtedy **3SAT** należy do ***P***,
- koniunkcja tylko funkcji *Horna* (**Horn3SAT**) lub tylko funkcji *coHorna* (**coHorn3SAT**) i wtedy **3SAT** należy do ***P***,
- koniunkcja funkcji *Horna* (**Horn3SAT**) i funkcji *coHorna* (**coHorn3SAT**),
- koniunkcja funkcji przywiedlnych, funkcji *Horna* i/lub funkcji *coHorna*.

Trzecia postać formuł, może w szczególnych przypadkach sprawiać (zmienne w funkcjach *Horna* i/lub *coHorna* tworzą rozłączne zbiory), że **3SAT** należy do ***P***. W ogólnym przypadku

takich postaci formuł (koniunkcja **Horn3SAT** i **coHorn3SAT** tworzy problem **MIX3SAT** definiowany w poprzednim rozdziale), konieczna jest bardziej pogłębiona analiza. W szczególności dotyczy to sugerowanego wcześniej składania wektorów rozwiązań obu spełnianych problemów **Horn3SAT** i **coHorn3SAT**.

Czwarta postać formuł, w istocie reprezentuje koniunkcję problemu **2SAT** i **MIX3SAT** (albo inaczej koniunkcję **2SAT**, **Horn3SAT** i **coHorn3SAT**, z których dwa ostanie są spełniane, a spełnialność **2SAT** łatwo sprawdzić). Tak więc, ta postać formuły jak i poprzednia, odpowiadają sytuacji, w której do rozstrzygania spełnialności **3SAT** w ogólnym przypadku powinniśmy dysponować algorytmem rozstrzygania spełnialności dla przypadku koniunkcji dwóch lub trzech spełnialnych niezależnie od siebie problemów należących do ***P***. Także w tym przypadku, konstrukcję algorytmu można oprzeć na poszukiwaniu odpowiedniego składania wektorów rozwiązań poszczególnych problemów.

Pewne sugestie w tym zakresie może przynieść analiza związków **2SAT** i **3SAT** na poziomie strukturalnym (w poprzednim rozdziale pokazano podobieństwo modeli referencyjnych obu problemów).

3.2.2 Przekształcenia strukturalne w *kSAT*

Rezultaty dyskusji w poprzednim punkcie, upoważniają do sformułowania stwierdzenia, że każdy problem **KSAT** można przekształcić do postaci **(K+1)SAT**, i odwrotnie, każdy tak otrzymany **(K+1)SAT** daje się przedstawić w ekwiwalentnej postaci **KSAT**. Podstawą takiego przekształcenia jest możliwość wyrażenia funkcji *k+1* zmiennych przez koniunkcję funkcji *k* zmiennych, w postaci $f^{k+1} = f^2(f^k(x_1, x_2, \dots, x_k), x_{k+1})$ lub innych, na przykład $f^{k+1} = f^2(f^k(x_1, x_2, \dots, x_{k-1}), f^k(x_2, x_3, \dots, x_{k+1}))$.

Wiemy także, że przekształcenie (redukcja) **(K+1)SAT** do **KSAT** jest możliwe tylko w pewnych określonych przypadkach (na przykład przekształcenie **3SAT** do **2SAT** jest możliwe tylko wtedy, gdy funkcje **3SAT** są przywiedlne).

Wiemy również, że przekształcenia **KSAT** do postaci **(K+1)SAT** mogą być realizowane na wiele sposobów. W dalszej dyskusji ograniczymy się do wzajemnych przekształceń problemów **2SAT** i **3SAT**, wykorzystując możliwość wyrażenia funkcji $f(x_r, x_s, x_t)$

trzech zmiennych (dla $r \in \{1, 2, \ldots n\text{-}2\}$, $s \in (r+1, r+2, \ldots, n\text{-}1\}$ i $t \in \{s+1, s+2, \ldots n\}$, tj. $r < s < t$) przez wyrażenia $f^3(f^2(x_r, x_s), f^2(x_r, x_t))$ lub $f^3(f^2(x_r, x_s), f^2(x_s, x_t))$ lub $f^3(f^2(x_r, x_t), f^2(x_s, x_t))$.

Rozważane będą dwa typy przekształceń, które dalej oznaczane będą jako P_H i P_V. Przekształcenie P_H wykorzystujące wyrażenie $f^3(f^2(x_r, x_s), f^2(x_r, x_t))$, ma zastosowanie do elementarnych struktur wierszowych problemu **2SAT**. Przekształcenie P_V wykorzystujące wyrażenia $f^3(f^2(x_r, x_t), f^2(x_s, x_t))$ ma zastosowanie do elementarnych struktur kolumnowych problemu **2SAT**.

W analizie przekształceń typu P_H ograniczymy się dowolną wierszową strukturą elementarną modelu referencyjnego *Full2SAT* opisaną formułą $\Theta_i = h(x_i, x_{i+1}) \wedge h(x_i, x_{i+2}) \wedge \ldots \wedge h(x_i, x_{n-1}) \wedge h(x_i, x_n)$ dla $i = 1, 2, \ldots, n\text{-}1$.

Funkcje **2SAT** oznaczać będziemy przez $h_a(x_i\ x_r)$, natomiast funkcje **3SAT** przez $f_b(x_i\ x_r\ x_s)$, gdzie $a \in \{0, \ldots, 15\}$ jest numerem funkcji dwóch zmiennych, natomiast $b \in \{0, \ldots, 255\}$ numerem funkcji trzech zmiennych. Jeśli będzie taka potrzeba, to w oznaczeniach funkcji posługiwać się będziemy dodatkowo górnym indeksem.

Zgodnie z przyjętą konwencją oznaczeń, w rozpatrywanych wariantach przekształceń P_H, formuła $\Theta_i = h(x_i, x_{i+1}) \wedge h(x_i, x_{i+2}) \wedge \ldots \wedge h(x_i, x_{n-1}) \wedge h(x_i, x_n)$ zawiera $(n\text{-}i)$ funkcji dwóch zmiennych o dowolnych numerach $a \in \{0, \ldots, 15\}$. Numery wynikające z uporządkowania funkcji w formule Θ_i uwzględnimy posługując się górnym indeksem w oznaczeniu funkcji.

Pierwszy wariant przekształcenia typu P_H realizowany jest w oparciu o zależność, którą w postaci ogólnej można przedstawić następująco (pamiętamy, że h oznacza dowolne funkcje dwóch zmiennych; f – pewne funkcje trzech zmiennych; b – dolny indeks określający funkcję trzech zmiennych; j – górny indeks określający kolejność funkcji w formule):

$$f_b^{\,j}(x_i\, x_{i+1}\, x_{j+2}) = h_a^{\,i}(x_i\, x_{i+1}) \wedge h_a^{\,j+1}(x_i\, x_{j+2}),$$

gdzie $f_b^{\,j}(x_i\, x_{i+1}\, x_{j+2})$ stanowią funkcje wynikowej formuły *3SAT*, oznaczanej przez Ω_i.

Przykładowo, dla Θ_1 otrzymamy funkcje:

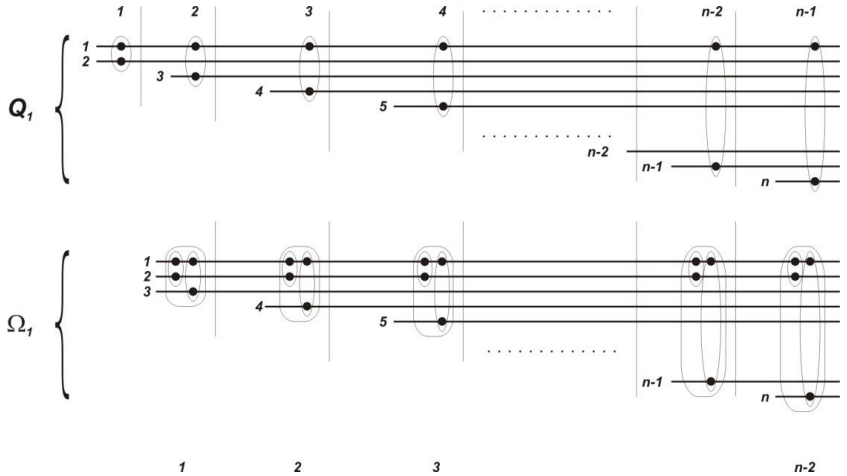
$$f_b^{\,1}(x_1\, x_2\, x_3) = h_a^{\,1}(x_1\, x_2) \wedge h_a^{\,2}(x_1\, x_3)$$
$$f_b^{\,2}(x_1\, x_2\, x_4) = h_a^{\,1}(x_1\, x_2) \wedge h_a^{\,3}(x_1\, x_4)$$

. . . . . .

$$f_b^{n-2}(x_1\, x_2\, x_n) = h_a^1(x_1\, x_2) \wedge h_a^{n-1}(x_1\, x_n).$$

Uzyskany w wyniku przekształcenia formuły Θ_1 maksymalnego problemu *Full2SAT* schemat formuły **3SAT**, prezentowany jest na rysunku 3.1. W tym wariancie przekształcenia P_H formuła wynikowa **3SAT** ma strukturę niepełnej formuły Ω_1 maksymalnego problemu *Full3SAT*. Liczebność funkcji w formule Ω_1 **3SAT** jest o jeden mniejsza od liczebności funkcji formuły Θ_1 **2SAT**, i będzie tak w przypadku każdej formuły Ω_i jako wyniku tego wariantu przekształcenia formuły Θ_i **2SAT**.

Na rysunku wyraźnie zaznaczono, że uzyskane w wyniku przekształcenia funkcje $f_b^j(x_1\, x_2\, x_{j+2})$ formuły Ω_1 **3SAT** zawierają wspólną funkcję $h_a^1(x_1\, x_2)$ formuły Θ_1 **2SAT**. Można to uogólnić na przypadek przekształcania dowolnej formuły Θ_i **2SAT** dla $i=1, \ldots, n\text{-}2$ (uwaga $\Omega_{n-2} = h_a^1(x_{n-2}, x_{n-1}) \wedge h_a^2(x_{n-2}, x_n) \wedge h_a^1(x_{n-1}, x_n)$). Wspólnymi funkcjami w Ω_i będą funkcje diagonalne $h_a^1(x_i, x_{i+1})$.

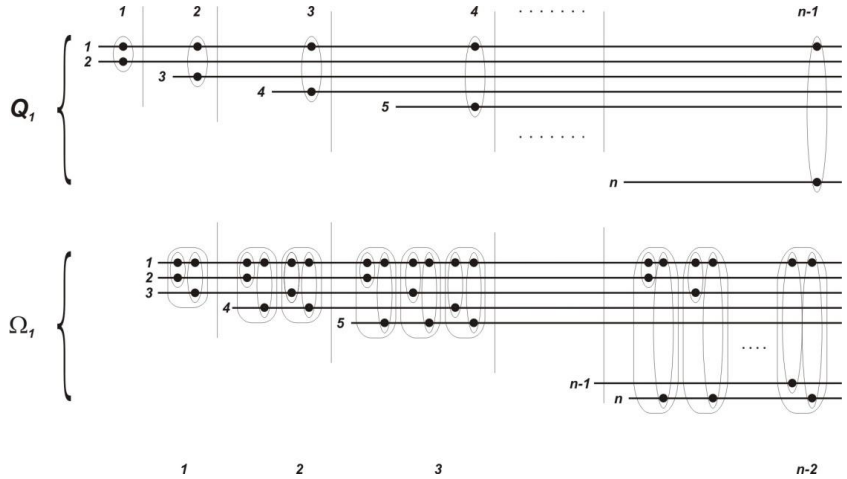


Rys. 3.1 Schemat przekształcenia typu P_H formuły Θ_1 *(i=1, 2, ... , n-1)* problemu *2SAT* do formuły Ω_1 problemu *3SAT*. Odpowiednio u góry i u dołu rysunku podano numery porządkowe funkcji *2SAT* (formuła Θ_1) oraz funkcji *3SAT* (formuła Ω_1).

Podobnie jak w pierwszym przypadku przekształcenia typu P_H, również w drugim przypadku, jako formuła wyjściowa rozważana będzie pewna formuła Θ_i **2SAT**. Ale teraz przekształ-

cenie do **3SAT** realizowane będzie tak, by uzyskać pełną postać formuły Ω_i określającą elementarną strukturę wierszową maksymalnego problemu *Full3SAT*. Oznacza to na przykład dla *i=1*, że dysponując *n-1* funkcjami formuły Θ_1 **2SAT**, formuła Ω_1 wyrażona będzie przez *(n-1)*(n-2)/2* funkcji trzech zmiennych (taka jest bowiem liczebność funkcji w formule Ω_1 maksy-malnego problemu *Full3SAT*).

Bez wprowadzenia dodatkowych oznaczeń, np. poprzez wprowadzenie numeracji podformuł zawierających funkcje o tych samych skrajnych numerach zmiennych, wyrażenie przekształcenia w postaci ogólnej jest kłopotliwe. Ograniczymy się więc do ilustracji graficznej w postaci schematu przedsta-wionego na rys. 3.2.



Rys. 3.2 Schemat przekształcenia typu P_H formuł Θ_i *(i=1, 2, ... , n-1)* dla problemu **2SAT** do formuły Ω_i **3SAT** na przykładzie zbioru Θ_1.

Podobnie jak na rys. 3.1, u góry i u dołu rysunku pokazano numery porządkowe. Tym razem, w wynikowej formule Ω_1 **3SAT** oznaczają one nie pojedyncze funkcje, ale grupy funkcji i odpowiadają drugiemu indeksowi podformuł $F_{i,j}$. w formułach Ω_i *Full3SAT*, zdefiniowanych w dyskusji modelu referencyjnego (rozdział II). Na przykład dla *i=1* $\Omega_1 = F_{1,1} \wedge F_{1,2} \wedge F_{1,3} \wedge \ldots \wedge F_{1,j} \wedge \ldots \wedge F_{1,n-2}$ oraz $\Theta_1 = h(x_1, x_2) \wedge h(x_1, x_3) \wedge \ldots \wedge h(x_1, x_{n-1}) \wedge h(x_1, x_n)$.

Istota tego przypadku przekształcenia typu P_H polega na takim konstruowaniu podformuł $\boldsymbol{F}_{i,j}$ formuły $\boldsymbol{\Omega}_i$, by dla ustalonego j, funkcje tworzące $\boldsymbol{F}_{i,j}$ powstały poprzez koniunkcję wszystkich poprzednich funkcji $h_{a^k}(x_i, x_k)$, $k=2, \ldots, j-1$ z funkcją $h_{a^j}(x_i, x_j)$ formuły $\boldsymbol{\Theta}_i$ **2SAT**.

W prezentowanym na rysunku 3.2 schemacie formuły $\boldsymbol{\Omega}_1$ **3SAT**, uzyskanym w wyniku przekształcenia formuły $\boldsymbol{\Theta}_1$ maksymalnego problemu *Full2SAT*, formuła wynikowa $\boldsymbol{\Omega}_1$ **3SAT** przekształcenia typu P_H ma strukturę pełnej formuły $\boldsymbol{\Omega}_1$ maksymalnego problemu *Full3SAT*.

Liczebność funkcji **3SAT** formuły $\boldsymbol{\Omega}_1$ wynosi $(n-1)*(n-2)/2$ i jest równa liczebność funkcji w formule $\boldsymbol{\Omega}_1$ maksymalnego problemu *Full3SAT*. Widać także, że ilość podformuł $\boldsymbol{F}_{1,j}$ w formule $\boldsymbol{\Omega}_1$ jest o jeden mniejsza od liczebności podgrup (ale także funkcji – bo podgrupy są jednoelementowe) w zbiorze $\boldsymbol{\Theta}_1$ **2SAT**. Ogólnie te zależności są prawdziwe dla dowolnego i.

3.2.3 Operacja przenumerowania zmiennych w KSAT

W konstrukcji algorytmów, mogą być użyte operacje przenumerowania zmiennych, przy czym operację przenumerowania można analizować w trzech aspektach: - w aspekcie zmiany porządku zmiennych w funkcji, - w aspekcie zmiany schematów wierszowego, kolumnowego i diagonalnego oraz w aspekcie całego *Full3SAT*. W istocie operacja przenumerowania zmiennych skutkuje ukrytą zmianą uporządkowania obszaru rozwiązań.

Operacja przenumerowania powoduje zmianę numeru funkcji, nie powodując zmiany jej rzędu. Można łatwo sprawdzić, że funkcje trzech zmiennych będące odwzorowaniem klauzul *Horna* w wyniku przenumerowania pozostają odwzorowaniem tej samej bądź innej klauzuli *Horna*. Podobnie jest dla funkcji odwzorowujących klauzule *coHorna*.

Przykładowo, operacja przenumerowania zmiennych funkcji $f_{95}(x_1\ x_2\ x_3)$ trzech zmiennych, w którym w wyniku przenumerowania zmienna x^*_1 odpowiada zmiennej x_3, zaś zmienne x^*_2 i x^*_3 zmiennym x_1 oraz x_2 odpowiednio, może być zilustrowana następująco:

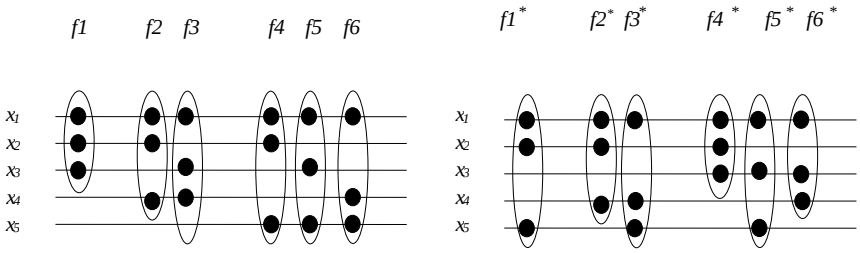
Zmienne i uporządkowany obszar rozwiązań przed operacją przenumerowania		Zmienne i nieuporządkowany obszar rozwiązań po operacji przenumerowania		Zmienne i uporządkowany obszar rozwiązań po operacji przenumerowania	
x_1	0 1 0 1 0 1 0 1	x^*_1	0 0 0 0 1 1 1 1	x^*_1	0 1 0 1 0 1 0 1
x_2	0 0 1 1 0 0 1 1	x^*_2	0 1 0 1 0 1 0 1	x^*_2	0 0 1 1 0 0 1 1
x_3	0 0 0 0 1 1 1 1	x^*_3	0 0 1 1 0 0 1 1	x^*_3	0 0 0 0 1 1 1 1
$f_{95}(x_1\, x_2\, x_3)$	1 1 1 1 1 0 1 0		1 1 1 1 1 0 1 0	$f_{119}(x^*_1\, x^*_2\, x^*_3)$	1 1 1 0 1 1 1 0

Operacja przenumerowania może być uogólniona na przypadek funkcji o dowolnych rozpiętościach δ_1 oraz δ_2 numerów jej zmiennych i wtedy skutkuje zmianą rozpiętości zmiennych po przenumerowaniu, a tym samym wpływa na uporządkowanie funkcji w strukturach **3SAT** i może wpływać na uporządkowanie zmiennych funkcji.

Przenumerowanie jest operacją wielomianową – dotyczy części funkcji spośród wszystkich występujących w *Full3SAT*. Wyjątek stanowi operacja pełnej inwersji numerów zmiennych – tj. $x_1 \Leftrightarrow x_n$, $x_2 \Leftrightarrow x_{n-1}$ itd. Wtedy operacja obejmuje wszystkie funkcje *Full3SAT*, ale i w tym przypadku pozostaje wielomianową.

Operacja przenumerowania może się okazać szczególnie przydatną przy konstruowaniu algorytmu dla **3SAT**, ponieważ może być wykorzystana do zamiany miejscami funkcji diagonalnej $\boldsymbol{d}_i$ z inną funkcją należącą do $\boldsymbol{\Omega}_i$, mającą niższy rząd. Wtedy będzie się ona odnosić zawsze tylko do dwóch zmiennych. Do zamiany z funkcją diagonalną $\boldsymbol{d}_i$, kwalifikują się te funkcje z $\boldsymbol{\Omega}_i$, których dwie pierwsze zmienne mają numery i oraz $i+1$ (takich funkcji w podformule $\boldsymbol{\Omega}_i$ modelu referencyjnym jest $n-i-2$) bądź numery i oraz $i+2$ (takich funkcji w podformule $\boldsymbol{\Omega}_i$ modelu referencyjnego jest także $n-i-2$). Należy pamiętać, że przenumerowanie będzie dotyczyć także zmiennych w pewnych funkcjach kolejnych podformuł $\boldsymbol{\Omega}_{i+1}, \ldots, \boldsymbol{\Omega}_{n-2}$ oraz wcześniejszych $\boldsymbol{\Omega}_1, \ldots, \boldsymbol{\Omega}_{i-1}$. Celem tej operacji w przypadku modelu referencyjnego *Full3SAT* jest uzyskanie możliwie najprostszej postaci podproblemu *Diag3SAT*.

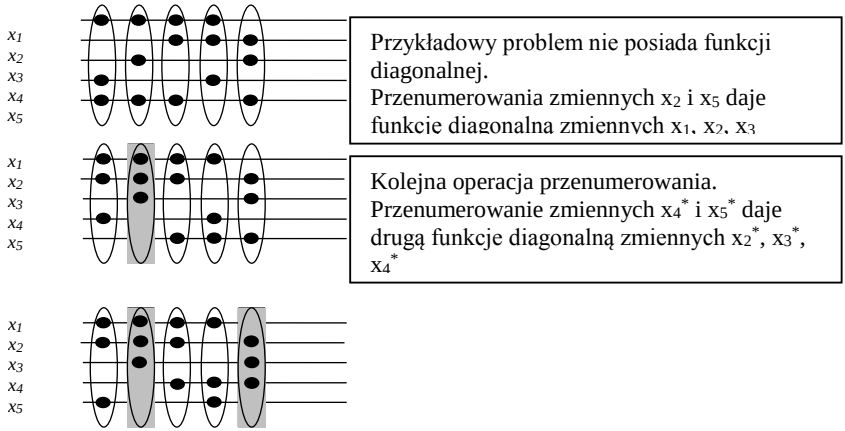
Przykład operacji przenumerowania zmiennych, w celu osiągnięcia efektu obniżenia rzędu funkcji diagonalnej w schemacie wierszowym ilustruje rysunek 3.3.



Rys. 3.3 Obniżenie rzędu funkcji diagonalnej w wyniku przenumerowania

Zakładamy, że funkcja *f1* ma rząd równy 5 (pięć elementarnych koniunkcji – mogą to być na przykład funkcje f_{31}, f_{79}, f_{143}), natomiast funkcja *f4* ma rząd równy 1 (jedna elementarna koniunkcja - mogą to być na przykład funkcje f_1, f_2, f_8, f_{128}). Jeśli dokonamy przenumerowania x_3 z x_5, uzyskamy nową funkcję diagonalną $d1^* = f4^*$, której rząd będzie równy 1. Na rysunku widoczna jest zmiana pierwotnego uporządkowania funkcji w schemacie wierszowym po wykonaniu operacji przenumerowania.

Kolejny przykład zastosowania operacji przenumerowania zmiennych przedstawia rysunek 3.4.



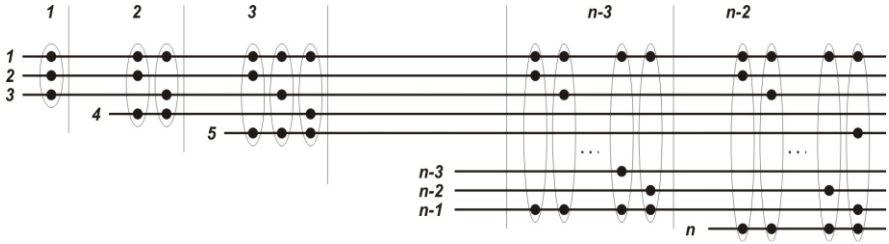
Rys. 3.4 Operacja przenumerowania dla problemu bez funkcji diagonalnej.

Wyjściowy problem charakteryzuje się brakiem funkcji diagonalnej. W wyniku wykonania sekwencji operacji przenu-

merowania zmiennych, w wynikowej postaci problemu pojawiają się dwie funkcje diagonalne.

3.3 Analiza schematu wierszowego w *3SAT*

Formuła **3SAT** mającego postać struktury wierszowej, ma własność, która pozwala w czasie wielomianowym uzyskać rozwiązanie zarówno dla problemu decyzyjnego jak i obliczeniowego. Zostanie to pokazane jest na przykładzie **3SAT** zdefiniowanego przez $\boldsymbol{\Omega_1}$. W prowadzonej dyskusji pomocna może być reprezentacja graficzna (rys. 3.5). W każdej chwili rozważania te można odnieść do przypadku dowolnego **3SAT** zdefiniowanego przez $\boldsymbol{\Omega_i}$ ($i=1, 2, \ldots, n-2$).



Rys. 3.5 Schemat wierszowy *3SAT*.

Jak wcześniej zostało to zasygnalizowane, w formule $\boldsymbol{\Omega_1}$ (a ogólnie w $\boldsymbol{\Omega_i}$) można wyodrębnić pewne grupy funkcji. Oznaczyliśmy je jako podformuły $\boldsymbol{F}_{i,j}$ dla $i = 1, \ldots, n-2$ oraz $j \in \{i, \ldots, n-2\}$ w zapisie formuły *Full3SAT* pozwalającej postrzegać cały problem w kategoriach tablicy trójkątnej górnej. W przypadku $\boldsymbol{\Omega_1}$ formuł $\boldsymbol{F}_{1,j}$ będzie $M_1 = n-2$ (a ogólnie $M_i = n-i-1$). O przynależności do grupy decyduje δ_c rozpiętość całkowita. W każdej tak wyodrębnionej grupie, począwszy od drugiej, będą po dwie funkcje zależne od dwóch zmiennych występujących w funkcji diagonalnej. W każdej z pozostałych funkcji wystąpi pierwsza zmienna funkcji diagonalnej $d_1 = f(x_1, x_2, x_3)$.

Do rozwiązania **3SAT** mającego postać schematu wierszowego $\boldsymbol{\Omega_i}$ ($i=1, \ldots, n-2$), można zastosować kilka strategii, np.:

- rozpatrzeć dwa przypadki wartościowania zmiennej x_i (alternatywnie mogą to być x_{i+1} albo x_{i+2}),

- rozpatrzeć k przypadków (k – rząd funkcji diagonalnej), zakładając wartościowania zmiennych x_i, x_{i+1}, x_{i+2} określone przez kolejne elementarne koniunkcje funkcji diagonalnej,
- rozpatrzeć dwa przypadki wartościowania ostatniej zmiennej x_n,
- rozpatrzeć maksymalnie cztery przypadki uwzględniające wartościowania zmiennej x_i (alternatywnie x_{i+1} albo x_{i+2}) oraz zmiennej x_n.

Doboru strategii dla **3SAT** z Ω_i można dokonać na przykład w oparciu o wprowadzony współczynnik wypełnienia modelu referencyjnego. Pokazanie, że **3SAT** $\equiv \Omega_i$ w wersji decyzyjnej należy do P jest proste i dotyczy to także Ω_i mających postaci „zdegenerowane", np. – brak funkcji tworzących określoną jedną lub kilka podformuł $F_{i,j}$, a w szczególności kiedy każdą z podformuł $F_{i,j}$ definiuje tylko jedna funkcja. Wystarczy zauważyć, że znajomość (lub założenie) wartościowania zmiennej x_i poprzez jej podstawienie do wyrażeń funkcji tworzących Ω_i przekształca nasz **3SAT** w **2SAT** $\in P$.

Nieco trudniej pokazać, że rozwiązania w wersji obliczeniowej (jako zwrot wektora wartościowań zmiennych) także jest wielomianowe. Dyskusję tego przypadku rozpoczniemy od **3SAT** $\equiv \Omega_1$ stosując strategię zakładającą rozpatrzenie k przypadków (k – rząd funkcji diagonalnej) wartościowania zmiennych x_1, x_2 i x_3.

Niech $f(x_1,x_2,x_3)$ będzie pewną funkcją f_α zapisaną w *DNF*, gdzie α – numer funkcji jednoznacznie określa ilość i postać elementarnych koniunkcji w jej postaci *DNF*.

Niech c_j oznacza taką elementarną koniunkcję, $j \in \{1, \ldots, k\}$, wtedy

$$f_\alpha (x_1,x_2,x_3) = c_1 \vee c_2 \vee \ldots \vee c_k, \text{ gdzie } k \text{ jest rzędem funkcji } (k \leq 8)$$

Niech pozostałe funkcje w Ω_1 będą oznaczone Ω_1^* (przy czym Ω_1^* pozostaje nadal *3SAT*), wtedy

$$\Omega_1 = f_\alpha \wedge \Omega_1^* = (c_1 \wedge \Omega_1^*) \vee (c_2 \wedge \Omega_1^*) \vee \ldots \vee (c_k \wedge \Omega_1^*)$$

Ta postać pozwala rozpatrywać **niezależnie** k problemów **3AT**, a każdy z nich jest wielomianowy, tak w wersji decyzyjnej jak i obliczeniowej. Elementarne koniunkcje c_j, $j \in \{1, \ldots, k\}$ określają ustalone wartościowanie trzech pierwszych zmiennych x_1,x_2,x_3 naszego problemu **3SAT**.

Ponieważ część funkcji tworzących Ω_1^* zależy od dwóch par zmiennych (x_1,x_2 lub x_1,x_3), a pozostałe od zmiennej x_1, oraz ponieważ sposób wartościowania tych trzech zmiennych funkcji diagonalnej $f_\alpha\ (x_1,x_2,x_3)$ jest ustalony przez rozpatrywaną w danych przypadku elementarną koniunkcję c_s dla $s \in \{1, \ldots, k\}$, to każda funkcja tworząca Ω_1^* może być niezależnie sprawdzona na spełnialność. Sprawdzanie będzie polegało na podstawieniu dla występujących w tych funkcjach zmiennych x_1, x_2 oraz x_3, wartościowań tych trzech zmiennych ustalonych przez rozpatrywaną elementarna koniunkcję c_s. Proces sprawdzania będzie realizowany w czasie wielomianowym – funkcji w Ω_1^* jest mniej niż $(n-1)(n-2)/2$.

Spełnialność każdej funkcji Ω_1^* badana niezależnie nie gwarantuje jednak spełnialności całej Ω_1 – może być potraktowana jako warunek konieczny spełnialności naszego **3SAT**.

Aby ominąć wskazaną trudność, proces weryfikacji spełnialności Ω_1^* przy założeniu znajomości wartościowania określonego przez rozpatrywaną elementarną koniunkcję c_s, $s \in \{1, \ldots, k\}$ prowadzony będzie w nieco odmienny sposób.

Zauważmy (wykorzystując wiedzę o strukturze grup funkcji $F_{i,j}$) , że Ω_1^* może być wyrażony w postaci

$$\Omega_1^* = \varphi_1 \wedge \varphi_2$$

gdzie:

$$\varphi_1 = f(x_1, x_2, x_4) \wedge f(x_1, x_2, x_5) \wedge \ldots \wedge f(x_1, x_2, x_j) \wedge \ldots \wedge f(x_1, x_2, x_{n-1}) \wedge f(x_1, x_2, x_n) \wedge f(x_1, x_3, x_4) \wedge f(x_1, x_3, x_5) \wedge \ldots \wedge f(x_1, x_3, x_j) \wedge \ldots \wedge f(x_1, x_3, x_{n-1}) \wedge f(x_1, x_3, x_n)$$

oraz

$$\begin{aligned}\varphi_2 = {} & f(x_1, x_4, x_5) \\ & \wedge f(x_1, x_4, x_6) \wedge f(x_1, x_5, x_6) \wedge \\ & \wedge f(x_1, x_4, x_7) \wedge f(x_1, x_5, x_7) \wedge f(x_1, x_6, x_7) \\ & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ & \wedge f(x_1, x_4, x_{n-1}) \wedge f(x_1, x_5, x_{n-1}) \wedge \ldots \wedge f(x_1, x_{n-2}, x_{n-1}) \\ & \wedge f(x_1, x_4, x_n) \wedge f(x_1, x_5, x_n) \wedge \ldots \wedge f(x_1, x_j, x_n) \wedge \ldots \wedge f(x_1, x_{n-2}, x_n) \wedge f(x_1, x_{n-1}, x_n)\end{aligned}$$

W φ_1 jest liniowa względem n ilość funkcji - dla naszego **3SAT** ≡ Ω_1 wyraża się jako $M_1 = 2\cdot(n-3)$. W φ_2 jest wielomianowa względem n ilość funkcji - dla naszego **3SAT** ≡ Ω_1 wyraża się jako $M_2 = (n-1)\cdot(n-2)/2 - M_1 - 1$. Przy znajomości wartościowań zmiennych x_1, x_2, x_3 , sprawdzanie sekwencyjnie na spełnialność każdej funkcji z $\varphi_1 \equiv \Omega_1$

zwraca albo ustalone wartościowanie trzeciej zmiennej – jedno z dwóch (prawda - *true* albo fałsz - *false*), bądź określa, że oba wartościowania są dopuszczalne. Możliwe jest oczywiście stwierdzenie, że nie istnieje takie wartościowanie trzeciej zmiennej by funkcja była spełnialna. Wyniki prowadzonej weryfikacji zapisywane są w wektorze rozwiązań o długości *n*. Wynik weryfikacji każdej funkcji, zapisywany jest na odpowiedniej pozycji wektora, odpowiadającej numerowi trzeciej zmiennej – *0* albo *1* gdy wartościowanie jest określone jednoznacznie, symbol * gdy dopuszczalne jest dowolne wartościowanie. Brak warunków spełnialności można odnotować specjalnym symbolem np. #.

Przy znajomości wartościowania zmiennej x_1, φ_2 staje się **2SAT**, a ten jest w ***P*** i sprawdzanie sekwencyjnie każdej funkcji z φ_2 zwraca - zależnie od jej rzędu - albo ustalone wartościowanie obu zmiennych – jedno z dwóch (prawda - *true* albo fałsz - *false*), albo tylko drugiej lub trzeciej zmiennej, bądź określa, że oba wartościowania są dopuszczalne dla drugiej i trzeciej zmiennej. Możliwe jest oczywiście stwierdzenie, że nie istnieje takie wartościowanie jej zmiennych by funkcja była spełnialna.

Po weryfikacji φ_1, wektor rozwiązań może mieć postać:

- dla wszystkich zmiennych $x_4, x_5, \ldots, x_n$ jest ustalone wartościowanie,
- dla części zmiennych spośród $x_4, x_5, \ldots, x_n$ jest ustalone wartościowanie, dla pozostałych dowolne,
- dla wszystkich zmiennych $x_4, x_5, \ldots, x_n$ dopuszczalne jest dowolne wartościowanie,
- wystąpił przypadek braku spełnialności.

W pierwszym przypadku uzyskujemy bezpośrednio możliwość sprawdzenia spełnialności funkcji tworzących φ_2 (jest ich wielomianowa ilość).

W trzecim przypadku pozostaje nam rozwiązać problem **2SAT** zdefiniowany przez φ_2, traktując go jako całkowicie niezależny od x_1, x_2, x_3 i konkatenując jego rozwiązanie z ustalonym wartościowaniem zmiennych x_1, x_2, x_3.

Czwarty przypadek, dla problemu decyzyjnego zwraca odpowiedź ***NIE*** (brak spełnialności **3SAT**).

Drugi przypadek pozwala wprost weryfikować część funkcji tworzących φ_2, a dla tych niepodlegających weryfikacji wprost,

pozwala zdefiniować w oparciu o nie nowy problem $\varphi_2^* \equiv$ **2SAT**. Zdefiniowany w ten sposób nowy problem φ_2^* nie jest jednak niezależny. Niezależność φ_2^* gwarantuje wyłącznie sytuacja, gdy zmienne dopuszczające wartościowania stanowią sekwencję ciągłą od pewnej zmiennej x_s aż do x_n. Ilustruje to poniższy przykład.

Niech w wyniku sekwencyjnego sprawdzania funkcji z φ_1, wektor rozwiązań ma postać (powyżej podane są odpowiednie zmienne),

x_1	x_2	x_3	. . .	x_{k-1}	x_k	x_{k+1}	. . .	x_{s-1}	x_s	x_{s+1}	. . .	x_n
1	**0**	**0**	**. . .**	**1**	*	*	**. . .**	**0**	*	*	**. . .**	**1**

w której zmienne od x_1 do x_{k-1} mają ustalone jednoznaczne wartościowanie oraz niech φ_2 będzie przykładowo określone następująco:

$$\varphi_2 = f_{40}(x_1, x_k, x_s) \wedge f_{130}(x_1, x_{k+1}, x_s) \wedge f_{40}(x_1, x_k, x_{s+1}) \wedge f_{136}(x_1, x_s, x_{s+1}) \wedge f_{34}(x_1, x_{k+1}, x_n)$$

gdzie dla x_1=*true:*

$$f_{40}(x_1, x_k, x_s) = (x_1\, x_k\, \bar{x_s} \vee x_1\, \bar{x_k}\, x_s) \equiv (x_k\, \bar{x_s} \vee \bar{x_k}\, x_s) = h_6(x_k, x_s)$$
$$f_{130}(x_1, x_{k+1}, x_s) = (x_1\, \bar{x_{k+1}}\, \bar{x_s} \vee x_1\, x_{k+1}\, x_s) \equiv (\bar{x_{k+1}}\, \bar{x_s} \vee x_{k+1}\, x_s) = h_9(x_{k+1}, x_s)$$
$$f_{40}(x_1, x_k, x_{s+1}) = (x_1\, x_k\, \bar{x_{s+1}} \vee x_1\, \bar{x_k}\, x_{s+1}) \equiv (x_k\, \bar{x_{s+1}} \vee \bar{x_k}\, x_{s+1}) = h_6(x_k, x_{s+1})$$
$$f_{136}(x_1, x_s, x_{s+1}) = (x_1\, \bar{x_s}\, \bar{x_{s+1}} \vee x_1\, x_s\, x_{s+1}) \equiv (\bar{x_{k+1}}\, \bar{x_s} \vee x_{k+1}\, x_s) = h_{10}(x_s, x_{s+1})$$
$$f_{34}(x_1, x_{k+1}, x_n) = (x_1\, \bar{x_{k+1}}\, \bar{x_n} \vee x_1\, \bar{x_{k+1}}\, x_n) \equiv (\bar{x_{k+1}}\, \bar{x_n} \vee \bar{x_{k+1}}\, x_n) = h_5(x_{k+1}, x_n)$$

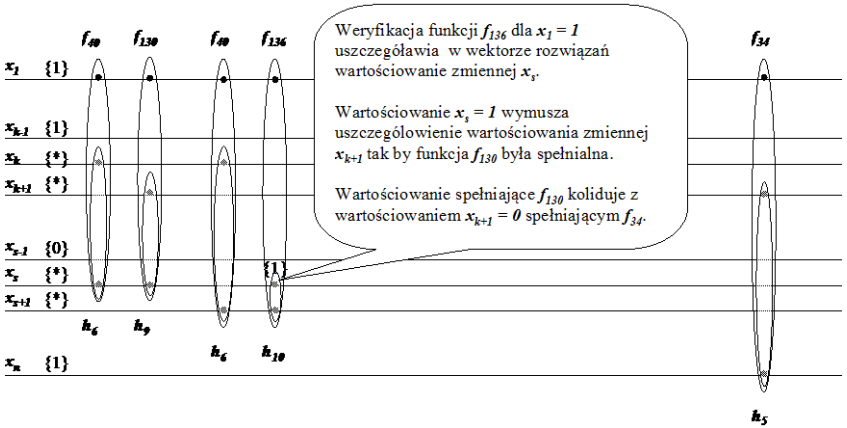
Przy znajomości wartościowania zmiennej x_1, funkcje trzech zmiennych z φ_2 redukują się do funkcji dwóch zmiennych (ich postać pokazana jest po prawej stronie znaku tożsamości) i wtedy φ_2 ma postać **2SAT**.

$$\varphi_2 \equiv \textbf{2SAT} = h_6(x_k, x_s) \wedge h_9(x_{k+1}, x_s) \wedge h_6(x_k, x_{s+1}) \wedge h_{10}(x_s, x_{s+1}) \wedge h_5(x_{k+1}, x_n)$$

Dokonywana w oparciu o wektor rozwiązań sekwencyjna weryfikacja koniunkcji trzech pierwszych funkcji $h_6(x_k, x_s) \wedge h_9(x_{k+1}, x_s) \wedge h_6(x_k, x_{s+1})$ jest pozytywna. Także funkcja h_{10} gwarantuje spełnialność formuły φ_2, ale z wartościowaniem $x_s = 1$, co powoduje, że $h_9(x_{k+1}, x_s)$ będzie spełniana tylko dla $x_{k+1} = 1$, a nie tak jak zapisane w wektorze rozwiązań dowolne wartościowanie. Gdyby proces weryfikacji był kontynuowany bez uwzględnienia tej sytuacji, to stwierdzalibyśmy spełnialnośc ostatniej funkcji h_5

(x_{k+1}, x_n) i tym samym spełnialnośc φ_2 i całego $\mathbf{3SAT} \equiv \Omega_1$, co nie jest prawdą.

Dyskutowany proces weryfikacji przedstawiony jest na rys. 3.6.



Rys. 3.6 Ilustracja procesu weryfikacji formuły φ_2 dla rozpatrywanego przykładu.

Powodem występowania takich sytuacji jest różnorodność możliwych sposobów wartościowania drugiej i trzeciej zmiennej funkcji trzech zmiennych przy ustalonym wartościowaniu pierwszej zmiennej.

Przy znajomości wartościowania pierwszej zmiennej istnieją cztery możliwe sposoby wartościowania drugiej i trzeciej zmiennej.

Druga zmienna	0 1 0 1
Trzecia zmienna	0 0 1 1

Prowadząc rozważania na poziomie funkcji trzech zmiennych, trzeba brać pod uwagę wszystkie możliwe kombinacje tych czterech sposobów wartościowania drugiej i trzeciej zmiennej– kombinacje $\binom{4}{1}$, $\binom{4}{2}$, $\binom{4}{3}$ i $\binom{4}{4}$.

Dla kombinacji $\binom{4}{1}$ sposobów wartościowania drugiej i trzeciej zmiennej, co odpowiada funkcjom dwóch zmiennych rzędu 1 (jedna elementarna koniunkcja), do wektora rozwiązań zwracane jest jednoznaczne wartościowanie drugiej i trzeciej zmiennej.

Dla kombinacji $\binom{4}{2}$ sposobów wartościowania drugiej i trzeciej zmiennej, zachodzić będzie sytuacja obrazowana następująco:

Możliwe sposoby wartościowania drugiej i trzeciej zmiennej weryfikowanej funkcji przy ustalonym wartościowaniu pierwszej zmiennej						
Druga zmienna	0 1	0 0	0 1	1 0	1 1	0 1
Trzecia zmienna	0 0	0 1	0 1	0 1	0 1	1 1
Postać zwracana do wektora rozwiązań						
Druga zmienna	*	0	*	*	1	*
Trzecia zmienna	0	*	*	*	*	1

Dla kombinacji $\binom{4}{3}$ i $\binom{4}{4}$ sposobów wartościowania drugiej i trzeciej zmiennej, do wektora rozwiązań zwracane są dowolne wartościowania. Dla porządku, przywołane są w tabeli poniżej.

Możliwe sposoby wartościowania drugiej i trzeciej zmiennej weryfikowanej funkcji przy ustalonym wartościowaniu pierwszej zmiennej					
Druga zmienna	0 1 0	0 1 1	0 0 1	1 0 1	0 1 0 1
Trzecia zmienna	0 0 1	0 0 1	0 1 1	0 1 1	0 0 1 1
Postać zwracana do wektora rozwiązań					
Druga zmienna	*	*	*	*	*
Trzecia zmienna	*	*	*	*	*

Wiedząc, jaka postać wartościowania drugiej i trzeciej zmiennej zwracana jest przy znajomości wartościowania pierwszej zmiennej, można weryfikację spełnialności φ_2 prowadzić sukcesywnie dla kolejnych jej funkcji, zaczynając od funkcji o najmniejszej rozpiętości całkowitej.

Weryfikacja każdej kolejnej funkcji polega na sprawdzeniu czy funkcja może być spełniana dla ustalonego wartościowania zmiennej x_1 i jeśli tak, to określamy postać wartościowania drugiej i trzeciej zmiennej. Jeśli wynikiem jest jednoznaczne wartościowanie drugiej lub trzeciej albo obu zmiennych to kontrolujemy je z zapisami w wektorze rozwiązań – jeśli są zgodne, przechodzimy do weryfikacji kolejnej funkcji, - jeśli nie są zgodne to albo uszczegóławiamy zapis wektora rozwiązań, np. zastępując symbol(e) * dowolnego wartościowania jednoznacznym wartościowaniem, by móc uwzględnić go przy weryfikacji kolejnych funkcji albo stwierdzamy brak spełnialności naszego problemu.

J E D N A K uszczegółowienie rozpatrywanej zmiennej może powodować konieczność uszczegółowienia wartościowania

innych wcześniejszych zmiennych, które w wektorze rozwiązań są oznaczone symbolem dowolnego wartościowania. Może to wynikać z powiązań występujących między zmiennymi we wcześniejszych funkcjach (zwróciliśmy na to uwagę, wskazując możliwość niesekwencyjnego ustalania warunkowego wartościowania zmiennych funkcji trzech zmiennych). Z tych powodów proces weryfikacji, po operacji uszczegółowienia trzeba ponowić od początku, zaczynając od pierwszej funkcji w φ_2.

Całość procedury realizowana jest w czasie wielomianowym – ilość funkcji w φ_2 jest wielomianowa (wynosi $M_2 = (n-1)\cdot(n-2)/2 - M_1 + 1$), możliwa ilość operacji uszczegółowienia nie będzie większa od n ilości zmiennych.

Główne kroki procesu weryfikacji problemu **3SAT** = Ω_1 można opisać w postaci procedury (zakładamy, że formuły φ_1 i φ_2 są uporządkowane):

dla każdej elementarnej koniunkcji funkcji diagonalnej $d_1(x_1, x_2, x_3)$ wykonuj:

(1) przy znanych wartościowaniach zmiennych x_1, x_2, x_3 dla kolejnych funkcji φ_1 badaj ich spełnialność
jeśli spełnialna - ustal sposób wartościowania trzeciej zmiennej wstawiając odpowiednio 0, 1 * w wektorze rozwiązań na pozycji odpowiadającej numerowi trzeciej zmiennej rozpatrywanej funkcji, inaczej pisz komunikat BRAK SPEŁNIALNOŚCI i przejdź do weryfikacji wg kolejnej elementarnej koniunkcji

(2) rozpatrz przypadki postaci wektora rozwiązań

(2.1) gdy wszystkie zmienne $x_4, \ldots, x_n$ mają jednoznaczne wartościowanie 0 lub 1 wykonuj:
dla kolejnych funkcji z φ_2 badaj spełnialność wg wartościowań z wektora rozwiązań, jeśli brak spełnialności pisz komunikat BRAK SPEŁNIALNOŚCI i przejdź do weryfikacji wg kolejnej elementarnej koniunkcji

(2.2) gdy część lub wszystkie zmienne mają dowolne wartościowanie wykonuj:
dla kolejnych funkcji z φ_2 badaj spełnialność i kontroluj z zapisem w wektorze rozwiązań, jeśli brak spełnialności pisz komunikat BRAK SPEŁNIALNOŚCI, inaczej jeśli zgodne kontynuuj

inaczej zmień zapis w wektorze rozwiązań i rozpocznij badanie φ_2 od początku.

Opisany sposób postępowania dla **3SAT** mającego strukturę wierszową, można zastosować do **3SAT** skonstruowanego z trzech kolejnych schematów wierszowych Ω_1, Ω_2 oraz Ω_3. Problem φ_1 (jako **1SAT**) obejmie dodatkowo dla $j=4, \ldots, n$ funkcje $f(x_2, x_3, x_j)$ z Ω_2. Także φ_2 (jako **2SAT**) powiększy się o dodatkowe funkcje. Mimo powiększenia ilości funkcji w tak skonstruowanym problemie – realizacja procedury będzie się odbywać w czasie wielomianowym.

Problem φ_2 w tym przypadku, opisany jest przez koniunkcję trzech sekwencji funkcji. Pierwszą sekwencję tworzą funkcje mające ustalone wartościowanie zmiennej x_1. Drugą sekwencję tworzą funkcje mające ustalone wartościowanie zmiennej x_2. Trzecią sekwencję tworzą funkcje mające ustalone wartościowanie zmiennej x_3 (te ustalone wartościowania zmiennych określa rozpatrywana elementarna koniunkcja funkcji diagonalnej d_1).

Wszystko to sprawia, że każda z tych sekwencji może być rozpatrywana w kategoriach problemu **2SAT**. A z racji tego, że po każdej ewentualnej operacji uszczegóławiania wartościowania zmiennych w wektorze rozwiązań, weryfikowanie spełnialności funkcji ponawiane jest od pierwszej funkcji tworzącej φ_2 – kolejność uszczegóławiania wartościowania zmiennych nie odgrywa istotnej roli. Może być tak, że uszczegółowienie zmiennej x_4 nastąpi w końcowej fazie weryfikacji spełnialności przez ostatnią w trzeciej sekwencji funkcję $f_\alpha(x_3, x_{n-1}, x_n)$.

W odniesieniu do opisanej procedury, pojedynczą operację uszczegóławiania wartościowania konkretnej zmiennej można utożsamiać z przemieszczeniem funkcji (jednej bądź nawet kilku) ze zbioru funkcji tworzących $\varphi_2 \equiv$ **2SAT** do zbioru funkcji stanowiących $\varphi_1 \equiv$ **1SAT**. Postrzeganie w ten sposób poszczególnych faz procedury weryfikacji może być pomocne przy opisie próby uogólnienia tej procedury na przypadek *Full3SAT*, jeśli przedstawimy go w postaci

$$Full3SAT = \varphi_1 \wedge \varphi_2 \wedge \varphi_3$$

gdzie: $\varphi_1 \equiv$ **1SAT**, $\varphi_2 \equiv$ **2SAT** i są skonstruowane ze schematów wierszowych Ω_1, Ω_2, Ω_3 oraz $\varphi_3 \equiv$ **3SAT** $= \Omega_4 \wedge \Omega_5 \wedge \ldots \wedge \Omega_{i-1} \wedge \Omega_i \wedge \Omega_{i+1} \wedge \ldots \wedge \Omega_{n-3} \wedge \Omega_{n-2}$

W sytuacji opisanej w ten sposób - dla *Full***3SAT** każda operacja uszczegółowienia wartościowania dokonana w oparciu o weryfikację $\varphi_2 \equiv$ **2SAT**, będzie skutkowała oprócz przemieszczenia funkcji (jednej lub kilku) ze zbioru tworzącego **2SAT** do zbioru tworzącego $\varphi_1 \equiv$ **1SAT**, także przemieszczeniem funkcji (jednej lub kilku) ze zbioru funkcji tworzących $\varphi_3 \equiv$ **3SAT** do zbioru funkcji tworzących $\varphi_2 \equiv$ **2SAT**, a niekiedy (zależnie od rzędu funkcji) wprost do $\varphi_1 \equiv$ **3SAT**.

Wykorzystując sekwencyjną weryfikację funkcji, wobec zasygnalizowanej możliwej niesekwencyjności uszczegóławiania wartościowań zmiennych, w ogólnym przypadku (począwszy od Ω_4) tracimy „komfort" posługiwania się schematem zakładającym znajomość pierwszej zmiennej w funkcjach. Rozwiązaniem tej trudności może być adaptowanie tablic wartościowania warunkowego na przypadek znajomości drugiej i trzeciej zmiennej, a także wykonanie operacji przenumerowania zmiennych.

Największą trudność stanowi przypadek, gdy w wyniku zakończenia procedury weryfikacji realizowanej w czasie wielomianowym – (ilość przeglądanych funkcji w sekwencjach φ_2 i φ_3 funkcji, łącznie jest mniejsza od *M* ilości funkcji w całym *Full3SAT*, a ilość operacji uszczegóławiania jest mniejsza od ilości *n* zmiennych) - część funkcji nie będzie miała ustalonego wartościowania żadnej z jej trzech zmiennych i nadal „pozostanie" w $\varphi_3 \equiv$ **3SAT**. Tego typu trudność nie występuje przy realizacji procedury dla problemu mającego postać schematu wierszowego **3SAT** = Ω_1, czy jego rozszerzenia na trzy schematy wierszowe **3SAT** = $\Omega_1 \wedge \Omega_2 \wedge \Omega_3$, bo wtedy funkcje „pozostające" w φ_2 tworzą przecież **2SAT**.

Jeśli dla rozpatrywanego problemu opisanego jako *Full***3SAT** = $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$ wystąpi sytuacja, że część zmiennych nie będzie miała ustalonego wartościowania i nadal „pozostanie" w $\varphi_3 \equiv$ **3SAT**, to mogą zachodzić przypadki, które opiszemy w kategoriach zbiorów funkcji tworzących φ_1, φ_2, φ_3.

Jeśli po zakończeniu procedury sekwencyjnego (z nawrotami po każdorazowym uszczegółowieniu) weryfikowania całego *Full3SAT* nie ma funkcji, które tworzą $\varphi_2 \equiv$ **2SAT**, to oznacza , że te które nie należą do tego „pozostałego" φ_3 należą do $\varphi_1 =$ **1SAT**. Pozwala to rozwiązać ten „pozostały" $\varphi_3 \equiv$ **3SAT** całkowicie

niezależnie i jego rozwiązania konkatenować z wcześniej ustalonymi wartościowaniami zmiennych **1SAT**.

W sytuacji, gdy φ_2 nie jest pusty, rozwiązujemy **2SAT**, a uzyskane rozwiązanie możemy wykorzystać do sprawdzenia „pozostałego" φ_3. Ilość funkcji jest wielomianowa, nie wiemy jednak czy ilość zwróconych rozwiązań **2SAT** będzie wielomianowa. Poza tym zmienne φ_2 nie koniecznie będą pokrywały wszystkie zmienne φ_3.

Wydaje się, że jest lepsze rozwiązanie, polegające na doprowadzeniu do „wyzerowania" φ_2. Spowodujemy „włożenie" każdej funkcji φ_2, traktowanych jako funkcje dwóch zmiennych, do odpowiednich funkcji w φ_3 - utworzymy zmodyfikowany φ_3^*. Pomysł polega na wyznaczeniu iloczynów logicznych każdej funkcji h_a z φ_2 (funkcje dwóch zmiennych) z funkcjami φ_3 (funkcje trzech zmiennych), w których występują pary zmiennych funkcji h_a. Możliwy będzie przy tym efekt redukcji rzędu funkcji tworzących zmodyfikowaną $\varphi_3^* = \mathbf{3SAT}$.

W ocenie skuteczności rozpatrywanego sposobu rozwiązania problemu *Full***3SAT**, wykorzystującego uogólnienie wielomianowej procedury dla schematu wierszowego (wielomianowego także dla jego postaci rozszerzonej do trzech schematów wierszowych), napotykamy pewną barierę. Tą barierą jest fakt stosowania operacji konkatenacji – może to sugerować konieczność posługiwania się strukturą drzewiastą. Właśnie ilość poziomów konkatenacji będzie decydowała o tym, czy rozwiązanie uzyskamy w czasie wielomianowym. Wiemy teraz tylko tyle, że wykorzystanie pomysłu „wkładania" funkcji dwóch zmiennych w funkcje trzech zmiennych, sprzyja redukcji rzędu tych ostatnich. Póki co, pytanie w jakim stopniu i jak szybko postępuje ta redukcja w kolejnych koniecznych do zastosowania konkatenacjach, wymaga dodatkowej analizy. Niewątpliwie wpływ na efektywność całej procedury w przypadku jej uogólnienia na *Full***3SAT** ma wpływ rząd funkcji diagonalnych – im ich rząd będzie mniejszy tym nawet konieczność zastosowania mechanizmu konkatenacji na kilku etapach nie będzie tak krytyczna. Na tym etapie można jedynie stwierdzić, że jak pokazane zostanie to dalej, na wybór funkcji pełniącej rolę funkcji diagonalnej w schemacie wierszowym i w całym *Full***3SAT** mamy wpływ.

Ponadto, jeśli po zakończeniu procedury sekwencyjnego weryfikowania całego *Full***3SAT**, wystąpi wyodrębnienie niezależnego **3SAT** = φ_3 , to jest to sygnał, że w problemie obliczeniowym możemy mieć do czynienia z ponad-wielomianową ilością rozwiązań. Możliwe jest jednak skuteczne prowadzenie rozstrzygania problemu decyzyjnego. Pokazuje to przypadek dyskutowany poniżej.

Zakładamy, że wektor rozwiązań zwraca ustalone wartościowania pierwszych *i-1* zmiennych. Pozostałe zmienne, w oparciu o formuły ***F*** każdego z *i-1* pierwszych schematów wierszowych, są w wektorze rozwiązań określone jako dowolnie wartościowane. Występuje tu opisana sytuacja wyodrębnienia niezależnego **3SAT**, którego wektor można konkatenować z wektorem pierwotnego problemu.

Analizę rozpoczynamy od Ω_i rozpatrując oddzielnie oba wartościowania zmiennej x_i – ***true*** oraz ***false***. Jeśli w obu przypadkach uzyskamy wynik weryfikacji określający dowolne wartościowanie zmiennych x_{i+1} do x_n, to przechodzimy do dalszej weryfikacji, czyli do analizy Ω_{i+1}, pozostawiając w wektorze rozwiązań oznaczenie dowolnego wartościowania x_i.

Możliwe jest jednak, by dla jednego wartościowania x_i pozostałe x_{i+1} do x_n można wartościować dowolnie, zaś dla drugiego wartościowania x_i – spośród x_{i+1} do x_n tylko część dopuszczała dowolne wartościowanie. W takiej sytuacji proces weryfikacji Ω_i powtarzamy z tymi ustalonymi wartościami zmiennych spośród x_{i+1} do x_n oraz tym ustalonym wartościowaniem x_i, dodatkowo zaznaczając takie wartościowanie zmiennej x_i pewnym znacznikiem. Procedurę kontynuujemy tak długo, aż osiągniemy Ω_{n-2} lub ponownie natkniemy się na wyodrębniony przypadek niezależności pewnych zmiennych i jeśli w opisanej sytuacji stwierdzony zostanie brak dopuszczalnego wartościowania, to należy wrócić do zmiennej oznaczonej znacznikiem, zmienić jej wartościowanie w wektorze rozwiązań na przeciwne usuwając znacznik i przejść do procedury weryfikacji od następnego Ω_{i+1}.

Rozstrzygnięcie problemu decyzyjnego wg tej procedury realizowane będzie w czasie wielomianowym.

Jeśli w każdym kolejnym etapie weryfikacji Ω_j ($j = i, i+1, \ldots, n-2$) dla obu wartościowań zmiennych x_j oddzielnie będziemy

stwierdzać dowolne wartościowanie zmiennych x_{j+1} do x_n (tak będzie np. gdy wszystkie Ω_j skonstruowane będą przez funkcje stałe f_{255} lub funkcje odwzorowania klauzul *Horna* tj. f_{247} , f_{223} , f_{191} oraz f_{127} - możemy tak przyjąć, bo rozpatrujemy model uogólniony *Full***3SAT**), to cała procedura wymagać będzie *(n-i)* krotnego wykonania po 2 razy weryfikacji schematów wierszowych, a te są wykonywane w czasie wielomianowym. W tej sytuacji w istocie mamy do czynienia z (*n-i)* krotnym konkatenowaniem.

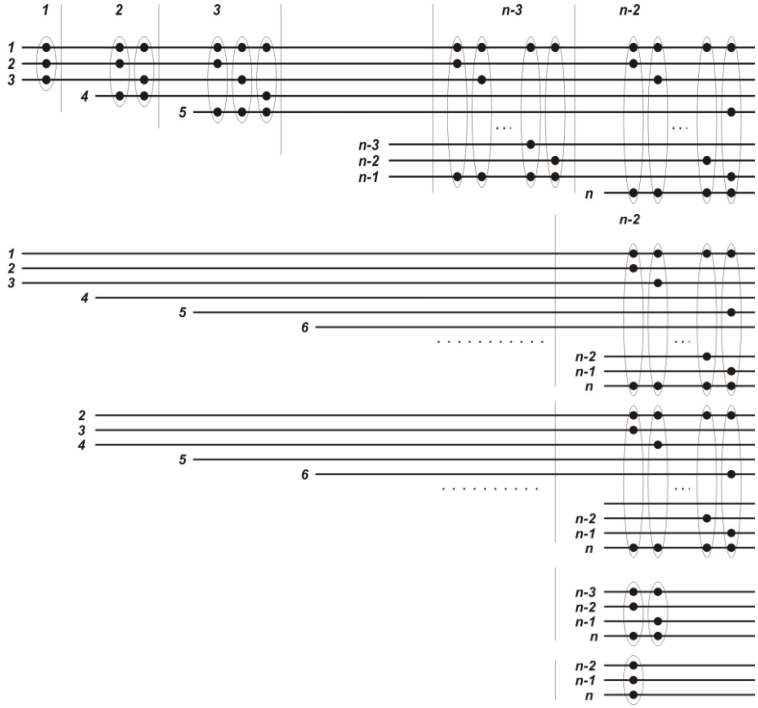
Jeśli na pewnym kolejnym etapie weryfikacji Ω_j *(j=i, i+1, …, n-2)* wystąpi stwierdzenie konieczności konkatenowania, to wszystkie poprzednie etapy są realizowane w czasie wielomianowym, a sama operacja (jak pokazano wyżej) konkatenacji wykona się także wielomianowo.

Jeśli dla ustalonej zmiennej ze znacznikiem x_z, stwierdzimy na kolejnym etapie brak rozwiązania (brak dopuszczalnego wartościowania przynajmniej jednej zmiennej spośród x_{i+1} do x_n), to również w tym przypadku przeprowadzenie całej procedury wymagać będzie czasu wielomianowego. Będzie tak, ponieważ procedura zakłada wtedy powrót do ponownej weryfikacji począwszy od tej x_z zmiennej z przeciwnym wartościowaniem. Więc jeśli stwierdzimy brak dopuszczalnego wartościowania ostatniej zmiennej przy znaczniku konkatenacji ustawionym dla *i-tej* zmiennej, to powtórzenie procesu weryfikacji dokona się także w czasie wielomianowym.

3.4 Analiza schematu kolumnowego w *3SAT*

Dyskusję tego schematu ograniczymy do stwierdzenia, że schemat kolumnowy jest transponowaną postacią schematu wierszowego. *j – ty* schemat kolumnowy *Full***3SAT** ma postać koniunkcji grup funkcji $F_{i,j}$ *(i=1, … , n-2)* $j \in \{1, \dots, n-2\}$ i charakteryzuje się wystąpieniem co najwyżej jednej funkcji trzech zmiennych o kolejnych numerach (będzie to *j-ta* funkcja diagonalna). Pozostałe funkcje w każdej z możliwych *i-tych* grup są funkcjami wszystkich takich kombinacji trzech spośród *n* zmiennych, w których trzecią zmienną jest zmienna x_n a pierwszą – zmienna x_i. Na rys. 3.7 przedstawiono graficznie wynik transponowania schematu wierszowego Ω_1 do postaci

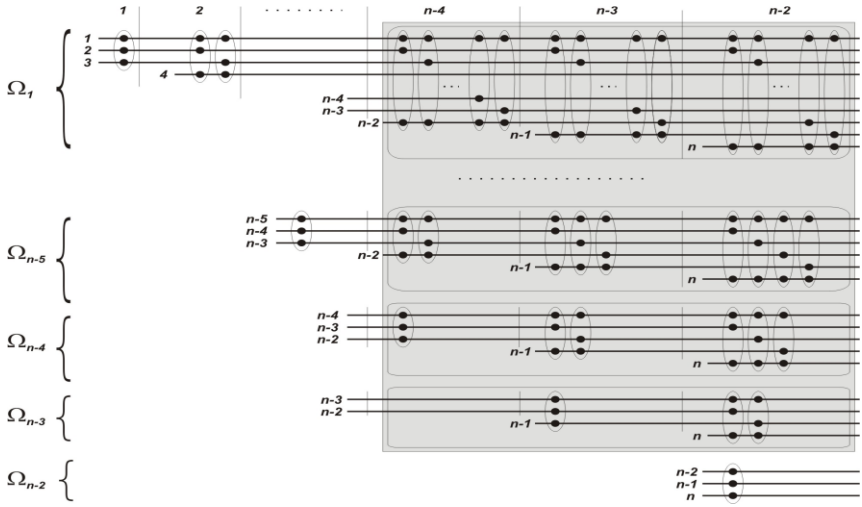
schematu kolumnowego. Operacja transponowania sprowadza się do przenumerowania wszystkich zmiennych tak by nastąpiła zamiana x_1 z x_n, x_2 z x_{n-1} itd. Ten sposób przenumerowania nazwiemy pełną inwersją numerów. W trakcie operacji pełnej inwersji numerów zmiennych schematu wierszowego, skrajne zmienne w każdej funkcji zamieniają się miejscami. Konsekwencją zamiany jest zmiana numeru funkcji (np. funkcja f_{65} po zamianie miejscami skrajnych zmiennych przekształca się w funkcję f_{95}).



Rys. 3.7. schemat wierszowy przekształcony w schemat kolumnowy.

Schemat kolumnowy, będąc transponowaną postacią schematu wierszowego, posiada wszystkie cechy tego ostatniego – jest **3SAT** i istnieje w stosunku do niego możliwość realizowania sekwencyjnego weryfikowania jego spełnialności. Zachowuje także możliwość stosowania analogicznych strategii weryfikowania jego spełnialności.

Przykład jednej z możliwych strategii rozwiązania *Full3SAT* wykorzystującego schemat kolumnowy jako schemat „startowy" ilustruje rys. 3.8.



Rys. 3.8. Ilustracja konstrukcji problemu *2SAT* przy ustalonym wartościowaniu zmiennych funkcji diagonalnej $f(x_{n-2}, x_{n-1}, x_n)$ w schemacie kolumnowym.

Jeśli funkcja diagonalna $f(x_{n-2}, x_{n-1}, x_n)$ *Full3SAT* jest rzędu $k=1$ (jedna elementarna koniunkcja), tzn. jest wartościowana przez ustaloną kombinację wartości swoich zmiennych, to w pełnym schemacie *Full3SAT* trzy określone podschematy kolumnowe przekształcają się w problem **2SAT** wszystkich n zmiennych. Na rys. 3.8 pokazano funkcje podschematów $\boldsymbol{\Omega}_i$, które mogą być rozpatrywane jako **2SAT** przy założeniu ustalonego wartościowania zmiennych funkcji $f(x_{n-2}, x_{n-1}, x_n)$. Wyodrębniony w ten sposób problem **2SAT** obejmuje wszystkie funkcje pogrupowane w schematach kolumnowych $\boldsymbol{\Omega}_i$ (albo inaczej – obejmuje grupy funkcji $\boldsymbol{F}_{i,n-4}$, $\boldsymbol{F}_{i,n-3}$ oraz $\boldsymbol{F}_{i,n-2}$ wszystkich schematów wierszowych od $\boldsymbol{\Omega}_i$ do $\boldsymbol{\Omega}_{n-3}$.

Jeśli problem **2SAT** skonstruowany wg schematu na rys. 3.8 daje wektor rozwiązań o ustalonym wartościowaniu zmiennych $x_1, \ldots x_{n-3}$, to wystarczy uzyskane rozwiązanie sprawdzić dla pozostałych funkcji *Full3SAT*.

Jeśli jednak wektor rozwiązań problemu **2SAT** dopuszcza dowolne wartościowanie zmiennych x_1, … x_{n-3} , tj ma postać **[*** … *x_{n-2} x_{n-1} x_n]**, to dla znalezienia rozwiązania możemy dalej ograniczyć się rozpatrywaniem problemu **3SAT** zmiennych x_1, … x_{n-3} (weryfikacja w oparciu o pozostałe funkcje nie wchodzące do **2SAT** nie ma racji bytu ze względu na konieczność wyczerpującego wartościowania 2^{n-3}). Inaczej, jeśli wektor rozwiązań problemu **2SAT** dla ustalonych wartościowań zmiennych x_{n-2} , x_{n-1} , x_n ma postać **[*** … *x_{n-2} x_{n-1} x_n]**, to oznacza, że wszystkie funkcje wyodrębnionego problemu **2SAT** można w dalszych rozważaniach pominąć, bo generowany obszar rozwiązań dopuszczalnych pokrywa się z obszarem rozwiązań problemu **3SAT** *(n-3)* pierwszych zmiennych. Pozwala to konkatenować wektor rozwiązań problemu **3SAT** *(n-3)* pierwszych zmiennych z ustalonymi wartościowaniami zmiennych x_{n-2} , x_{n-1} oraz x_n.

Jeśli rozpatrywana funkcja diagonalna jest rzędu $k>1$, to postępowanie sprowadza się do rozpatrzenia k problemów **2SAT**.

3.5 Analiza schematu diagonalnego w *3SAT*

W każdej z wyodrębnionych podformuł Ω_i modelu referencyjnego wybieramy funkcje diagonalne $d_i = f(x_i , x_{i+1} , x_{i+2})$ a pozostałe funkcje podformuł Ω_i oznaczamy Ω_i^*. Jeśli koniunkcję wszystkich funkcji diagonalnych d_i modelu referencyjnego *Full3SAT* oznaczymy *Diag3SAT*, zaś koniunkcję wszystkich podformuł Ω_i^* oznaczymy *Comp3SAT*, to możemy *Full***3SAT** zapisać jako koniunkcję dwóch podproblemów, z których każdy pozostaje **3SAT**, czyli *Full3SAT = Diag3SAT* ∧ *Comp3SAT:*

$$\boldsymbol{Full3SAT} = d_1 \wedge d_2 \wedge \ldots \wedge d_{n-2} \wedge \Omega_1^* \wedge \Omega_2^* \wedge \Omega_3^* \wedge \ldots \wedge \Omega_{n-3}^* \wedge \Omega_{n-2}^*$$

Otrzymana postać pozwala wnioskować, że jeśli formuła *Diag3SAT* nie jest spełniana, to pełna formuła *Full3SAT* nie jest spełniana. Można przyjąć, że spełnialność *Diag3SAT* jest warunkiem koniecznym spełnialności *Full3SAT* i badanie spełnialności rozpocząć od *Diag3SAT*.

Uzasadnieniem takiego wyboru jest to, że w modelu referencyjnym *Full3SAT* liczebność n_d funkcji stanowiących

podformułę *Diag3SAT* jest równa n-2, zaś reszta spośród n_c=M-n+2 funkcji modelu *Full3SAT* tworzy podformułę *Comp3SAT* więc zachodzi relacja $n_d << n_c$. Są więc przesłanki – mała liczba funkcji, mała rozpiętość δ_1 i δ_2 wskazująca na „krótkie" powiązania zmiennych między sobą, a wręcz niezależność części od innych - by sądzić, że problem w tej postaci można stosunkowo łatwo rozwiązać.

Trzeba jednak mieć na uwadze fakt, że *Diag3SAT* musi być rozwiązany jako problem obliczeniowy (musi zwrócić wektor wartościowania wszystkich zmiennych), po to by móc wykorzystać je do sprawdzenia formuły *Comp3SAT*. W przeciwnym razie, nawet gdyby *Comp3SAT* był rozwiązywalny, choćby w wersji decyzyjnej, to jego rozwiązanie mogłoby być rozłączne względem *Diag3SAT* i nie można wnioskować o spełnialności wyjściowego *Full3SAT*.

Jest jeszcze jeden powód, dla którego problem diagonalny warto dokładnie przeanalizować. Szereg wyodrębnionych do tej pory problemów **3SAT** charakteryzuje się wprost strukturą diagonalną, bądź strukturą bardzo podobną do diagonalnej. Przykładem może być zdefiniowany w książce Papadimitriou[1] następujący problem (w postaci *CNF* klauzul): *„Problem 3SAT pozostaje NP-zupełny dla wyrażeń, w których każda zmienna może się pojawić najwyżej trzy razy, a każdy literał – najwyżej dwa razy"*.

Otóż w oparciu o powyższe określenie można skonstruować w pełni diagonalny problem **3SAT**, w którym wystąpią wyłącznie klauzule *Horna* (wtedy problem ma rozwiązanie wielomianowe). Możliwe jest także skonstruowanie problemu diagonalnego, w którym występować mogą tylko klauzule *coHorna*, bądź jednocześnie oba rodzaje klauzul – *Horna* i *coHorna*. Prosty przykład takiego problemu może mieć postać (zapis *CNF* klauzul):

$$\boldsymbol{3SAT} = (\neg x_1 \vee \neg x_2 \vee x_3)(x_2 \vee x_3 \vee \neg x_4)(\neg x_3 \vee x_4 \vee \neg x_5)(\neg x_4 \vee x_5 \vee x_6)$$

Mamy tu dwie klauzule *Horna* – pierwsza i trzecia oraz dwie klauzule *coHorna* – druga i czwarta. Wymogi dotyczące ilości zmiennych i ich literałów są spełnione.

---

[1] Christos H. Papadimitriou, "**Computational Complexity" –**first edition 1994. (*tłum. polskie – „**Złożoność obliczeniowa", wydanie drugie WNT 2007**) – **uwaga 9.3***

zmienna	x_1	x_2	x_3	x_4	x_5	x_6
ilość zmiennych	1	2	3	3	2	1
ilość literałów	1	2	2	2	2	1

Ponieważ, po pierwsze – z analizy schematu wierszowego wiemy, że dowolny *Full3SAT* (w tym także w postaci „zdegenerowanej" jakim jest schemat diagonalny) ma w wersji decyzyjnej rozwiązanie wielomianowe, a po drugie – stwierdziliśmy, że dla rozwiązania *Full3SAT* przedstawionego w postaci *Diag3SAT* ∧ *Comp3SAT*, konieczne jest rozwiązanie problemu w aspekcie obliczeniowym, analizę *Diag3SAT* popro-wadzimy w tym kierunku.

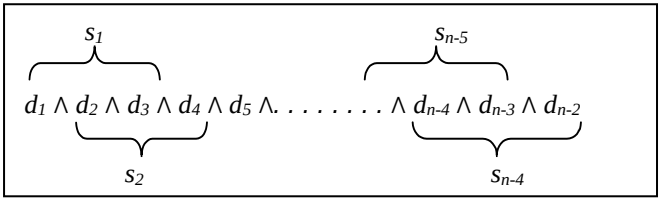
Zaczniemy od spostrzeżenia, że dla pary sąsiednich funkcji diagonalnych, aby ich koniunkcja była spełniana musi zachodzić jednocześnie identyczne wartościowanie drugiej i trzeciej zmiennej z pierwszej funkcji oraz pierwszej i drugiej zmiennej z drugiej funkcji rozpatrywanej pary odpowiednio.

W efekcie weryfikacja spełnialności *Diag3SAT* sprowadza się do sprawdzenia spełnialności wszystkich koniunkcji postaci $s_i = d_i \wedge d_{i+1} \wedge d_{i+2}$ dla $i = 1, 2, \ldots, n\text{-}4$ (d_{i+3} i kolejne są niezależne od d_i). Sprawdzenie wszystkich koniunkcji s_i realizowane jest w czasie wielomianowym i może być wykonywane niezależnie.

Drugi krok polega na sprawdzeniu czy s_i oraz s_{i+1} posiadają wspólne wartościowanie, takie, że s_{i+1} gwarantuje wartościowanie spełniające dla d_{i+3}. Ten krok wykonywany jest także w czasie wielomianowym i sprawdzenie każdej pary (s_i, s_{i+1}) może być wykonane niezależnie.

Jeśli każda para koniunkcji (s_i, s_{i+1}) posiada spełniające wartościowanie wspólnych zmiennych gwarantujący spełnialność d_{i+3}, to *Diag3SAT* jest spełnialny.

Związki funkcji diagonalnych d_i i koniunkcji s_i w opisanej procedurze weryfikowania *Diag3SAT* ilustruje rys. 3.9



Rys. 3.9 Związki funkcji diagonalnych

Aby efektywnie prowadzić proces sprawdzania warunków gwarantujących przy spełnialności pary koniunkcji (s_i, s_{i+1}) spełnialność funkcji d_{i+3} skorzystamy z transformacji (nazywamy ją „łuskową") polegającej na zastąpieniu w słowie binarnym każdych kolejnych ciągów trzech symboli ich reprezentacją dziesiętną. Słowa binarne interpretujemy jako możliwe wartościowania zmiennych. Reprezentacja dziesiętna ciągu trzech binarnych symboli jest interpretowana jako wartościowanie odpowiednich elementarnych koniunkcji w opisie funkcji trzech zmiennych. Efekt wprowadzonej transformacji dla słów przestrzeni rozwiązań pięciu zmiennych przedstawia rys. 3.10.

Przestrzeń rozwiązań w reprezentacji binarnej

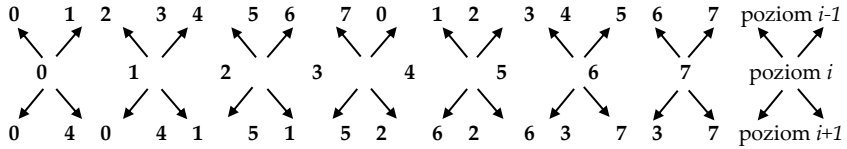
x_1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
x_2	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
x_3	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
x_4	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
x_5	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Przestrzeń rozwiązań po transformacji "łuskowej"

$d_1(x_1, x_2, x_3)$	0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
$d_2(x_2, x_3, x_4)$	0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7
$d_3(x_3, x_4, x_5)$	0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6 7 7 7 7

Rys. 3.10 Przykład reprezentacji obszaru rozwiązań po transformacji „łuskowej"

Analiza postaci przestrzeni rozwiązań traktowanej jako wynik transformacji „łuskowej", pozwala stwierdzić, że dla każdego z możliwych symboli w odpowiednich słowach, istnieją tylko dwa symbole – następniki i odwrotnie, każdy z 8 symboli ma tylko 2 symbole – poprzedniki. Związki te można przedstawić w formie diagramu (rys. 3.11).



Rys. 3.11 Diagram związków „następnik-poprzednik".

Opisana wyżej koncepcja rozwiązania problemu *Diag3SAT* może być wykorzystana do konstrukcji algorytmu dla problemu decyzyjnego, a po jego rozszerzeniu także dla problemu obliczeniowego.

Całość postępowania wymaga, aby wykorzystując wprowadzoną transformację, dla każdej funkcji diagonalnej d_i ($i = 1, 2, \ldots, n-2$) utworzyć zbiór $\boldsymbol{K_i}$ odwzorowań jej elementarnych koniunkcji.

Dysponując zbiorami odwzorowań $\boldsymbol{K_i}$, dla każdych kolejnych trzech zbiorów $\boldsymbol{K_i}$, $\boldsymbol{K_{i+1}}$ oraz $\boldsymbol{K_{i+2}}$ należy ustalić w oparciu o diagram związków „poprzednik-następnik" zbior $\boldsymbol{S_i}$ ($i=1, 2, \ldots, n-4$) słów charakteryzujących spełnialność koniunkcji kolejnych trzech funkcji diagonalnych d_i, d_{i+1} oraz d_{i+2}. Liczebność takich słów w każdym ze zbiorów $\boldsymbol{S_i}$ nie będzie większa niż 32 (będzie równa 32 w przypadku koniunkcji diagonalnych funkcji f_{255} ***TRUE*** – zwracamy na to uwagę, bo taką możliwość dopuszcza model referencyjny). Jeśli którykolwiek ze zbiorów $\boldsymbol{S_i}$ będzie pusty, tj. nie istnieje żadne słowo charakteryzujące spełnialność odpowiadającej $\boldsymbol{S_i}$ koniunkcji trzech funkcji diagonalnych, to cały analizowany problem *Diag3SAT* nie jest spełnialny.

W kolejnym kroku, dla par zbiorów $\boldsymbol{S_i}$ oraz $\boldsymbol{S_{i+1}}$ ($i=1, 2, \ldots, n-4$) ustalamy słowa mające wspólne elementy – drugi element słowa ze zbioru $\boldsymbol{S_i}$ z pierwszym elementem słowa ze zbioru $\boldsymbol{S_{i+1}}$ oraz trzeci element słowa ze zbioru $\boldsymbol{S_i}$ z drugim elementem w słowie ze zbioru $\boldsymbol{S_{i+1}}$. Zbiór takich słów uzyskamy w wyniku skończonej (maksymalnie 32 razy 32) ilości porównań słów każde z każdym w rozpatrywanych zbiorach. Z kolei liczebność ustalonych tak wspólnych słów nie będzie większa niż 16, bo każdy pierwszy (może ich być maksymalnie 8) element tak ustalanego wspólnego słowa może mieć tylko dwa następniki).

Jeśli nie zostaną znalezione takie wspólne słowa, oznacza to że analizowany problem nie jest spełnialny.

Problem obliczeniowy sprowadzać się będzie do skonstruowania drzewa (drzew) rozwiązań w oparciu o zweryfikowane zbiory $\boldsymbol{S_i}$ słów mających wspólne elementy ze słowami zbioru $\boldsymbol{S_{i+1}}$. Utworzone drzewo (drzewa) umożliwiają odczytywanie wektora rozwiązań - słów spełniających wyrażenie badanego problemu *Diag3SAT* (oczywistym jest konieczność wykonania transformacji odwrotnej do „łuskowej").

W ogólnym przypadku możemy mieć do czynienia z ponadwielomianową ilością słów spełniających badanego *Diag3SAT*.

Dla zilustrowania opisanej procedury w odniesieniu do decyzyjnego oraz obliczeniowego problemu *Diag3SAT*, posłużymy się dwoma przykładami. Problem zdefiniowany w pierwszym przykładzie jest problemem spełnialnym, drugi nie jest spełnialny. W obu przykładach problemy zdefiniowane są z użyciem zapisu *DNF*. Używamy tej postaci, ponieważ ułatwia przejście do transformacji „łuskowej".

Przykład 3.3 Problem diagonalny *Diag3SAT* = $d_1 \wedge d_2 \wedge d_3 \wedge d_4$ definiują funkcje diagonalne $d_1=f_{197}(x_1, x_2, x_3)$, $d_2=f_{216}(x_2, x_3, x_4)$, $d_3=f_{234}(x_3, x_4, x_5)$ oraz $d_4=f_{89}(x_4, x_5, x_6)$.

$f_{197}(x_1, x_2, x_3) = \overline{x}_1\,\overline{x}_2\,\overline{x}_3 \vee x_1\,\overline{x}_2\,x_3 \vee \overline{x}_1\,x_2\,x_3 \vee x_1\,x_2\,x_3$
$f_{216}(x_2, x_3, x_4) = x_2\,x_3\,\overline{x}_4 \vee \overline{x}_2\,\overline{x}_3\,x_4 \vee \overline{x}_2\,x_3\,x_4 \vee x_2\,x_3\,x_4$
$f_{234}(x_3, x_4, x_5) = x_3\,\overline{x}_4\,\overline{x}_5 \vee x_3\,x_4\,\overline{x}_5 \vee x_3\,\overline{x}_4\,x_5 \vee \overline{x}_3\,x_4\,x_5 \vee x_3\,x_4\,x_5$
$f_{89}(x_4, x_5, x_6) = \overline{x}_4\,\overline{x}_5\,\overline{x}_6 \vee x_4\,x_5\,\overline{x}_6 \vee \overline{x}_4\,\overline{x}_5\,x_6 \vee \overline{x}_4\,x_5\,x_6$

W pierwszym kroku ustalamy zbiory $\boldsymbol{K_{1\text{-}4}}$ odwzorowań elementarnych koniunkcji funkcji diagonalnych. Wynik prezentowany jest w drugiej kolumnie zestawienia tabelarycznego.

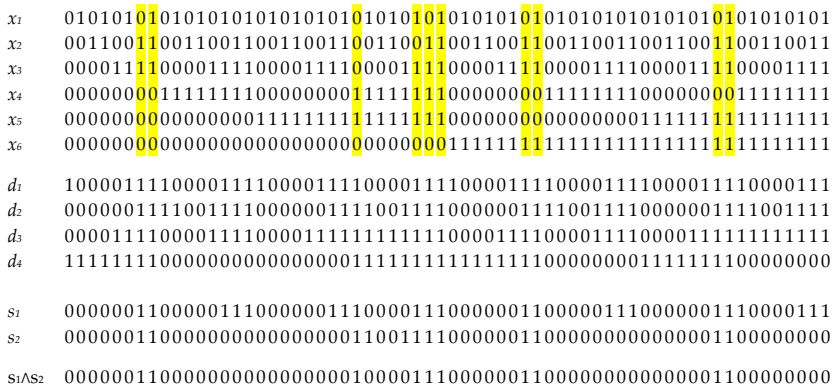
Dalej, ustalamy zbiory $\boldsymbol{S_1}$ i $\boldsymbol{S_2}$ słów charakteryzujących spełnialność koniunkcji funkcji diagonalnych $d_1 \wedge d_2 \wedge d_3$ oraz $d_2 \wedge d_3 \wedge d_4$ odpowiednio. Zbiór $\boldsymbol{S_1}$ stanowi 11 słów $s_{1\text{-}11}$, których elementy pokazane są w układzie kolumnowym w trzeciej kolumnie zestawienia. W kolumnie czwartej zestawienia pokazano 6 słów zbioru $\boldsymbol{S_2}$.

funkcje diagonalne	zbiory $\boldsymbol{K_i}$ odwzorowania elementarnych koniunkcji	zbiór $\boldsymbol{S_1}$	zbiór $\boldsymbol{S_2}$	wspólne elementy słów zbiorów $\boldsymbol{S_1}$ i $\boldsymbol{S_2}$
1	**2**	**3**	**4**	**5**
		słowa $s_{1\text{-}11}$ zbioru $\boldsymbol{S_1}$	słowa $s_{1\text{-}8}$ zbioru $\boldsymbol{S_2}$	
		S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11	S1 S2 S3 S4 S5 S6 S7 S8	
		słowa $s_{1\text{-}11}$ zbioru $\boldsymbol{S_1}$	słowa $s_{1\text{-}8}$ zbioru $\boldsymbol{S_2}$	
		S1 S2 S3 S4 S5 S6 S7S8 S9 S10 S11	S1 S2 S3 S4 S5 S6	
d_1	{ 0, 5, 6, 7 }	**0 5 5 6 6 6 6 7 7 7 7**		
d_2	{ 3, 4, 6, 7 }	**4 6 6 3 3 7 7 3 3 7 7**	3 3 3 4 6 7	4 6 3 3 7
d_3	{ 1, 3, 5, 6, 7 }	**6 3 7 1 5 3 7 1 5 3 7**	1 1 5 6 7 7	6 7 1 5 7
d_4	{ 0, 3, 4, 6 }		0 4 6 3 3 7	

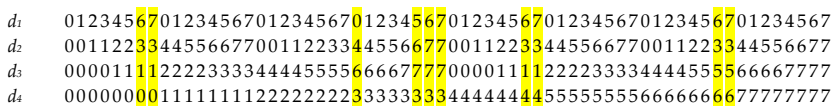
Ostatecznie ustalone zostały elementy wspólne słów ze zbiorów S_1 i S_2. Zbiór tych wspólnych elementów stanowią pary (4, 6), (6, 7), (3, 1), (3, 5) oraz (7, 7). Ponieważ zbiór nie jest pusty, rozstrzygamy spełnialność naszego problemu *Diag3SAT*.

W wersji obliczeniowej, w oparciu o pary elementów wspólnych konstruujemy wektory rozwiązań. Dla przykładu, dla pary wspólnych elementów (3, 1), w zbiorze S_1 odnajdujemy dwa słowa s_4 i s_8, a w zbiorze S_2 słowa s_1 i s_2, to w efekcie uzyskujemy cztery wektory rozwiązań [6, 3, 1, 0], [6, 3, 1, 4], [7, 3, 1, 0] oraz [7, 3, 1, 4]. Postępując analogicznie dla pozostałych par, uzyskujemy kolejne wektory rozwiązań: [0, 4, 6, 3], [5, 6, 7, 3], [6, 3, 5, 6], [6, 7, 7, 3], [7, 3, 5, 6] oraz [7, 7, 7, 3].

Przestrzeń rozwiązań w reprezentacji binarnej



Przestrzeń rozwiązań w reprezentacji „łuskowej”



Przykład 3.4 Problem diagonalny *Diag3SAT* = $d_1 \wedge d_2 \wedge d_3 \wedge d_4$ definiują funkcje diagonalne $d_1=f_{133}(x_1, x_2, x_3)$, $d_2=f_{67}(x_2, x_3, x_4)$, $d_3=f_{240}(x_3, x_4, x_5)$ oraz $d_4=f_{169}(x_4, x_5, x_6)$.

$$f_{133}(x_1, x_2, x_3) = \overline{x}_1\,\overline{x}_2\,\overline{x}_3 \vee \overline{x}_1\,x_2\,\overline{x}_3 \vee x_1\,x_2\,x_3$$

$$f_{67}(x_2, x_3, x_4) = \overline{x}_2\,\overline{x}_3\,\overline{x}_4 \vee x_2\,\overline{x}_3\,\overline{x}_4 \vee \overline{x}_2\,x_3\,x_4$$

$$f_{240}(x_3, x_4, x_5) = \overline{x}_3\,\overline{x}_4\,x_5 \vee x_3\,\overline{x}_4\,x_5 \vee \overline{x}_3\,x_4\,x_5 \vee x_3\,x_4\,x_5$$

$$f_{169}(x_4, x_5, x_6) = x_4\,\overline{x}_5\,\overline{x}_6 \vee x_4\,x_5\,\overline{x}_6 \vee x_4\,\overline{x}_5\,x_6 \vee x_4\,x_5\,x_6$$

Podobnie jak w poprzednim przykładzie, poszczególne etapy będą prezentowane w zestawieniu tabelarycznym w niezmienionym układzie kolumn. Ustalamy więc najpierw zbiory $K_{1\text{-}4}$ odwzorowań elementarnych koniunkcji funkcji diagonalnych. Następnie ustalamy zbiory S_1 i S_2 słów charakteryzujących spełnialność koniunkcji funkcji diagonalnych $d_1 \ldots d_4$. Analizując zbiory S_1 i S_2 w poszukiwaniu słów ze wspólnymi elementami, stwierdzamy, że takich słów nie ma. Na tej podstawie rozstrzygamy brak spełnialności zdefiniowanego w przykładzie problemu *Diag3SAT*.

funkcje diagonalne	zbiory K_i odwzorowania elementarnych koniunkcji	zbiór S_1	zbiór S_2	wspólne elementy słów zbiorów S_1 i S_2
1	**2**	**3**	**4**	**5**
		słowa zbioru S_1	słowa zbioru S_2	
		słowa $s_{1\text{-}2}$ zbioru S_1	słowa $s_{1\text{-}2}$ zbioru S_2	
		s_1 s_2	s_1 s_2	
d_1	{ 0, 2, 7 }	**0 2**		
d_2	{ 0, 1, 6 }	**0 1**		-
d_3	{ 4, 5, 6, 7 }	**4 4**	**6 6**	-
d_4	{ 1, 3, 5, 7 }		**7 7**	
			3 7	

Przestrzeń rozwiązań w reprezentacji binarnej

x_1 0101010101010101010101010101010101010101010101010101010101010101
x_2 0011001100110011001100110011001100110011001100110011001100110011
x_3 0000111100001111000011110000111100001111000011110000111100001111
x_4 0000000011111111000000001111111100000000111111110000000011111111
x_5 0000000000000000111111111111111100000000000000001111111111111111
x_6 0000000000000000000000000000000011111111111111111111111111111111

d_1 1010000110100001101000011010000110100001101000011010000110100001
d_2 1111000000001100111100000000110011110000000011001111000000001100
d_3 0000000000000000111111111111111100000000000000001111111111111111
d_4 0000000011111111000000001111111100000000111111110000000011111111

s_1 0000000000000000101000000000000000000000000000001010000000000000
s_2 0000000000000000000000000000001100000000000000000000000000001100

$s_1 \wedge s_2$ 0000000000000000000000000000000000000000000000000000000000000000

Przestrzeń rozwiązań w reprezentacji „łuskowej”

d_1 0123456701234567012345670123456701234567012345670123456701234567
d_2 0011223344556677001122334455667700112233445566770011223344556677
d_3 0000111122223333444455556666777700001111222233334444555566667777
d_4 0000000011111111222222223333333344444444555555556666666677777777

3.6 Podsumowanie

W przypadku problemu spełnialności **SAT**, analiza jego wariantów, w szczególności **2-** i **3SAT** okazuje się prostszą dzięki wykorzystaniu modeli referencyjnych.

Także skorzystanie w prowadzonej dyskusji wzajemnej ekwiwalentności rachunku zdań i funkcji logicznych ułatwiło wykazanie i przeanalizowanie związków między funkcjami logicznymi 2 i 3 zmiennych, które definiują **2-** i **SAT**. Efektem przeprowadzonej analizy tych związków jest stwierdzenie, że wśród 255 funkcji 3 zmiennych dla części z nich istnieją odwzorowania funkcji 3 zmiennych przez funkcje 2 zmiennych. Wyjawnione odwzorowania funkcji mogą być podstawą dwóch typów przekształceń w odniesieniu do struktur **2SAT**. W szczególności przekształcenie typu P_V umożliwia przekształcenie struktury kolumnowej **2SAT** do części struktury wierszowej **3SAT**. Jeśli przekształcenie to zastosujemy do wszystkich struktur kolumnowych **2SAT**, to wynikiem będzie pełna elementarna struktura wierszowa **3SAT**.

Mniej interesujące może być przekształcenie typ P_H, bowiem efektem jego zastosowani do struktury wierszowej **2SAT** jest wierszowa struktura **3SAT**. Warto jednak mieć je na uwadze, ponieważ może być przydatne w pewnych przypadkach przy przekształceniach odwrotnych struktur **3SAT** do **2SAT**.

Poza dyskusją poświęconą przekształceniom wzajemnym **2SAT** i **3SAT**, dyskutowane są konstrukcje algorytmów dla wydrębnionych struktur problemu **3SAT** (**3SAT** interesuje nas przede wszystkim dlatego, że ogólnie jest klasyfikowany jako problem *NP-zupełny*). W szczególności dyskusja pokazuje, że dla problemu **3SAT** charaktryzującego się strukturą wierszową Ω_1 możliwe jest skonstruowanie algorytmu wielomianowego. Podobnie jest dla problemu **3SAT** o strukturze diagonalnej.

Wskazanie, że istnieje algorytm wielomianowy dla **3SAT** o strukturze wierszowej Ω_1, nie rozstrzyga, że problem *PvNP* jest rozwiązany i ma postać równości klas *P* i *NP*.

Jednak warto zauważyć, że przy pomocy przekształcenia typu P_V do postaci o strukturze wierszowej Ω_1 można sprowadzić dowolny problem **2SAT**. Może to być jego uogólnienie takie jak na przykład **MAX2SAT** (problem polegający na stwierdzeniu czy istnieje wartościowanie spełniające przynajmniej ustaloną ilość klauzul) [1 s. 202].

Zasygnalizowana kwestia nie jest tutaj dyskutowana – sugeruje jednak, tą drogą możliwe jest ustalenie relacji pomiędzy klasami *P* i *NP*.

BIBLIOGRAFIA

[1] Papadimitriou Christos, Złożoność obliczeniowa, Warszawa, WNT, 2007

[2] Sipser Michael, Wprowadzenie do teorii obliczeń, Warszawa, WNT, 2009

[3] Traczyk Wiesław, Układy cyfrowe. Podstawy teoretyczne i metody syntezy, Warszawa, WNT, 1986

R o z d z i a ł c z w a r t y

Dyskusja relacji inkluzji $NC \subseteq P \subseteq NP$

Marek Malinowski

Technologie Teleinformatyczne

W dyskutowanej relacji inkluzji $NC \subseteq P \subseteq NP$, *NC* oznacza klasę związaną z pojęciem równoległości obliczeń i rozpatrywaną w modelu sieci logicznych.
Fakt, że fizyczną formą sieci logicznych są układy kombinacyjne, to do ich formalnego opisu można stosować różne postaci algebry Boole'a: algebry zdań, algebry zbiorów, algebry sygnałów i algebry wektorów. Na przykładach problemu 2SAT i 3SAT pokazano, że w obu przypadkach po ustaleniu wektorów początkowych możliwe jest ich składanie w strukturze drzewa binarnego na podobieństwo struktury multiplikatora równoległego.
Prowadzenie procesu składania wektorów rozwiązań częściowych problemu 2SAT rozstrzyga jego przynależność do klasy NC. Ponieważ w podobny sposób prowadzone jest składanie wektorów 3SAT, to rozstrzyga jego przynależność także do klasy NC.
W efekcie relacja inkluzji $NC \subseteq P \subseteq NP$ sprowadza się do równości klas *NC, P i NP*.

4

Uściślaj stopniowo

- od szczegółu do ogółu

[z teorii rozwiązywania problemów]

4.1 Równoległość w algorytmice

Znaczenie równoległości w algorytmice i teorii złożoności przejawia się zarówno w aspekcie praktycznym, jak i teoretycznym. Równoległość, w ujęciu praktycznym wykorzystawana jest do usprawnienia obliczeń konkretnych problemów, natomiast w ujęciu teoretycznym służy do scharakteryzowania relacji między całymi grupami problemów.

Tytułowy ciąg relacji inkluzji ***NC*** ⊆ ***P*** ⊆ ***NP*** definiuje, oprócz kluczowej dla współczesnej informatyki kwestii *P versus NP*, drugą nie mniej ważną kwestię *NC versus P*. Kwestia ta, podobnie jak *P versus NP*, sprowadza się do określenia charakteru relacji między klasami ***NC*** i ***P***.

Klasa ***NC*** definiowana jest formalnie w różny sposób, zależnie od rozpatrywanego modelu obliczeń. Jedna z definicji odnosi się do sieci (obwodów) logicznych [5 s. 458], inna odnosi się do modelu maszyn równoległych o dostępie swobodnym *PRAM* [4 s. 393].

Niezależnie od modelu, klasa ***NC*** może być określona jako problemy mające szybkie, nie gorsze niż *czas polilogarytmiczny* $O(\log^k n)$ rozwiązania równoległe przy tylko wielomianowej liczbie procesorów. Tak więc, z klasą ***NC*** wiąże się pojęcie równoległości, rozumianej jako wykorzystywanie wielu rozłącznych, działających jednocześnie i współpracujących ze sobą elementów przetwarzających (procesorów).

Taki sposób prowadzenia obliczeń mogą realizować komputery wieloprocesorowe ze współdzieloną pamięcią (odwzorowa-

nie modelu *PRAM*). Mogą to być także sieci o stałych połączeniach, w których role elementów przetwarzających spełniają zwykle procesory o ograniczonych możliwościach obliczeniowych (odwzorowanie modelu sieci logicznych).

W obu modelach, za miarę złożoności obliczeniowej algorytmu równoległego przyjmuje się złożoność iloczynową ***rozmiar*** x ***czas***.

W modelu sieci logicznych, w którym najprostszą realistyczną realizacją są układy kombinacyjne budowane z bramek logicznych, ***rozmiar*** określa po prostu ilość bramek w układzie. Z kolei drugi człon, określający równoległą złożoność czasową, wyraża głębokość układu, tj. odległość między bramkami wejściową i wyjściową.

W odróżnieniu od modelu maszyny *PRAM*, bardzo istotnym dla modelu sieci logicznych, jest to, że konkretna sieć rozwiązuje problem o ustalonej liczbie zmiennych wejściowych. Natomiast przy pomocy maszyny *PRAM* można rozwiązywać problemy o dowolnej ilości zmiennych.

W tej sytuacji, aby mówić o rozwiązywaniu problemu o dowolnej ilości zmiennych przy pomocy sieci logicznych, musimy dysponować rodziną sieci, po jednej sieci dla danej ilości zmiennych definiujących problem. Dodatkowo wymagane jest, by każdą z tych sieci można było skonstruować w pamięci logarytmicznej. Ten wymóg określany jest mianem jednostajności sieci i musi być uwzględniany przy próbach konstruowania algorytmów równoległych.

Nie jest to jednak jedyne ograniczenie w aspekcie teoretykcznych rozważań dotyczących relacji między klasami problemów. Koniecznym jest uwzględnianie tego co wiąże się z ideą zupełności.

Dyskutowany ciąg relacji ***NC⊆ P⊆ NP*** jest jednym z wielu określonych w teorii złożoności uszeregowań klas problemów według skali trudności. Podstawą tego i innych uszeregowań jest redukcja, a dla rozstrzygnięcia charakteru relacji między elementami tych uszeregowań jest wykorzystywane pojęcie zupełności.

Redukcja więc, pozwala w obrębie zdefiniowanej klasy ustalić *„równoważność"* dwóch lub więcej problemów w takim sensie, że każdy z nich nie jest gorszy obliczeniowo (czasowo czy

pamięciowo) od pozostałych. Wszystkie tak „*równoważne*" w obrębie klasy problemy określane są jako problemy zupełne w danej klasie. Dodatkowo, przyjmuje się, że problemy zupełne są w szczególny sposób reprezentatywne w danej klasie i jest najmniej prawdopodobne, by mogły należeć do *słabszej* klasy w rozpatrywanym uszeregowaniu.

Rozumiana w ten sposób idea zupełności wzbogaciła metodologię teorii złożoności o istotny element. Złożoność jednego problemu zupełnego w danej klasie związana została ze złożonością całej klasy. Aby stwierdzić równość dwóch różnych klas, wystarczy bowiem pokazać, że problem zupełny w „*silniejszej*" klasie posiada cechy problemów „*słabszej*" klasy.

4.2 Koncepcja algorytmu równoległego

Wobec dosyć powszechnego przekonania o tym, że w ciągu relacji $NC \subseteq P \subseteq NP$ obie są ostre, zwykle rozpatrywane są one jako dwie odrębne, zasygnalizowane na początku kwestie. Jednak rezultaty przedstawione w poprzednich dyskusjach dotyczących:

- jednostajnych wielomianowych sieci logicznych problemów kSAT i rozstrzygnięcie $P=NP$ (rozdział 1);
- strukturalnego modelu referencyjnego kSAT, pozwalającego realizować na jego elementach szereg operacji w czasie logarytmicznym (rozdział 2),
- czy wreszcie wyniki analizy elementarnych struktur wierszowej, kolumnowej i diagonalnej oraz możliwe przekształcenia w modelu referencyjnym kSAT (rozdział 3), wskazujące na możliwość wykorzystania równoległości,

można potraktować jako przesłanki i zachętę do podjęcia prób rozstrzygnięcia relacji równości klas $NC=P$, czy wprost równości $NC=NP$.

Samo istnienie przesłanek to nie to samo co pokazanie konkretnego rozstrzygnięcia – tu pokazania równoległego algorytmu rozwiązania problemów **2SAT** i **3SAT**. Zwykle przy wymyślaniu sposobu rozwiązania jakiegoś problemu sugerowane jest sięgnięcie w pierwszej kolejności po kilka ogólnych metod. David Harel w swojej książce [3 s. 94] pisze *„ … projektant algorytmów może odnieść korzyści wprowadzając właśnie te wzorce w pierwszej kolejności. Jednak z całą pewnością projektowanie*

algorytmów jest zajęciem twórczym, które niekiedy może wymagać naprawdę ogromnej pomysłowości."

Z kolei J. Gleen Brookshear [1 s. 194] pisze „*Jest oczywiście wiele metod rozwiązywania problemów, z których każda może przynieść sukces w określonych sytuacjach …"* i dalej *„ … Wykonanie pierwszego kroku w kierunku rozwiązania oraz podanie sposobu rozszerzenia tego pierwszego spostrzeżenia do pełnego rozwiązania wymaga twórczego myślenia ze strony osoby rozwiązującej, Jest jednak kilka ogólnych metod zaproponowanych przez Polyi i innych, które pomagają zrobić dobry początek."* Do tych ogólnych metod, cytowani autorzy, zaliczają: - szukanie analogii, - stopniowe uściślanie z metodyką zstępującą (dekompozycja) i wstępującą (uogólnienie).

Na szczególną uwagę w kontekście dysponowania modelami referencyjnymi problemów **kSAT** zasługuje technika dekompozycji. Jest tak, bo dekompozycja może być prowadzona z uwzględnieniem różnorodnych aspektów, np. tak jak w przypadku maszyny Turinga w aspekcie danych, struktur sterujących oraz wykonywanych operacji.

Upraszczanie w aspekcie danych może być realizowane w różny sposób. Może polegać to na sprowadzeniu zadania *„do mniejszych zadań tego samego rodzaju i rozwiązując je … połączyć te częściowe rozwiązania w rozwiązanie ostateczne pierwotnego zadania."* [3 s. 97] Takie podejście zostało zastosowane w dyskusji jednostajnej wielomianowej sieci logicznej dla **3SAT** (rozdział 1). Podstawą końcowego rezultatu było wyodrębnienie elementarnego problemu 3SAT, a po uogólnieniu do kSAT, pokazanie dla nich elementarnej sieci logicznej mającej cechy sieci jednostajnej.

Inną strategią dekompozycji może być podział zadnia na „podzadania", według której *„ … Wszystkie zadania częściowe rozwiązuje się w kolejności wzrastania ich rozmiaru, a wyniki przechowuje się w jakiejś strukturze danych, tak aby ułatwić otrzymanie rozwiązań większych zadań."*, [3 s.104] Należy jednak mieć wtedy na uwadze, że *„ ... jeśli wynikiem podziału na podzadania jest drzewo podzadań, to w wielu wypadkach drzewo to rośnie szybko, zazwyczaj wykładniczo."* [7 s. 149]

Wymienione wyżej przesłanki dotyczą problemów **kSAT** - **2SAT** ∈ ***P*** i **3SAT** ∈ ***NP*** są problemami zupełnymi w swoich klasach. Wybór tych problemów jako kandydatów do prób konstruowania

algorytmów równoległych, spełnia podstawowy wymóg rozstrzygania równości klas do których należą ze „*słabszą*" klasą.

4.2.1 Ogólne założenia

W przyjętych założeniach koncepcji algorytmu uwzględnione zostały także kwestie spełniania innych wymogów sformułowanych wobec algorytmów problemów zaliczanych do klasy ***NC***, w tym dotyczących pamięci logarytmicznej i równoległego czasu logarytmicznego.

Po pierwsze, rozpatrywane warianty spełnialności – **2SAT** i **3SAT** – są przedstawione w postaci odpowiadającej ich modelom referencyjnym *Full2SAT* i *Full3SAT*. Ogólnie w modelach referencyjnych *FullkSAT* (k>1) występuje stała wielomianowa liczba jednorodnych elementów (funkcji), równa $n\cdot(n-1)\cdot \ldots \cdot(n-k+1)/k!$. Dla *Full2SAT* jest ich $n\cdot(n-1)/2$, a dla *Full3SAT* $n\cdot(n-1)\cdot(n-2)/6$. Modele referencyjne mają regularną, uporządkowaną strukturę, w której elementy mogą być ponumerowane. W ten sposób wszystkie operacje odnoszące się do nich mogą być realizowane z wykorzystaniem indeksów, a operowanie nimi możliwe jest w pamięci logarytmicznej.

Przyjęcie założenia, że dla konkretnego problemu, konstrukcja algorytmu dotyczyć będzie jego ekwiwalentnej, uporządkowanej postaci modelu referencyjnego, nie wpływa na złożoność całego rozwiązania. Nie podając dokładnego sposobu uporządkowania, wiemy, że można je wykonać równolegle w teoretycznym czasie logarytmicznym - „optymalna sieć sortująca" [3. s.287], także wtedy gdy na klucz składa się kilka kryteriów (w przypadku modelu referencyjnego uporządkowanie odbywa się według dwóch kryteriów – numeru pierwszej zmiennej oraz rozpiętości).

Po drugie, przyjęto, że algorytm zwraca rozwiązanie problemu obliczeniowego. Jest oczywistym, że jeśli istnieje tylko jeden sposób wartościowania zmiennych spełniający formułę, to będzie ono miało postać wektora binarnego. Jeśli jednak formuła może być spełniana na wiele sposobów wartościowania zmiennych (w ogólności może ich być ponadwielomianowa ilość), to trudno sobie wyobrazić możliwość operowania tak obszernym zbiorem wektorów. W takich przypadkach konieczne jest uogólnienie wektora binarnego. Wzorując się na algorytmach genetycznych,

można przyjąć za takie uogólnienie wzorzec podzbioru wektorów binarnych podobnych ze względu na ustalone pozycje (w algorytmach genetycznych taki wzorzec nazywany jest schematem [2 s. 36]). Ostatecznie, rozwiązanie problemu w wersji obliczeniowej będzie mogło mieć postać wzorca(ów) wektorów binarnych.

Kolejny wymóg, jaki musi spełniać algorytm równoległy, ma charakter konstrukcyjny. W ujęciu sieci logicznych, jeśli czas równoległy ma wyrażać się czasem logarytmicznym, to stru-ktura sieci musi mieć postać drzewa binarnego, w którym wyjściowy procesor zwraca wynik.

Ponieważ wcześniej przyjęto założenie, że celem jest konstrukcja algorytmu dla problemu w wersji obliczeniowej, to lementami postulowanej struktury drzewiastej na każdym jej poziomie będą wzorce wektorów rozwiązań. Zbudowanie takiej struktury nie powinno nastręcza trudności. Można w tym miejscu przywołać analogię z multiplikatorem (układ mnożący) realizującym algorytm Booth'a. W odróżnieniu od multipolikatora, w miejsce operacji sumowania iloczynów cząstkowych, tutaj wykonywane będą odpowiednio zdefiniowane operacje składania wzorców wektorów rozwiązań.

Dla operacji składania wzorców wektorów istotne jest nie tylko jej formalne zdefiniowanie, ale istotna jest postać wzorców wektorów oraz sposób ich ustalenia na wejściach struktury drzewa.

Wyjaśnienie tych kwestii dogodnie jest przedstawić poprzez odwołanie się do podstawowych elementów teorii układów kombinacyjnych.

4.2.2 Elementy teorii układów kombinacyjnych

Układy kombinacyjne są jedną z form realizacji układów cyfrowych. Przetwarzają informację przyjmującą przedstawianą najczęściej w postaci sygnałów przyjmujących dwie różne wartości – sygnałów binarnych. W odróżnieniu od innych, układy kombinacyjne charakteryzuje brak elementów pamięci. Oznacza to, że stan wyjść układu jest jednoznacznie określony przez aktualny stan wejść.

Działanie układów kombinacyjnych można opisywać na wiele sposobów - słownie, przy pomocy tabelek itp.. Jednak fizyczna realizacja układu poprzez połączenie za sobą bramek logicznych wymaga formalnego opisu. Korzysta się przy tym z różnych postaci algebry Boole'a: algebry zadań, algebry zbiorów, algebry sygnałów i algebry wektorów binarnych. Poniżej przedstawiono najistotniejsze, wybrane arbitralnie, stwierdzenia odnoszące się do każdej z nich.

Fakt 4.1: Odwzorowanie zbior stanów wejściowych w zbiór stanów wyjściowych układu przez wyrażenie boolowskie nazywane jest funkcją przełączającą. Każda funkcja przełączająca może być rozbita na dwa składniki względem każdej zmiennej.

Rozbicie funkcji $\varphi_1 (x_1, x_2, \dots , x_n)$ względem x_1 można zapisać następująco (dla zwiększenia czytelności symbol koniunkcji "$\wedge$" zastąpiono symbolem "$\cdot$").

$$\varphi (x_1, x_2, \dots , x_n) = x_1 \cdot \varphi (1, x_2, \dots , x_n) \vee \bar{x}_1 \varphi (0, x_2, \dots , x_n)$$

Rozbicie dalej względem x_2 można zapisać w postaci:

$$\varphi (x_1, x_2, \dots , x_n) = x_1 x_2 \cdot \varphi (1, 1, x_3, \dots , x_n) \vee x_1 \bar{x}_2 \varphi (1, 0, x_3, \dots , x_n)$$
$$\vee \bar{x}_1 x_2 \varphi (0, 1, x_3, \dots , x_n) \vee \bar{x}_1 \cdot \bar{x}_2 \varphi (0, 0, x_3, \dots , x_n)$$

Proces rozbicia względem pozostałych zmiennych prowadzi do uzyskania formuły, w której każdy człon jest sumy jest iloczynem wartości funkcji dla konkretnego stanu wejść (jest ich 2^n) przez pełny iloczyn wszystkich zmiennych.

Wprowadzając oznaczenie pełnego iloczynu wszystkich zmiennych jako $K_E(X) = x_1^{e1} \cdot x_2^{e2} \cdot \dots \cdot x_n^{en}$, jego postać dla konkretnego stanu wejść jako wektora $E=(e_1, e_2, \dots, e_n)$ i $e_i \in \{0,1\}$ określana jest w oparciu o zależność $x^e=x$ gdy $e=1$ oraz $x^e=\bar{x}$ gdy $e=0$. Dodatkowo, jeśli przez F^1 oznaczony będzie zbiór wektorów E, dla których funkcja $\varphi(E)$ przyjmuje wartość logiczną *true*, to końcowa formuła rozbicia może być zapisana w postaci wektorowej (postać zbiorów binarnych) następująco:

$$\varphi (x_1, x_2, \dots , x_n) = \varphi (X) = \bigvee_{E \in F^1} K_E(X)$$

Ponadto, w kategorii algebry zbiorów, prawdziwe są zależności:

jeżeli $\varphi_1 (X) = \bigvee K_E(X)$ oraz $\varphi_2 (X) = \bigvee K_E(X)$

$$\text{to } \varphi_1(X)\cdot\varphi_2(X) = \bigvee_{E \in F_1^1 \cap F_2^1} K_E(X) \quad \text{i} \quad \varphi_1(X) \vee \varphi_2(X) = \bigvee_{E \in F_1^1 \cup F_2^1} K_E(X)$$

Postać końcowa rozbicia jest elementem dowodu, że działanie układu kombinacyjnego można opisać przy pomocy sumy pełnych iloczynów (tzw. postać kanoniczna sumy) [6 s. 29]. Przy czym, w wyrażeniu występują tylko takie pełne iloczyny wszystkich zmiennych, dla których wartością funkcji φ jest wartość logiczna *true*.

Fakt 4.2: Przy konstruowaniu układów kombinacyjnych występują sytuacje, w których:

(i) na wejściach $x_1, x_2, \ldots, x_n$ występują wszystkie 2^n możliwe stany
i dla każdego stanu znane są wartości funkcji,
(ii) na wejściach $x_1, x_2, \ldots, x_n$ występują wszystkie 2^n możliwe stany, ale dla niektórych z nich wartość funkcji jest obojętna,
(iii) na wejściach $x_1, x_2, \ldots, x_n$ występują nie wszystkie spośród 2^n możliwych stanów i wtedy nie dla wszystkich stanów wejść mamy określone wartości funkcji, co sprawia, że mamy do czynienia z niezupełną funkcją przełączającą.

Jedną z metod syntezy układów kombinacyjnych w ostatnim przypadku jest rozszerzenie niezupełnej funkcji przełączającej do postaci zupełnej poprzez przypisanie wybranej wartości funkcji (*0* lub *1*) nie występującym stanom wejść. Dzięki temu, oprócz pojedynczej funkcji f_i związanej z jednym wyjściem można wskazać jeszcze inne funkcje f_i', f_i'', itd., będące równoważnymi dla f_i (równoważne w sensie wartości funkcji określającej stan wyjścia).

W rezultacie, zabieg ten często umożliwia uzyskanie uproszczonych wyrażeń opisujących działanie syntezowanego układu. W procesie upraszczania (minimalizacji), wychodząc od postaci kanonicznej sumy, wykorzystując grupowania i wyłączenia przed nawias, uzyskujemy postać normalną sumy. W tak uzyskanym wyrażeniu z reguły nie będą występowały pełne iloczyny wszystkich zmiennych.

Przykład 4.1. Funkcja y zadana jest w postaci tabelki. Jest funkcją niezupełną, bo stan wyjścia dla wektora binarnego

określającego stan wejść $x_1=1$, $x_2=0$ oraz $x_3=1$ jest dowolny, co zostało oznaczone symbolem "*".

$x_3\ x_2\ x_1$	y
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	*
1 1 0	1
1 1 1	1

$$y = \overline{x}_3 \cdot \overline{x}_2 \cdot x_1 \vee \overline{x}_3 \cdot x_2 \cdot x_1 \vee x_3 \cdot x_2 \cdot \overline{x}_1 \vee x_3 \cdot x_2 \cdot x_1 \qquad (1)$$

$$y = \overline{x}_3 \cdot x_1 \cdot (\overline{x}_2 \vee x_2) \vee x_3 \cdot x_2 \cdot (\overline{x}_1 \vee x_1) = \overline{x}_3 \cdot x_1 \vee x_3 \cdot x_2 \qquad (2)$$

$$y = \overline{x}_3 \cdot \overline{x}_2 \cdot x_1 \vee \overline{x}_3 \cdot x_2 \cdot x_1 \vee x_3 \cdot \overline{x}_2 \cdot x_1 \vee x_3 \cdot x_2 \cdot \overline{x}_1 \vee x_3 \cdot x_2 \cdot x_1 \qquad (3)$$

$$y = \overline{x}_3 \cdot x_1 \cdot (\overline{x}_2 \vee x_2) \vee x_3 \cdot x_1 \cdot (\overline{x}_2 \vee x_2) \vee x_3 \cdot x_2 \cdot (\overline{x}_1 \vee x_1) =$$
$$= \overline{x}_3 \cdot x_1 \vee x_3 \cdot x_1 \vee x_3 \cdot x_2 = x_1 \vee x_3 \cdot x_2 \qquad (4)$$

Dla zadanej w tabeli niezupełnej funkcji y, jej postać kanoniczną i postać normalną sumy po grupowaniu i wyłączeniach określają wyrażenia (1) i (2) odpowiednio.

Zupełną funkcję y (po uwzględnieniu stanu wejść oznaczonego "*") w postaci kanonicznej i w postaci normalnej sumy po grupowaniu i wyłączeniach (dwukrotnie użyto składnika $x_3 \cdot x_2 \cdot x_1$) określają wyrażenia (3) i (4) odpowiednio.

Fakt 4.3 Algebra wektorów binarnych jest uogólnieniem algebry sygnałów na wektory.

Jeśli A, B, C mają postać:

$A=(a_1, a_2, \ldots, a_n)$, $B=(b_1, b_2, \ldots, b_n)$ i $C=(c_1, c_2, \ldots, c_n)$ przy $a, b, c \in \{ 0, 1 \}$ to:

$$A \vee B = C \Leftrightarrow a_i \vee b_i = c_i$$
$$A \cdot B = C \Leftrightarrow a_i \cdot b_i = c_i$$
$$A = B \Leftrightarrow a_i = b_i \qquad i = 1, 2, \ldots, n$$

Fakt 4.4 Ciągi $E = (e_1, e_2, \ldots e_n)$ przy e $\in \{ 0, 1, * \}$ mogą być uważane za opis zbioru wektorów binarnych. Symbolu "*" używa się do oznaczenia, że w jego miejsce można użyć dowolnej wartości spośród *0* i *1*. Na przykład, ciąg *(1 * 0)* określa zbiór dwóch wektorów binarnych *{ 100, 110 }*.

Na ciągach E traktowanych jako zbiory można wykonywać operacje algebry zbiorów, np.:

$$(* * 1) \cap (1 * *) = (1 * 1)$$

oraz $$(1 * *) \cup (1 0 1) = (1 * *)$$

Fakt 4.5 Ciągi $E = (e_1, e_2, \ldots e_n)$ przy e $\in \{ 0, 1, * \}$ traktowane jako zbiór wektorów binarnych ułatwiają przejście od zbiorów F do wyrażeń boolowskich poprzez odwzorowanie:

$$\eta_x(e_1, e_2, \ldots e_n) = x_1^{e1} \cdot x_2^{e2} \cdot \ldots \cdot x_n^{en}$$

i przyjmując, że:
$x^e = x$ dla $e = 1$;
$x^e = \overline{x}$ dla $e = 0$;
$x^e = \overline{x} \vee x = 1$ dla $e = *$.

Szarsze omówienie podstaw teorii układów kombinacyjnych można znaleźć w pracy [6 s. xxx]. Wydaje się jednak, że te wybrane i przytoczone w tym miejscu stwierdzenia, wystarczająco uzasadniają przyjęte założenia konstruowanego algorytmu równoległego. W szczególności dotyczy to stwier-dzenia, że możliwe jest operowanie wektorami w opisie układów kombinacyjnych.

W dalszej dyskusji, w opisie konstrukcji kolejnych kroków algorytmu pojawią się odwołania do poszczególnych stwierdzeń.

4.3 Konstrukcje algorytmu – sekwencja kroków

Przyjęte wcześniej założenia, w zasadzie zostały przedstawione w kolejności wskazującej na sekwencję kroków konstruowanego algorytmu: (i) określenie zbioru wektorów na wejściu, (ii) „składanie" wektorów w strukturze drzewa binarnego.

Założono, że punktem wyjścia będzie możliwa do uzyskania równolegle z postaci zapisu *CNF* dwu- i trójspełnialności postać odpowiadająca ich modelom referencyjnym *Full2SAT* i *Full3SAT*. Wyraźnie trzeba podkreślić, że ogólnie tylko taka postać na pewno zawierać będzie w sobie przypadki najgorsze (chociaż nie muszą być określone wprost). Dodatkowo, dla obu problemów w takiej postaci można określić górne ograniczenie złożoności całego algorytmu.

Jeśli dla konstruowanego algorytmu wymagane jest przekształcenie do postaci ogólnie *FullkSAT* i stanowi to pierwszy krok algorytmu, to złożoność całego algorytmu nie może być

lepsza niż złożoność tej fazy. Ponieważ postać modelu referencyjnego uzyskiwana jest w wyniku operacji sortowania, to złożoność wyraża się dla całkowitą pracą $O(n^k)$ przy jednakowej złożoności czasowej $O(log(n))$.

Złożoność fazy ustalania wektorów początkowych zależy od ilości wektorów, ich długości oraz ich struktury. Wskazane jest, by zarówno ich ilość i długość była powiązana w prosty sposób z ilością n zmiennych definiujących problem. Ułatwi to ocenę złożoności operacji i całej fazy algorytmu.

Wyrażenia, które będą służyły do określenia wartości elementtów poszczególnych wektorów, powinny mieć postać umożliwiającą łatwe ustalenie wykorzystywanych dalej wektorów.

Ograniczając się dalej do problemów dwu- i trójspełnialności, wydaje się, że te warunki spełniają postacie wyrażeń *Full2SAT* i *Full3SAT* zapisane następująco:

$Full2SAT(x_1, x_2, \ldots, x_n) = \Theta_1(x_1, \ldots, x_n) \wedge \Theta_2(x_2, \ldots, x_n) \wedge \ldots \wedge \Theta_{n-1}(x_{n-1}, x_n)$

$Full3SAT(x_1, x_2, \ldots, x_n) = \Omega_1(x_1, \ldots, x_n) \wedge \Omega_2(x_2, \ldots, x_n) \wedge \ldots \wedge \Omega_{n-2}(x_{n-2}, x_{n-1}, x_n)$

gdzie: $\Theta_i(x_i, \ldots, x_n)$ jest koniunkcją funkcji $h_a(x_i, x_j)$ dwóch zmiennych dla $j=i+1, i+2, \ldots, n$, przy czym $a \in \{1, 2, \ldots, 15\}$ oznacza numer dowolnej funkcji dwóch zmiennych $h_{1 \div 15}$ (opisuje więc *i-tą* elementarną strukturę wierszową *Full2SAT*)

oraz

$\Omega_i(x_i, \ldots, x_n)$ jest koniunkcją funkcji $f_b(x_i, x_j, x_k)$ dla $j=i+1, \ldots, n$ i $k=j+1, \ldots, n$, przy czym $b \in \{1, 2, \ldots, 255\}$ oznacza numer dowolnej funkcji trzech zmiennych $f_{1 \div 255}$ (opisuje więc *i-tą* elementarną strukturę wierszową *Full3SAT*).

Przy przyjętych oznaczeniach opis struktury wierszowej $\Theta_i(x_i, \ldots, x_n)$ dla $i=1,2, \ldots, n-1$ formuły *Full2SAT* w stosunku do zapisu w modelu referencyjnym nie zmienia się i pozostaje w postaci:

$$\Theta_i(x_i, \ldots, x_n) = h_a(x_i, x_{i+1}) \wedge h_a(x_i, x_{i+2}) \wedge \ldots \wedge h_a(x_i, x_n)$$

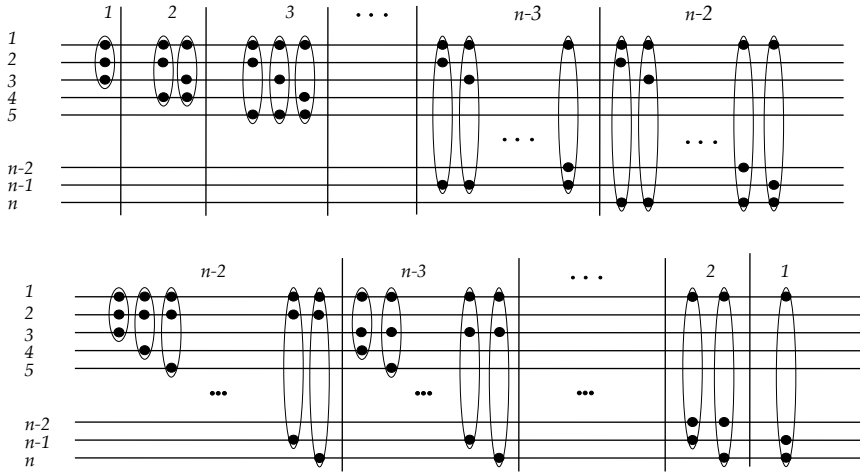
Uwaga 4.1 Postać wyrażenia jednoznacznie wskazuje, że zmienne $x_{i+1}, \ldots x_n$ są względem siebie niezależne.

W przypadku formuły *Full3SAT*, w wyrażeniu opisującym jego strukturę wierszową $\Omega_i(x_i, \ldots, x_n)$ $(i=1, \ldots, n-2)$, przyjęte będzie odmienne w stosunku do modelu referencyjnego

uporządkowanie funkcji. Zmiana uporządkowania funkcji w wyrażeniu Ω_i nie ma wpływu na jego spełnialność i spełnialność całej formuły *Full3SAT*, a przyjęto ją ze względu na prostotę i potrzeby dalszej dyskusji. Ostatecznie, opis struktury wierszowej Ω_i *Full3SAT* ma postać:

$$\Omega_i(x_i, \ldots, x_n) =$$
$$= f_b(x_i, x_{i+1}, x_{i+2}) \wedge f_b(x_i, x_{i+1}, x_{i+3}) \wedge \ldots \wedge f_b(x_i, x_{i+1}, x_{n-1}) \wedge f_b(x_i, x_{i+1}, x_n)$$
$$\wedge f_b(x_i, x_{i+2}, x_{i+3}) \wedge \ldots \wedge f_b(x_i, x_{i+2}, x_{n-1}) \wedge f_b(x_i, x_{i+2}, x_n)$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$\wedge f_b(x_i, x_{n-2}, x_{n-1}) \wedge f_b(x_i, x_{n-2}, x_n)$$
$$\wedge f_b(x_i, x_{n-1}, x_n)$$

Uwaga 4.2 Uporządkowanie funkcji sprawia, że w każdej z podformuł zapisanych w odrębnych wierszach, trzecie zmienne definiujące funkcje f_b są względem siebie niezależne. Wyraźnie widoczne jest to w ilustracji graficznej na rysunku 4.1b.



Rys. 4.1 b Uporządkowanie funkcji f_b w wyrażeniu $\Omega_1(x_1, \ldots, x_n)$ przed i po modyfikacji

4.4 Wektory w 2SAT

4.4.1 Ustalenie wektorów początkowych 2SAT

Ponieważ każda funkcja może być rozbita na dwa składniki względem każdej zmiennej (***Fakt 4.1***), to wyrażenia Θ_i w formułach *Full2SAT* dają się opisać w kategoriach algebry zbiorów.

Każde wyrażenie Θ_i można zapisać następująco:

$$\Theta_i(x_i, \ldots, x_n) = \quad (x_i \cdot h_a(1, x_{i+1}) \vee \overline{x}_i \cdot h_a(0, x_{i+1}))$$
$$\wedge \quad (x_i \cdot h_a(1, x_{i+2}) \vee \overline{x}_i \cdot h_a(0, x_{i+2}))$$
$$\ldots\ldots\ldots$$
$$\wedge \quad (x_i \cdot h_a(1, x_{n-1}) \vee \overline{x}_i \cdot h_a(0, x_{n-1}))$$
$$\wedge \quad (x_i \cdot h_a(1, x_n) \vee \overline{x}_i \cdot h_a(0, x_n))$$

a uwzględniając, że koniunkcje $x_i \cdot h_a(1, x_j) \wedge \overline{x}_i \cdot h_a(0, x_k)$ dla $j, k = \{i+1, \ldots, n\}$ i $k>j$ mają wartość logiczną *false*, ostatecznie

$$\Theta_i(x_i, \ldots, x_n) = x_i \cdot h_a(1, x_{i+1}) \cdot h_a(1, x_{i+2}) \cdot \ldots \cdot h_a(1, x_{n-1}) \cdot h_a(1, x_n)$$
$$\vee \quad \overline{x}_i \cdot h_a(0, x_{i+1}) \cdot h_a(0, x_{i+2}) \cdot \ldots \cdot h_a(0, x_{n-1}) \cdot h_a(0, x_n)$$

Postać końcowa jednoznacznie wskazuje, że w obu członach, zmienne $x_{i+1}, \ldots x_n$ są względem siebie niezależne.

Ponadto, w przypadku uporządkowanej pary dwóch zmiennych funkcji $h_a(x_r, x_s)$ przy $s>r$, jeśli ustalone jest wartościowanie jednej ze zmiennych, np. x_r, to możliwe jest:

(i) stwierdzenie braku wartościowania zmiennej x_s tak by funkcja przyjmowała wartość *true*,
(ii) jednoznaczne (*0* albo *1)* określenie wartościowania zmiennej x_s,
(iii) stwierdzenie dowolnego (*0* lub 1) wartościowania zmiennej x_s.

W efekcie, dla każdej funkcji $h_a(1, x_j)$ oraz $h_a(0, x_j)$, $(j= i+1, \ldots, n)$, w oparciu o jej numer $a \in \{1, 2, \ldots, 15\}$, można ustalić wartościowanie $e_j \in \{0, 1, *\}$ zmiennych x_j lub stwierdzić brak spełnialności (Tabela 4.1).

Tabela 4.1 Względne wartościowanie zmiennej x_s (druga zmienna w uporządkowanej parze) funkcji $h_a(x_r, x_s)$ przy ustalonym wartościowaniu zmiennej x_r (pierwsza w uporządkowanej parze).

$h_a(x_r, x_s)$	h_1		h_2		h_3		h_4		h_5		h_6		h_7		h_8		h_9		h_{10}		h_{11}		h_{12}		h_{13}		h_{14}		h_{15}	
x_r	*0*	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
x_s	*0*	#	#	0	0	0	1	#	*	#	1	0	*	0	#	1	0	1	#	*	0	*	1	1	*	1	1	*	*	*

W ten sposób, dla obu członów możliwe jest zapisanie ciągów (e_i, e_{i+1}, … , e_n) i traktowanie ich jako zbiory wektorów binarnych (***Fakt 4.5***). Jeśli przez E_i^0 i E_i^1 oznaczymy ciągi wartościowań zmiennych x_i, … x_n przy $x_i=0$ i $x_i=1$ odpowiednio, to można $\Theta_i(x_i, \dots, x_n)$ wyrazić w kategoriach algebry zbiorów i zapisać:

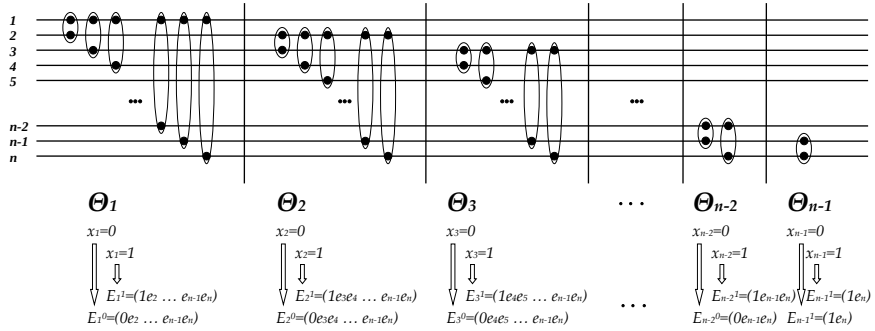
$$\Theta_i(x_i, \dots, x_n) = E_i^0 \cup E_i^1$$

Wynika stąd, że dla **2SAT** zdefiniowanego przez formułę Θ_1 rozwiąza-nie jest trywialne. Wobec faktu, że zmienne x_2, … , x_n w obu członach są względem siebie niezależne, wynik można ustalić w czasie $\boldsymbol{O}(1)$ przy pomocy n procesorów.

Jeśli każde wyrażenie Θ_i można traktować jako sumę wektorów, to wtedy *Full2SAT* także może być opisany w kategoriach algebry zbiorów wektorów binarnych:

$$Full2SAT = (E_1^0 \cup E_1^1) \cap (E_2^0 \cup E_2^1) \cap \dots \cap (E_{n-1}^0 \cup E_{n-1}^1)$$

Jednak, o ile ustalenie wektorów początkowych E_i^0 E_i^1 jest proste (rys. 4.2), to proces ich składania może nie być prosty. Wskazuje na to przedstawiona dalej analiza różnych przypadków reprezentacji *Full2SAT* przez wektory początkowe.



Rys. 4.2 Graficzna ilustracja ustalania wektorów początkowych *Full2SAT*

4.4.2 Analiza składania wektorów *Full2SAT*

Rozpoczynając dyskusję sposobu składania wektorów E_i traktowanych jako zbiór wektorów binarnych (wzorców, schematów), należy mieć na względzie, że jest mało prawdopodobne, aby rozwiązanie problemu obliczeniowego 2SAT (i ogólnie kSAT), jeśli istnieje, dało się wyrazić przy pomocy jednego wzorca. Wskazuje na to przykład prostego problemu

zdefiniowanego wyrażeniem w zapisie *2SAT CNF* $\varphi=(x_1 \vee x_2)(x_1 \vee x_3)$.

Zbiór możliwych kombinacji wartościowań zmiennych spełniających wyrażenie stanowią ciągi $(u_1\ u_2\ u_3) \in \{ 100, 110, 101, 011, 111 \}$. Używając wzorca wektorów binarnych (e_1, e_2, e_3) można to wyrazić w postaci zbioru $\{ (1^{**}), (^*11) \}$, którego elementy nie są rozłączne. Obu wzorcom odpowiada ciąg $(u_1\ u_2\ u_3) = (111)$.

Potencjalny brak możliwości określenia zbioru możliwych kombinacji wartościowań zmiennych spełniających wyrażenie definiujące problem przy pomocy jednego wzorca, sugeruje, że proces składania wzorców wektorów może wymagać etapowania.

Opisując fazę składania wektorów, przyjęto, że:
- przed rozpoczęciem składania wektorów znane będą wszystkie, łącznie $2 \cdot (n-1)$, początkowe wektory E_i^0 i E_i^1 ustalone odrębnie dla każdego wyrażenia Θ_i według reguł określonych w tabeli 4.1, do czego potrzeba n^2 procesorów;
- składanie będzie realizowane w strukturze drzewa binarnego na podobieństwo struktury multiplikatora równoległego, co powinno zapewnić, że operacje składania wektorów będą wykonywane na poziomach struktury o głębokości *log n*;
- inicjowanie składania wektorów E_i i E_{i+1} realizowane na pierwszym poziomie i dalej na kolejnych poziomach, określane jest według reguł podanych w tabeli 4.2 (przypadek stwierdzonego braku spełnialności oznaczany jest symbolem #).

Tabela 4.2

e_j wektora E_i	0				1				*				#			
e_j wektora E_{i+1}	0	1	*	#	0	1	*	#	0	1	*	#	0	1	*	#
e_j wektora wynikowego	0	#	0	#	#	1	1	#	0	1	*	#	#	#	#	#

W przypadku $Full2SAT(x_1, x_2, \ldots, x_n) = \Theta_1(x_1, \ldots, x_n) \wedge \Theta_2(x_2, \ldots, x_n) \wedge \ldots \wedge \Theta_{n-1}(x_{n-1}, x_n)$, każdy człon Θ_i będzie w ujęciu wektorowym reprezentowany przez parę wektorów E_i^0 i E_i^1 . Sąsiednie pary wektorów początkowych (E_i^0, E_i^1) oraz (E_{i+1}^0, E_{i+1}^1) różnią się długością o jedną pozycję (wektory E_{i+1}^0 i E_{i+1}^1 są krótsze). Na kolejnych poziomach ta różnica długości będzie się sukcesywnie podwajała.

Uwaga 4.3 Uwzględniając dyskutowany ***Fakt 4.2*** (rozszerzenie niepełnej funkcji przełączającej), można dokonać wyrównania długości każdej pary sąsiednich wektorów E_i i E_{i+1} poprzez dopisanie na pozycji wyrównującej oznaczenia określającego dowolne wartościowanie. Można tak zrobić i jest to zabieg „neutralny", ponieważ odpowiada to domyślnemu rozszerzeniu wyrażenia Θ_{i+1} o koniunkcję funkcji stałych $TRUE(x_i, x_{i+1}) \wedge TRUE(x_i, x_{i+2}) \wedge \ldots \wedge TRUE(x_i, x_n)$. Jeśli okaże się to potrzebne, to ten zabieg można zastosować do każdego wektora wyrównując go do długości n.

Charakteryzując dalej wektory początkowe $E_i^0 = (0, e_{i+1}, \ldots, e_n)$ i $E_i^1 = (1, e_{i+1}, \ldots, e_n)$, należy zauważyć, że przy wszystkich $e_{i+1}, \ldots, e_n$ jako niezależnych względem siebie, ciągi określające te wektory (zakładając spełnialność każdego Θ_i) mogą być scharakteryzowane następująco:

- wszystkie e_j są jednoznacznie określone (przyjmują wartość *0* albo *1*),
- część e_j jest jednoznacznie określona, a pozostałe dowolnie,
- wszystkie e_j są określone jako dowolne wartościowania.

Ponieważ wektory E_1^0 i E_1^1 w procesie składania wykorzystuje się jako inicjujące cały proces, ich postać może mieć istotne znaczenie. Ogólnie, wyszczególnione trzy postaci ciągów określających wektory E_i^0 i E_i^1 sprawiają, że proces składania wektorów może być mocno zróżnicowany – od elementarnie prostego do wymagającego być może dodatkowych działań wykraczających poza samo składanie.

Kolejne elementy dalszej dyskusji, uwzględniając kilka z wielu różnych odwzorowań problemu *Full2SAT* przez możliwe zestawy wektorów, pokazują sygnalizowane zróżnicowanie i bariery procesu składania wektorów. Analizowane będą sytuacje, w których:

(i) gdy bez względu na postać wektorów E_i^0 i E_i^1 $(i=2, \ldots, n-1)$ w obu wektorach E_1^0 i E_1^1 wszystkie e_j $(j=2, \ldots, n)$ są jednoznacznie określone;
(ii) gdy w postaci wybranego wektora E_1^0 lub E_1^1 wszystkie e_j , $j=2, \ldots, n$ są dowolnie wartościowane;
(iii) gdy w obu wektorach początkowych E_i^0 lub E_i^1 każdego Θ_i część zmiennych jest określona jednoznacznie, a pozostałe dowolnie i mogą się wzajemnie przeplatać.

W sytuacji (i), gdy bez względu na postać wektorów E_i^0 i E_i^1 *(i=2, … ,n-1)* w obu wektorach E_1^0 i E_1^1 wszystkie e_j *(j=2, … , n)* są jednoznacznie określone, składanie wektorów jest elementarnie proste. Przy takich założeniach jest oczywistym, że jeśli formuła jest spełniana, to przynajmniej jeden spośród wektorów E_1^0 i E_1^1 określa wartościowania zmiennych spełniających całą formułę.

Ponieważ założyliśmy, że w wybranym wektorze E_1^0 lub E_1^1 elementy e_j są jednoznaczne, to ich wartości określają jednoznacznie, który z pary wektorów (E_i^0, E_i^1) powinien być uwzględniony w składaniu na pierwszym poziomie. Identycznie będzie dla drugiego wektora z pary $(E_1^0$ lub $E_1^1)$.

Dalej, składanie na kolejnych poziomach polegać będzie na porównywaniu elementów e_j z dwóch sąsiednich wektorów ustalonych na poprzednim poziomie. Porównywanie może być wykonane równocześnie dla wszystkich e_j we wszystkich parach wektorów na danym poziomie drzewa.

Uwaga 4.4 Oczywistym jest, że sprawdzenie spełnialności formuły w takich sytuacjach może być zrealizowane w drzewiastej strukturze składania wektorów w czasie logarytmicznym (głębokość drzewa *log n*). Gdy formuła badanego ***SAT*** jest spełniana, wierzchołek tej struktury powinien zwrócić wynik w postaci E_1^0 lub E_1^1, zależnie od tego który z nich został wybrany dla inicjacji składania.

Uwaga 4.5 Alternatywą dla składania wektorów, jest równoczesne sprawdzenie pozostałych funkcji w $\Theta_2(x_2, \dots, x_n)$, … , $\Theta_{n-1}(x_{n-1}, x_n)$, czy są spełnialne dla ustalonych wartości e_j *(j=2, 3, … , n)* w rozpatrywanym wektorze $E_1^{(0)}$ i $E_1^{(1)}$, do czego potrzeba n^2 procesorów.

W sytuacji (ii), gdy w postaci wybranego wektora E_1^0 lub E_1^1 dla wszystkich e_j , *j=2, … , n* możliwe jest dowolne wartościowanie, proces składania wektorów jest nieco bardziej złożony. Należy przy tym pamiętać, że dla formuły *Full2SAT*, poza sytuacją kiedy wyrażenie Θ_i jest koniunkcją funkcji stałych $\boldsymbol{TRUE}(x_1, x_j)$, *j=2, … , n*, w obu wektorach E_1^0 i E_1^1 nie może zaistnieć jednoczesne dowolne wartościowanie zmiennych x_j przy $x_1=0$ oraz $x_1=1$.

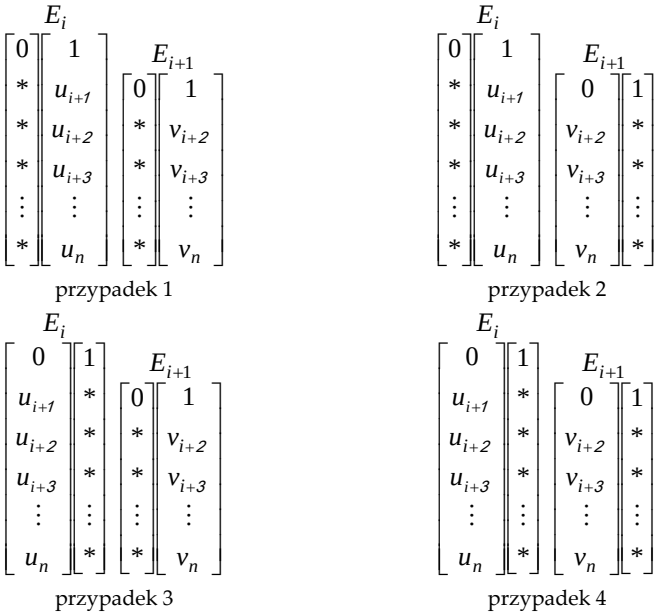
Zatem, jeśli dla Θ_1 przy $x_1=1$ (wektor E_1^1), wszystkie pozostałe zmienne mogą być dowolnie wartościowane, to funkcjami tworzącymi Θ_1 musiały być funkcje h_{10}, h_{11} lub h_{14} (pomijamy funkcję h_{15}, która w *Full2SAT* jest dopuszczalna) i wtedy:

- gdyby były to wyłącznie funkcje h_{11} i h_{14}, to przy przeciwnym wartościowaniu x_1 (tu $x_1=0$), wszystkie zmienne muszą mieć jednoznacznie określone wartościowanie odnotowane w wektorze E_1^0. To pozwala rozstrzygać spełnialność całej formuły danego ***SAT*** w elementarny prosty sposób opisany wcześniej (***Uwagi 4.2 i 4.2***). Jednak, gdy wektor E_1^0 nie rozstrzyga spełnialności całego ***SAT***, dalej pozostaje nam rozstrzyganie dla pierwotnie założonego wartościowania zmiennej x_1 (tj. $x_1=1$ i wektora E_1^1) z uwzględnieniem wyrażeń Θ_2, …, Θ_{n-1} niezależnych od x_1.
- gdyby w wyrażeniu Θ_1 występowała chociażby jedna funkcja h_{10}, to przy przeciwnym wartościowaniu x_1 (tu $x_1=0$ i wektor E_1^0) jest ona niespełniana. Tym samym, dla wartościowań określonych przez wektor E_1^0 stwierdzamy niespełnialność całej formuły danego ***SAT*** i dalej pozostaje nam rozstrzyganie dla pierwotnie założonego wartościowania zmiennej x_1 (tj. $x_1=1$ i wektora E_1^1) z uwzględnieniem wyrażeń Θ_2, …, Θ_{n-1} niezależnych od x_1.

Podobnie można dyskutować, gdy dla Θ_1 przy $x_1=0$ (wektor E_1^0) wszystkie pozostałe zmienne mogą być dowolnie wartościowane. Teraz jednak funkcjami tworzącymi Θ_1 musiałyby być funkcje h_5, h_7 i h_{13} (także tutaj pomijamy funkcje h_{15}) i wtedy:

- gdyby były to wyłącznie funkcje h_7 i h_{13}, to przy przeciwnym wartościowaniu x_1 (tu $x_1=1$), wszystkie zmienne muszą mieć jednoznacznie określone wartościowanie odnotowane w wektorze E_1^1. To pozwala rozstrzygać spełnialność całej formuły danego ***SAT*** w elementarny prosty sposób opisany wcześniej (***Uwagi 4.2 i 4.2***). Jednak, gdy wektor E_1^1 nie rozstrzyga spełnialności całego ***SAT***, dalej pozostaje nam rozstrzyganie dla pierwotnie założonego wartościowania zmiennej x_1 (tj. $x_1=0$ i wektora E_1^0) z uwzględnieniem wyrażeń Θ_2, …, Θ_{n-1} niezależnych od x_1.
- gdyby w wyrażeniu Θ_1 występowała chociażby jedna funkcja h_5, to przy przeciwnym wartościowaniu x_1 (tu $x_1=1$ i wektor E_1^1) jest ona niespełniana. Tym samym, dla wartościowań określonych przez wektor E_1^1 stwierdzamy niespełnialność całej formuły danego ***SAT*** i dalej pozostaje nam rozstrzyganie dla pierwotnie założonego wartościowania zmiennej x_1 (tj. $x_1=0$ i wektora E_1^0) z uwzględnieniem wyrażeń Θ_2, …, Θ_{n-1} niezależnych od x_1.

Przy rozszerzeniu przyjętych założeń dla Θ_1 na pozostałe wyrażenia od Θ_2 do Θ_{n-1}, na pierwszym poziomie drzewa dla dwóch sąsiednich par wektorów ustalonych dla wyrażeń Θ_i i Θ_{i+1} możliwe są wtedy cztery przypadki przedstawione na rys. 4.4.



Rys. 4.4 Postaci sąsiednich wektorów na pierwszym poziomie drzewa składania.

Na przykładzie pierwszego przypadku, składanie sąsiednich wektorów na pierwszym poziomie drzewa można zilustrować następująco:

$$\begin{bmatrix}0\\*\\*\\*\\\vdots\\*\end{bmatrix}\begin{bmatrix}1\\u_{i+1}\\u_{i+2}\\u_{i+3}\\\vdots\\u_n\end{bmatrix}\quad\begin{bmatrix}0\\*\\*\\\vdots\\*\end{bmatrix}\begin{bmatrix}1\\v_{i+2}\\v_{i+3}\\\vdots\\v_n\end{bmatrix}\quad\Rightarrow\quad \overset{(E_i^0\cup E_i^1)\cap(E_{i+1}^0\cup E_{i+1}^1)}{\begin{bmatrix}0\\00\\**\\**\\\vdots\\**\end{bmatrix}\cup\begin{bmatrix}0\\0\ 1\\*\,v_{i+2}\\*\,v_{i+3}\\\vdots\ \vdots\\*\ v_n\end{bmatrix}\cup\begin{bmatrix}0\\1\,0\\**\\**\\\vdots\ \vdots\\**\end{bmatrix}\cup\begin{bmatrix}0\\1\ \ 1\\*\,v_{i+2}\\*\,v_{i+3}\\\vdots\ \vdots\\*\ v_n\end{bmatrix}\cup\begin{bmatrix}1\\u_{i+1}\,0\\u_{i+2}\,*\\u_{i+3}\,*\\\vdots\ \vdots\\u_n\ *\end{bmatrix}\cup\begin{bmatrix}1\\u_{i+1}\\u_{i+1}\\u_{i+3}\\\vdots\\u_n\end{bmatrix}\begin{bmatrix}1\\v_{i+2}\\v_{i+3}\\\vdots\\v_n\end{bmatrix}}$$

$$= \begin{bmatrix} 0 \\ 0 \\ * \\ * \\ \vdots \\ * \end{bmatrix} \cup \begin{bmatrix} 0 \\ \# \\ v_{i+2} \\ v_{i+3} \\ \vdots \\ v_n \end{bmatrix} \cup \begin{bmatrix} 0 \\ \# \\ * \\ * \\ \vdots \\ * \end{bmatrix} \cup \begin{bmatrix} 0 \\ 1 \\ v_{i+2} \\ v_{i+3} \\ \vdots \\ v_n \end{bmatrix} \cup \left(\begin{bmatrix} 1 \\ 0 \\ u_{i+2} \\ u_{i+3} \\ \vdots \\ u_n \end{bmatrix} \cup \begin{bmatrix} 1 \\ 1 \\ u_{i+2} \circ v_{i+2} \\ u_{i+3} \circ v_{i+3} \\ \vdots \\ u_n \circ v_n \end{bmatrix} \right)$$

Spośród wektorów ujętych w nawias, tylko jeden, określając jednoznacznie wartościowania zmiennych x_{i+2}, … , x_n, może potencjalnie rozstrzygać spełnialność wyrażeń Θ_{i+2} do Θ_{n-1} oraz wyrażeń Θ_{i-1}, … , Θ_1. Decyduje o tym wartościowanie u_{i+1} zmiennej x_{i+1}, określone w wektorze E_i^1 wyrażenia Θ_i, jeśli jest spełniane.

Jeśli $u_{i+1}=0$, to rezultat składania zwracać będzie pierwszy wektor z pary ujętej w nawias i wtedy wynikiem składania mogą być trzy wektory. Natomiast jeśli $u_{i+1}=1$, to w rezultacie składania należy uwzględnić drugi wektor, a ponieważ w tym wypadku, porównywane na zgodność są wartościowania określone w wektorach E_i^1 oraz E_{i+1}^1, to gdy parami wszystkie (u_{i+2}, v_{i+2}), … , (u_n, v_n) są zgodne, ostatecznym wynikiem składania mogą być dwa wektory.

Zatem, zależnie od wartościowania u_{i+1}, dla rozpatrywanego pierwszego przypadku dwóch sąsiednich wektorów, wynik składania na pierwszym poziomie drzewa może mieć jedną z trzech postaci, w której zawsze występować będzie wektor określający dowolne wartościowania zmiennych x_{i+2}, … , x_n:

$$\begin{bmatrix} 0 \\ 0 \\ * \\ * \\ \vdots \\ * \end{bmatrix} \cup \begin{bmatrix} 0 \\ 1 \\ v_{i+2} \\ v_{i+3} \\ \vdots \\ v_n \end{bmatrix} \quad \left| \quad \begin{bmatrix} 0 \\ 0 \\ * \\ * \\ \vdots \\ * \end{bmatrix} \cup \begin{bmatrix} 0 \\ 1 \\ v_{i+2} \\ v_{i+3} \\ \vdots \\ v_n \end{bmatrix} \cup \begin{bmatrix} 1 \\ 0 \\ u_{i+2} \\ u_{i+3} \\ \vdots \\ u_n \end{bmatrix} \quad \right| \quad \begin{bmatrix} 0 \\ 0 \\ * \\ * \\ \vdots \\ * \end{bmatrix} \cup \begin{bmatrix} * \\ 1 \\ v_{i+2} \\ v_{i+3} \\ \vdots \\ v_n \end{bmatrix}$$

W pozostałych trzech przypadkach uzyskamy podobne wyniki. W tych pozostałych przypadkach, określenie dowolnego

wartościowania zmiennych x_{i+2}, … , x_n poprzedzać będą kombinacjami (0,1), (1,0) i (1,1) wartościowań zmiennych x_i oraz x_{i+1}.

We wszystkich przypadkach, przy przyjętych założeniach, że dla każdego Θ_i jeden wektor początkowy E_i^0 lub E_i^1 dopuszcza dowolne wartościowanie (ma to miejsce gdy wszystkie funkcje $h_a(x_i, x_j)$ w wyrażeniu Θ_i mają rzęd równy 3), cały ***SAT*** jest spełnialny niezależnie od tego jaką postać będzie miał wynik składania sąsiednich wektorów.

Ogólnie poczynione założenia dla tej sytuacji, nie są realistyczne. Wyrażenia *Full2SAT* mogą być definiowane przez funkcje $h_{15}(x_i, x_j) =$***TRUE***(x_i, x_j) dla $j \in \{i+1, \dots, n\}$ oraz funkcje, których rząd jest równy 1. Tak więc w obu wektorach początkowych E_i^0 lub E_i^1 każdego Θ_i mogą występować dowolne wartościowania *j-tej* zmiennej (zmiennych). Wtedy w procesie składania będą występowały wektory E_i^0 i E_i^1, w których część zmiennych jest określona jednoznacznie, a pozostałe dowolnie i mogą się wzajemnie przeplatać. Odpowiada to sytuacji ilustrowanej prostym przykładem 4.3.

Przykład 4.3 Problem zdefiniowany jest w postaci modelu referencyjnego *Full2SAT* $(x_1, x_2, x_3, x_4, x_5)$ i wyraża się formułą:

$$h_{14}(x_1,x_2) \wedge h_9(x_1,x_3) \wedge h_{10}(x_1,x_4) \wedge h_{14}(x_1,x_5) \wedge h_6(x_2,x_3) \wedge h_{15}(x_2,x_4) \wedge h_{13}(x_2,x_5) \wedge h_{11}(x_3,x_4) \wedge h_{10}(x_3,x_5) \wedge h_{11}(x_4,x_5)$$

gdzie: $\Theta_1 = h_{14}(x_1,x_2) \wedge h_9(x_1,x_3) \wedge h_{10}(x_1,x_4) \wedge h_{14}(x_1,x_5)$,
$\Theta_2 = h_6(x_2,x_3) \wedge h_{15}(x_2,x_4) \wedge h_{13}(x_2,x_5)$,
$\Theta_3 = h_{11}(x_3,x_4) \wedge h_{10}(x_3,x_5)$
oraz $\Theta_4 = h_{11}(x_4,x_5)$,

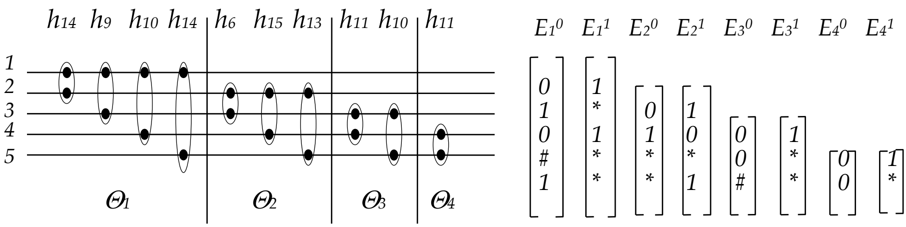
Rozpatrywany problem jest spełnialny dla wartościowań zmiennych $x_1, \dots, x_5$ określonych przez ciągi $(u_1, u_2, u_3, u_4, u_5) \in \{(10100), (10110), (10111)\}$. Na rys. 4.5 pokazano jak w przestrzeni rozwiązań reprezento-wanej binarnie, wartościowane są poszczególne funkcje tworzące rozpatrywany problem oraz całe wyrażenie *Full2SAT*.

x_1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
x_2	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
x_3	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
x_4	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
x_5	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

$h_{14}(x_1,x_2)$	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
$h_9(x_1,x_3)$	1 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1
$h_{10}(x_1,x_4)$	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
$h_{14}(x_1,x_5)$	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
$h_6(x_2,x_3)$	0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0
$h_{15}(x_2,x_4)$	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
$h_{13}(x_2,x_5)$	1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
$h_{11}(x_3,x_4)$	1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1
$h_{10}(x_3,x_5)$	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
$h_{11}(x_4,x_5)$	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1
Full2SAT	0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

Rys. 4.5 Przestrzeń rozwiązań przykładowego problemu *Full2SAT*.

Na rys. 4.6 przedstawiono graficzną ilustrację rozpatrywanego problemu, podając jednocześnie postać wszystkich wektorów E_i^0 i E_i^1.



Rys. 4.6 Ilustracja graficzna przykładowego problemu *Full2SAT*.

Ustalone i pokazane wektory E_i^0 i E_i^1 pozwalają wstępnie stwierdzić, czy odpowiednie wyrażenia Θ_i w formule problemu mogą być spełniane, a jeśli tak to jaki sposób wartościowania zmiennych zapewni ich spełnialność.

W przykładzie, dla $x_1=0$ oraz $x_3=0$ wyrażenia Θ_1 i Θ_3 są niespełniane odpowiednio, bo występujące w Θ_1 i Θ_3 funkcje h_{10} wtedy przyjmują wartość *false*. Odnotowane jest to w wektorach E_1^0 i E_3^0 symbolem #.

Po wstępnej analizie, które z wektorów początkowych należy uwzględnić w procesie składania, na kolejnych poziomach realizowany jest proces składania ilustrowany poniżej.

E_1^1 E_2^0 E_2^1 E_3^1 E_4^0 E_4^1

$$\begin{bmatrix}1\\*\\1\\*\\*\end{bmatrix} \qquad \begin{bmatrix}0\\1\\*\\*\end{bmatrix}\begin{bmatrix}1\\0\\*\\1\end{bmatrix} \qquad \begin{bmatrix}1\\*\\*\end{bmatrix} \qquad \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\*\end{bmatrix}$$

$E_1^1 \cap (E_2^0 \cup E_2^1)$ $\qquad$ $E_3^1 \cap (E_4^0 \cup E_4^1)$

$$\begin{bmatrix}1\\0\\1\\*\\*\end{bmatrix} \qquad \begin{bmatrix}1\\0\\0\end{bmatrix} \cup \begin{bmatrix}1\\1\\*\end{bmatrix}$$

$E_1^1 \cap (E_2^0 \cup E_2^1) \cap E_3^1 \cap (E_4^0 \cup E_4^1)$

$$\begin{bmatrix}1\\0\\1\\0\\0\end{bmatrix} \cup \begin{bmatrix}1\\0\\1\\1\\*\end{bmatrix}$$

W wyniku procesu składania uzyskaliśmy wektor jednoznacznie określający wartościowania zmiennych $x_1, \ldots, x_5$ spełniające formułę ***SAT*** oraz wektor-szablon, określający dwa kolejne sposoby wartościowania zmiennych spełniających formułę ***SAT***.

Przykładowy problem, chociaż jest bardzo prosty, sygnalizuje pewne problemy, z którymi można się zetknąć w procesie składania. Wyrażenie Θ_2 jest reprezentowane przez dwa wektory E_2^0 *i* E_2^1. Wobec dowolnego wartościowania zmiennej x_2 w wektorze E_1^1, oba wektory muszą być uwzględnione w pro-cesie składania. Co prawda, w przykładzie, wektor E_1^1 składany jest ostatecznie tylko z wektorem E_2^0 i wynikiem jest jeden wektor, ale w ogólnym przypadku dla dwóch sąsiednich par wektorów określonych przez Θ_i i Θ_{i+1}, wynikiem składania mogą być cztery wektory.

$$\underbrace{\begin{bmatrix}0\\*\\*\\1\\*\\*\end{bmatrix} \cup \begin{bmatrix}1\\*\\0\\1\\*\\0\end{bmatrix}}_{E_i} \cap \underbrace{\begin{bmatrix}0\\*\\*\\1\\0\end{bmatrix} \cup \begin{bmatrix}1\\*\\1\\*\\0\end{bmatrix}}_{E_{i+1}} = \underbrace{\begin{bmatrix}0\\*0\\**\\1*\\*1\\*0\end{bmatrix} \cup \begin{bmatrix}0-\\*1\\**\\11\\**\\*0\end{bmatrix} \cup \begin{bmatrix}1-\\*0\\0*\\1*\\*1\\00\end{bmatrix} \cup \begin{bmatrix}1-\\*1\\0*\\11\\**\\00\end{bmatrix}}_{(E_i^0 \cup E_i^1) \cap (E_{i+1}^0 \cup E_{i+1}^1)} = \begin{bmatrix}0\\0\\*\\1\\1\\0\end{bmatrix} \cup \begin{bmatrix}0\\1\\*\\1\\*\\0\end{bmatrix} \cup \begin{bmatrix}1\\0\\0\\1\\1\\0\end{bmatrix} \cup \begin{bmatrix}1\\1\\0\\1\\*\\0\end{bmatrix}$$

Jeśli podobnie byłoby dla następnych par wektorów określonych przez wyrażenia Θ_{i+2} i Θ_{i+3}, to mogłoby to prowadzić aż do szesnastu wynikowych wektorów przy składaniu na kolejnym poziomie drzewa itd., i byłby to dalej wykładniczy przyrost. Dzieje się tak za sprawą pośredniej zależności zmiennych x_{i+2}, … , x_n z wektorów E_{i+1} od zmiennej x_i w wektorach E_i.

Wynika z tego, że aby proces składania wektorów mógł być kontynuowany efektywnie, wartościowania zmiennych we wszystkich wektorach muszą zależeć zawsze od tej samej zmiennej (zmiennych).

Z taką możliwością już się zetknęliśmy. Sformułowany warunek efektywnego prowadzenia składania wektorów spełnia struktura wierszowa $\Omega_i(x_i, \dots, x_n)$ problemu *Full3SAT* (***Uwaga 4.2*** i jej ilustracja rys. 4.1 b). Możemy zatem, do rozwiązania *Full2SAT* wykorzystać tą postać wyrażenia, bo znane jest przekształcenie *Full2SAT* do takiej postaci.

Istota tego przekształcenia sprowadza się do określenia funkcji trzech zmiennych w oparciu o dwie funkcje dwóch zmiennych. Szczegóły przekształcenia, oznaczanego $\boldsymbol{P}_v$, zostały zawarte w rozdziale 3 (str. xx). Przekształcenie to jest możliwe do wykonania w czasie $\boldsymbol{O}(1)$ przy całkowitej pracy n^2 procesorów.

4.5 Wektory w 3SAT

4.5.1 Ustalenie wektorów początkowych 3SAT

Postępując podobnie w odniesieniu do wyrażeń Ω_i określających *Full3SAT*, można pokazać, że także w tym przypadku problem daje się wyrazić w kategoriach algebry zbiorów.

Jeśli dla formuły $\Omega_i(x_i, \dots, x_n)$

$$f_b(x_i, x_{i+1}, x_{i+2}) \wedge f_b(x_i, x_{i+1}, x_{i+3}) \wedge \dots \wedge f_b(x_i, x_{i+1}, x_{n-1}) \wedge f_b(x_i, x_{i+1}, x_n)$$
$$\wedge f_b(x_i, x_{i+2}, x_{i+3}) \wedge \dots \wedge f_b(x_i, x_{i+2}, x_{n-1}) \wedge f_b(x_i, x_{i+2}, x_n)$$
$$\dots\dots\dots\dots\dots$$
$$\wedge f_b(x_i, x_{n-2}, x_{n-1}) \wedge f_b(x_i, x_{n-2}, x_n)$$
$$\wedge f_b(x_i, x_{n-1}, x_n)$$

ograniczymy się do części stanowiącej pierwszą linię zapisu , tj.:

$$f_b(x_i, x_{i+1}, x_{i+2}) \wedge f_b(x_i, x_{i+1}, x_{i+3}) \wedge \ldots \wedge f_b(x_i, x_{i+1}, x_{n-1}) \wedge f_b(x_i, x_{i+1}, x_n)$$

i dokonamy rozbicia na dwa składniki względem zmiennej x_i, to uzyskamy postać:

$$\begin{aligned} & (x_i \cdot f_b(1, x_{i+1}, x_{i+2}) \vee \overline{x}_i \cdot f_b(0, x_{i+1}, x_{i+2})) \\ \wedge \; & (x_i \cdot f_b(1, x_{i+1}, x_{i+3}) \vee \overline{x}_i \cdot f_b(0, x_{i+1}, x_{i+3})) \\ & \ldots\ldots\ldots \\ \wedge \; & (x_i \cdot f_b(1, x_{i+1}, x_{n-1}) \vee \overline{x}_i \cdot f_b(0, x_{i+1}, x_{n-1})) \\ \wedge \; & (x_i \cdot f_b(1, x_{i+1}, x_n) \vee \overline{x}_i \cdot f_b(0, x_{i+1}, x_n)) = \end{aligned}$$

$$= x_i \cdot f_b(1, x_{i+1}, x_{i+2}) \cdot f_b(1, x_{i+1}, x_{i+3}) \ldots f_b(1, x_{i+1}, x_{n-1}) \cdot f_b(1, x_{i+1}, x_n) \vee$$
$$\vee \overline{x}_i \cdot f_b(0, x_{i+1}, x_{i+2}) \cdot f_b(0, x_{i+1}, x_{i+3}) \ldots f_b(0, x_{i+1}, x_{n-1}) \cdot f_b(0, x_{i+1}, x_n)$$

Tym razem, w odróżnieniu od przypadku rozbicia funkcji określającej Θ_i problemu *Full2SAT*, zmienne definiujące funkcje f_b w obu członach sumy nie są niezależne względem siebie. Mimo tego, że dla funkcji trzech zmiennych możliwe jest określenie sposobu wartościowania pozostałych dwóch zmiennych (również jako jednoznaczne wartościowanie lub dowolne), to ustalenie wektora wynikowego traktowanego jako zbiór wektorów binarnych nie jest tak proste jak w przypadku Θ_i.

Możemy jednak skorzystać z niekanonicznej postaci opisu funkcji trzech zmiennych, gdy ustalona jest wartość jednej ze zmiennych, wykorzystując w zapisie funkcje dwóch pozostałych nieokreślonych zmiennych, jak w przykładzie niżej.

Przykład 4.2 $f_{90}(x_1, x_2, x_3) = x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \vee x_1 x_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \vee \overline{x}_1 \cdot x_2 \cdot x_3 =$
$= x_1 \cdot (\overline{x}_2 \cdot \overline{x}_3 \vee x_2 \cdot \overline{x}_3) \vee \overline{x}_1 \cdot (\overline{x}_2 \cdot x_3 \vee x_2 \cdot x_3) =$
$= x_1 \cdot h_3(x_2, x_3) \vee \overline{x}_1 \cdot h_{12}(x_2, x_3)$

$\equiv \; x_1 \cdot f_{90}(1, x_2, x_3) \vee \overline{x}_1 \cdot f_{90}(0, x_2, x_3)$

Wykorzystując opisane podejście, wyrażenie uzyskane w wyniku rozbicia rozważanej części Ω_i względem zmiennej x_i , tj.

$$x_i \cdot f_b(1, x_{i+1}, x_{i+2}) \cdot f_b(1, x_{i+1}, x_{i+3}) \ldots f_b(1, x_{i+1}, x_{n-1}) \cdot f_b(1, x_{i+1}, x_n) \vee$$
$$\vee \overline{x}_1 \cdot f_b(0, x_{i+1}, x_{i+2}) \cdot f_b(0, x_{i+1}, x_{i+3}) \ldots f_b(0, x_{i+1}, x_{n-1}) \cdot f_b(0, x_{i+1}, x_n)$$

przyjmie postać:

$$x_i \cdot h_a(x_{i+1}, x_{i+2}) \cdot h_a(x_{i+1}, x_{i+3}) \ldots h_a(x_{i+1}, x_{n-1}) \cdot h_a(x_{i+1}, x_n) \vee$$
$$\vee \overline{x}_1 \, h_a(x_{i+1}, x_{i+2}) \cdot h_a(x_{i+1}, x_{i+3}) \ldots h_a(x_{i+1}, x_{n-1}) \cdot h_a(x_{i+1}, x_n)$$

Występujące w obu członach sumy koniunkcje funkcji h_a mają postać wyrażeń elementarnej struktury wierszowej Θ_i *Full2SAT*. Należy tylko mieć na uwadze, że w istocie każda z nich określa

inną rzeczywistą postać podproblemu *Full2SAT*, zależną od sposobu wartościowania zmiennej x_i.

Uwzględniając poczynioną uwagę i wprowadzając w wyrażeniu Ω_i *Full3SAT* oznaczenia $\Theta_{i+1}^{(0)}$ i $\Theta_{i+1}^{(1}$ dla określenia elementarnych struktur wierszowych *Full2SAT* przy $x_i = 0$ oraz przy $x_i = 1$ odpowiednio, rozpatrywana część $f_b(x_i, x_{i+1}, x_{i+2}) \wedge f_b(x_i, x_{i+1}, x_{i+3}) \wedge \ldots \wedge f_b(x_i, x_{i+1}, x_{n-1}) \wedge f_b(x_i, x_{i+1}, x_n)$ wyrażenia Ω_i może być przedstawiona w postaci:

$$x_i \cdot \Theta_{i+1}^{(1)} \vee \overline{x}_i \; \Theta_{i+1}^{(0)}$$

Postępując analogicznie dla kolejnych wyodrębnionych części wyrażenia Ω_i, tj. rozbijając każde odrębnie na dwa składniki względem x_i oraz stosując podejście oparte na niekanonicznej postaci opisu funkcji trzech zmiennych, możemy wyrażenie Ω_i przedstawić następująco:

$$\Omega_i = (x_i \cdot \Theta_{i+1}^{(1)} \vee \overline{x}_i \; \Theta_{i+1}^{(0)}) \cdot (x_i \cdot \Theta_{i+2}^{(1)} \vee \overline{x}_i \; \Theta_{i+2}^{(0)}) \cdot \ldots \cdot (x_i \cdot \Theta_{n-1}^{(1)} \vee \overline{x}_i \; \Theta_{n-1}^{(0)}) =$$
$$= (x_i \cdot \Theta_{i+1}^{(1)} \cdot \Theta_{i+2}^{(1)} \cdot \ldots \cdot \Theta_{n-1}^{(1)}) \vee (\overline{x}_i \cdot \Theta_{i+1}^{(0)} \cdot \Theta_{i+2}^{(0)} \cdot \ldots \cdot \Theta_{n-1}^{(0)})$$

Występujące w obu członach sumy powyższego zapisu koniunkcje $\Theta_{i+1}^{(1)} \cdot \Theta_{i+2}^{(1)} \cdot \ldots \cdot \Theta_{n-1}^{(1)}$ oraz $\Theta_{i+1}^{(0)} \cdot \Theta_{i+2}^{(0)} \cdot \ldots \cdot \Theta_{n-1}^{(0)}$ mają postać wyrażeń opisujących $Full2SAT(x_{i+1}, x_{i+2}, \ldots, x_n)$ z podobnym poczynionym wcześniej zastrzeżeniem, tj., że w istocie każde z nich określa inną postać problemu *Full2SAT*, zależną od sposobu wartościowania zmiennej x_i. Jeśli dla zaznaczenia zależności postaci *Full2SAT* od sposobu wartościowania zmien-nej x_i, użyjemy górnego indeksu, to Ω_i można wyrazić nastę-pująco:

$$\Omega_i = x_i \cdot Full2SAT^{(1)}(x_{i+1}, x_{i+2}, \ldots, x_n) \vee \overline{x}_i \cdot Full2SAT^{(0)}(x_{i+1}, x_{i+2}, \ldots, x_n)$$

Ponieważ każda z tak określonej postaci *Full2SAT* jako odmienny problem może być rozpatrywana w kategoriach algebry zbiorów, to i wierszowa struktura Ω_i *Full3SAT* będzie mogła być zapisana także w ten sposób.

Pozostając przy przyjętych dla wyrażeń Θ_i *Full2SAT* oznaczeniach wektorów $E_i^{(0)}$ i $E_i^{(1)}$, opatrując je dodatkowym górnym indeksem wskazującym na postać *Full2SAT* – $E_i^{1,(0)}$ dla $Full2SAT^{(1)}(x_{i+1}, x_{i+2}, \ldots, x_n)$ i $E_i^{0,(1)}$ dla $Full2SAT^{(0)}(x_{i+1}, x_{i+2}, \ldots, x_n)$ oraz wprowadzając wektory $E_i^{(1)} = [1^{***} \ldots *]$ i $E_i^{(0)} = [0^{***} \ldots *]$, wyrażenie Ω_i przyjmie postać:

$$\Omega_i = E_i^{(1)} \cap (E_{i+1}^{1,(0)} \cup E_{i+1}^{1,(1)}) \cap (E_{i+2}^{1,(0)} \cup E_{i+2}^{1,(1)}) \cap \ldots \cap (E_{n-1}^{1,(0)} \cup E_{n-1}^{1,(1)})$$
$$\cup E_i^{(0)} \cap (E_{i+1}^{0,(0)} \cup E_{i+1}^{0,(1)}) \cap (E_{i+2}^{0,(0)} \cup E_{i+2}^{0,(1)}) \cap \ldots \cap (E_{n-1}^{0,(0)} \cup E_{n-1}^{0,(1)})$$

wyrażające je w kategoriach algebry zbiorów.

4.6 Podsumowanie

W przeprowadzonej dyskusji klasa NC będąc związaną z pojęciem równoległości obliczeń była rozpatrywaną w modelu sieci logicznych.

Sieci logiczne mające fizyczną formę realizacji w postaciu układów kombinacyjnych, w dyskusji były opisywane z użyciem algebry wektorów. Na przykładach problemu **2SAT** i **3SAT** pokazano, że w obu przypadkach po ustaleniu wektorów początkowych możliwe jest ich składanie w strukturze drzewa binarnego na podobieństwo struktury multiplikatora równoległego. Prowadzenie w ten sposób procesu składania wektorów rozwiązań częściowych problemu **2SAT** rozstrzyga jego przynależność do klasy NC. Ponieważ w podobny sposób prowadzone jest składanie wektorów **3SAT**, to rozstrzygana jest także jego przynależność także do klasy NC.

W efekcie relacja inkluzji $NC \subseteq P \subseteq NP$ sprowadza się do równości klas *NC, P i NP*.

BIBLIOGRAFIA

[1] Brookshear Glenn J., Informatyka w ogólnym zarysie, Warszawa, WNT, 2003

[2] Goldberg David E., Algorytmy genetyczne i ich zastoso-wania, Warszawa, WNT, 1998

[3] Harel David, Rzecz o istocie informatyki. Algorytmika, Warszawa, WNT, 1992

[4] Papadimitriou Christos, Złożoność obliczeniowa, Warszawa, WNT, 2007

[5] Sipser Michael, Wprowadzenie do teorii obliczeń, Warszawa, WNT, 2009

[6] Traczyk Wiesław, Układy cyfrowe. Podstawy teoretyczne i metody syntezy, Warszawa, WNT, 1986

[7] Wirth Niklaus, Algorytmy + struktury danych = programy, Warszawa, WNT, 1980

R o z d z i a ł p i ą t y

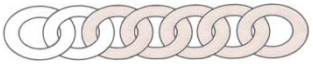
Dyskusja problemu *nwd(a,b)*

Marek Malinowski

Technologie Teleinformatyczne

Tytułową dyskusję i opis rozwiązania poprzedzają uwagi na temat luki algorytmicznej, cech algorytmów logarytmicznych i podobieństwa problemów klasyfikowanych do różnych klas złożoności. Przywołano także kilka znanych podstawowych faktów, które pozwalają poważnie traktować hipotezę o równości klas ***NC = P = NP***.
Ostatecznie opisano koncepcję rozwiązania problemu **nwd**(a,b). Przeanalizowany został jeden z wariantów możliwych ilorazów $s=a/b=p/q$, w którym p i q są liczbami względem siebie pierwszymi i są liczbami nieparzystymi. Pokazano, że dla tego przypadku istnieje kryterium umożliwiające wyszukiwanie wartości tego ilorazu w określonym zbiorze. Wykrzystanie tego kryterium umożliwia zastosowanie metody połowienia i skonstruowanie algorytmu równoległego o czasowej złożoności logarytmicznej.
Konstrukacja algorytmu równoległego dla **nwd**(a,b) sprawia, że klasy *NC* i *P* są równe.

5



nic nie jest tak zwodnicze jak oczywistość prawdy

[R. Ackoff], The Art of Problem Solving – tłum. wł.]

5.1 Wstęp – geneza, cel i zakres pracy

Zdecydowana większość prac koncentruje się na udowo-nieniu jednej z dwóch hipotez odnoszących się do relacji klas *P* i *NP*, formułowanych jako kwestia *P versus NP*. Mają temu służyć różnorodne techniki, formalizmy i modele abstrakcji, którymi operuje teoria złożoności obliczeniowej.

Alternatywą dla wybitnie teoretycznego kierunku prac jest podejście o wymiarze praktycznym. Sprowadza się ono do prób konstruowania algorytmów zamykających lub zmniejszających luki algorytmiczne konkretnych problemów kwalifikowanych do określonych klas złożoności.

Wyniki prezentowane w opracowaniu są efektem właśnie takiego kierunku prac. Kierunku mającego charakter „inżynierski", przy czym wykorzystywany jest bogaty znany dorobek teoretyczny.

Z uwagi na niekonwencjonalny sposób analizy tytułowej kwestii, w niniejszym opracowaniu przywoływane będą podstawowe fakty, wyraziście komentowane szczególnie w książce Dawida Harela „*Rzecz o istocie informatyki. Algorytmika*" [W-wa, WNT 1992].

***Fakt 1**:* Istnieje relacja inkluzji $NC \subseteq P \subseteq NP$, gdzie *NC* – klasa Nicka, charakteryzująca problemy mające równoległe rozwiązania w czasie co najwyżej polilogarytmicznym $O(\log^k \mathbf{N})$; *P* – klasa problemów mających algorytmy wielomianowe; *NP* – klasa problemów o niedeterministycznych algorytmach wielomianowych.

Fakt 2: Problemy *NP* mogą należeć lub nie do *NC* – tego nie wiemy. Potrafimy pokazać póki co, że mają równoległe rozwiązania w czasie wielomianowym.

Fakt 3: Wszystkie problemy z klasy *NC* należą do *P*. Nie wiadomo jednak, czy jest prawdziwa relacja odwrotna, tj. czy istnieje problem łatwo rozwiązywalny w sensie sekwencyjnym (należący do *P*), ale nie w sensie równoległym. Na przykład, znany jest algorytm wielomianowy dla problemu znajdywania największego wspólnego dzielnika dwóch liczb *nwd(a,b)*), ale nikt nie pokazał dla tego problemu algorytmu wykorzystującego równoległość. Stąd przypuszczenie, że *nwd(a,b)* nie należy do *NC*.

Przywołane powyżej fakty upoważniają sformułowanie następującego wniosku:

Wniosek: Jeśli dla dowolnego problemu *NP* istnieje szybkie rozwiązanie równoległe (w czasie co najwyżej polilogarytmicznym), które wymaga wielomianowej liczby procesów, to *P=NP* i *NC=P*.

Tak sformułowany wniosek stanowi podstawę tezy pracy, że dla problemów z klasy problemów *NP-zupełnych* istnieją algorytmy równoległe w czasie co najwyżej polilogarytmicznym i wymagające co najwyżej wielomianowej liczby procesorów, czyli że relacje zawierania klas $NC \subseteq P \subseteq NP$ przyjmują postać relacji równości $NC = P = NP$.

Hipoteza ta w świetle dotychczasowych wyników badań w dziedzinie teorii złożoności wydaje się nieprawdopodobna. Nie można jej jednak *a priori* odrzucić, bo jednak istnieją przesłanki przemawiające za nią, jak chociażby fakt, że różne warianty tych samych lub pokrewnych problemów kwalifikowane są do sąsiednich klas.

Z punktu widzenia stawianej hipotezy, byłoby dobrze by z pary takich pokrewnych problemów klasyfikowanych do sąsiednich klas, ten należący do klasy *P* charakteryzował się czasowa złożonością logarytmiczną (niekoniecznie musiałby to być problem *P-zupełny*).

5.1.1 Cechy algorytmów logarytmicznych

Analiza znanych algorytmów o złożoności czasowej $O(\log^k N)$ dla szeregu problemów należących do klasy *P*, pozwala stwierdzić, że w wielu przypadkach podstawą ich konstruowania jest metoda połowienia, bądź inne metody iteracyjne – jednokrokowe (styczne Newtona) lub dwu- i wielokrokowe.

Wspólną cechą tych algorytmów jest to, że dla ich zainicjowania konieczne jest określenie krańców obszaru rozwiązań i sukcesywne jego zawężanie w kolejnych krokach iteracji. Ponadto musimy dysponować kryterium, według którego proces zawężania będzie realizowany.

Wyszczególnione cechy algorytmów logarytmicznych stanowić będą wymogi stawiane wobec problemu kwalifikowanego do klasy *NP*, kandydata do analizy możliwości skonstruowania dla niego algorytmu w najgorszym razie polilogarytmicznego.

5.1.2 Typowanie problemu NP

Spośród różnych problemów *NP* (problemy dopasowania – „układanki", problemy grafowe, warianty spełnialności), na szczególną uwagę zasługuje problem rozkładu liczby na czynniki proste z jego przypadkiem *mult(n)*. Na potrzeby dalszych rozważań, problem *mult(n)* zdefiniujemy go jako poszukiwanie dwóch czynników *p* oraz *q* będących liczbami pierwszymi, gdy znany jest ich iloczyn.

Za takim wyborem przemawia to, że każdy z czynników *p*, *q* iloczynu $n = p \cdot q$ może być określony przez przynależność do przedziału $(2^{i-1}, 2^i-1)$, $i \in N$. Zakładając, że iloczyn $n = p \cdot q$ należy do przedziału $(2^{2k-1}, 2^{2k}-1)$ możemy rozpatrywać nie więcej niż $2k$ przypadków. Powyższe spostrzeżenie można zilustrować posługując się modelem binarnego multiplikatora sprzętowego. Dla $n \in (2^7, 2^8-1)$ mamy do rozpatrzenia dwa przypadki (*rys. 1a,b*).

Dodatkowo za takim wyborem przemawia pokrewieństwo problemu faktoryzacji z problemem *nwd(a,b)*, o którym wiemy, że należy do *P*. To pokrewieństwo polega na tym, że problem *nwd(a,b)* dla liczb *a* i *b* można rozwiązać poprzez faktoryzację

każdej z nich, a następnie obliczenie iloczynu wspólnych czynników pierwszych.

			p_3	p_2	p_1	p_0	
			q_3	q_2	q_1	q_0	
			p_3q_0	p_2q_0	p_1q_0	p_0q_0	
		p_3q_1	p_2q_1	p_1q_1	p_0q_1		
	p_3q_2	p_2q_2	p_1q_2	p_0q_2			
p_3q_3	p_2q_3	p_1q_3	p_0q_3				
n_7	n_6	n_5	n_4	n_3	n_2	n_1	n_0

		p_3	p_2	p_1	p_0	
		q_3	q_2	q_1	q_0	
		p_3q_0	p_2q_0	p_1q_0	p_0q_0	
	p_3q_1	p_2q_1	p_1q_1	p_0q_1		
p_3q_2	p_2q_2	p_1q_2	p_0q_2			
p_3q_3	p_2q_3	p_1q_3	p_0q_3			
n_6	n_5	n_4	n_3	n_2	n_1	n_0

Rys. 1 a,b

Problem faktoryzacji spełnia wszystkie, poza jednym, konieczne postulaty dla istnienia algorytmu logarytmicznego. Przypomnijmy – liniowa ilość procesorów odpowiadająca ilości koniecznych do rozpatrzenia przedziałów lokalizujących jeden z poszukiwanych czynników; precyzyjnie zdefiniowane krańce przedziałów poszukiwania rozwiązań.

Zasadniczym kwestią pozostaje pytanie, jak sformułować kryterium umożliwiające prowadzenie iteracyjnego zawężania obszaru poszukiwań w każdym z prowadzonych procesów.

W sformułowaniu takiego kryterium pomocne będą dwa spostrzeżenia. Pierwsze dotyczy problemu *nwd(a,b)*, drugie problemu *mult(n)* dyskutowanego w kolejnym rozdziale.

5.2 Dyskusja problemu *nwd(a,b)*

O tym, że problem *nwd(a,b)* można rozwiązać pośrednio poprzez faktoryzację liczb *a* i *b*, była mowa wyżej. W tym miejscu, analizując algorytm Euklidesa można stwierdzić, że operacje modularne w nim wykonywane „ukrywają", przesłaniają nam w istocie fakt wykonania sekwencji operacji dzielenia. Jeśli nie pójdziemy za daleko i poprze-staniemy na wyniku pierwszej operacji dzielenia, to jego wynik określa stosunek liczb *a* i *b*, ale także stosunek czynników pozostających względem siebie pierwszymi.

Niech $a=n \cdot p$, $b=n \cdot q$ takie , że $a>b$ i $p>q$ oraz n jest największym wspólnym dzielnikiem, wtedy

$$s = a/b = (n \cdot p / (n \cdot q) = p/q$$

Liczba s określająca stosunek p i q pozwala szukać rozwiązania w liniowej ilości przedziałów o dokładnie sprecyzowanych krańcach. Po ustaleniu wartości p lub q określimy n, czyli największy wspólny dzielnik liczb a i b. Uzasadnienie liniowej ilości rozpatrywanych przedziałów można poprowadzić analogicznie jak zrobiono to wyżej dla problemu *mult(n)*.

Wniosek: Problem *nwd(a,b)* można alternatywnie do algorytmu Euklidesa rozwiązać równolegle przy pomocy liniowej ilości procesorów określonej w najgorszym przypadku przez wykładnik $2k$ określający górny zakres przedziału przynależności większej spośród liczb a i b (oczywiście o ile możliwym okaże się skonstruowanie algorytmu loga-rytmicznego).

Wychodząc z pokazanych związków problemu *nwd(a,b)* i *mult(n)*, przy próbie skonstruowania algorytmu logarytmicznego przyjmiemy, że parametrem umożliwiającym realizację iteracyjnego zawężania każdego z rozpatrywanych przedziałów będzie stosunek poszukiwanych czynników znanego iloczynu.

Przyjęcie takiego założenia implikuje wymóg własności proporcjonalności modelu obliczeniowego algorytmu loga-rytmicznego.

Wiemy, że jeśli dwie liczby całkowite dodatnie a i b zapisane binarnie mają odpowiednio długości w_1 i w_2, to długość liczby będącej iloczynem $p \cdot q$ zapisanej binarnie jest równa $w=w_1+w_2$ lub $w=w_1+w_2-1$.

Tak więc, jeśli dla $a=n \cdot p$ jej długość jest równa w i przyjmiemy, że długość n jest równa w_1, to liczba p może mieć długość $w_2=w-w_1$ lub $w_2=w-w_1+1$. Długość liczby p zapisanej binarnie pozwala określić przedział do którego ta liczba należy. Podobnie dla $b=n \cdot q$, jeśli długość b jest równa v i dalej zakładając, że długość n jest równa $v_1=w_1$, to liczba q może mieć długość $v_2=v-w_1$ lub $v_2=v-w_1+1$.

Jeśli zakładana liczba n o długości w_1 jest największym wspólnym dzielnikiem liczb a i b, to liczba $s=a/b=p/q$ jako iloraz dwóch liczb będących względem siebie pierwszymi musi zawierać się w jednym z czterech dobrze zdefiniowanych zbiorów S_1, S_2, S_3 *i* S_4. Każdy z tych zbiorów zawiera elementy określające wszystkie możliwe ilorazy liczb o długościach odpowiednio

$$\frac{w-w_1}{v-w_1}; \frac{w-w_1}{v-w_1+1}; \frac{w-w_1+1}{v-w_1} oraz \frac{w-w_1+1}{v-w_1+1}$$

Jeśli w żadnym z tak określonych zbiorów ilorazów nie występuje element o wartości s, to żadna liczba n o długości w_1 nie może być największym wspólnym dzielnikiem liczb a i b.

Ponieważ największy wspólny dzielnik dla danych dwóch liczb $a>b$ nie może być większy od mniejszej z nich, to ilość zbiorów, w których należy sprawdzić czy zawierają elementy określające wartość s, nie jest większa niż $4 \cdot v$ (v jest długością liczby b zapisaną binarnie).

W ten sposób konstruowany algorytm równoległy spełnia pierwszy warunek przynależności do klasy problemów NC – co najwyżej wielomianową liczbę procesorów.

5.3 Analiza modelu i kryterium zbieżności

Zasadniczym pytaniem jest, w jakim czasie możliwe jest rozstrzygnięcie, czy w rozpatrywanym zbiorze S istnieje element o wartości określonej przez znany iloraz $s=p/q$.

Przede wszystkim ustalimy zależności jakie zachodzą pomiędzy elementami tych zbiorów. Element zbioru odpowiadający ilorazowi liczb i/j będziemy oznaczać $s_{i,j}$. Zakładamy, że i oraz j należą do pewnych przedziałów wartości zdefiniowanych przez ich długości w zapisie binarnym. Dodatkowo przyjmujemy, że i jest większe od $j+1$.

Wtedy dla $i=const$ przy rosnącym j mamy do czynienia z ciągiem malejącym, w którym różnica sąsiednich wyrazów jest określona przez iloraz $i/(j \cdot (j+1))$, a różnica wyrazów $s_{i,j+k}-s_{i,j}=k \cdot i/(j \cdot (j+k))$.

Dla $j=const$ przy rosnącym i mamy do czynienia z rosnącym ciągiem arytmetycznym o różnicy równej odwrotności j tj. $s_{i+1,j}-s_{i,j}=1/j$.

Przy $i>j+1$ które założyliśmy zachodzi także $s_{i,j}>s_{i+1,j+1}$.

Ponieważ dla ustalonego $j=const$ mamy do czynienia z ciągiem arytmetycznym, średnia arytmetyczna wyrazów symetrycznie oddalonych od wyrazu $s_{i,j}$ jest równa temu wyrazowi. W szczególności $s_{i,j}=(s_{i+j,,j}+s_{i-j,,j})/2$, czyli średnia arytmetyczna ilorazów sumy $i+j$ oraz różnicy $i-j$ jest równa ilorazowi liczb i/j.

Ponadto, w oparciu o własność ciągu arytmetycznego, ilorazy $(i+j)/j$ oraz $(i-j)/j$ są równe odpowiednio $s_{i,j}+1$ oraz $s_{i,j}-1$.

Pokazane związki i relacje zachodzące pomiędzy elementami każdego rozpatrywanego zbioru *S* oraz własności ciągów jakie mogą być wyodrębnione, umożliwiają odwzorowanie tego zbioru w postaci tablicy o uporządkowanej strukturze.

Przyjmiemy, że w kolejnych kolumnach tablicy znajdują się ilorazy wszystkich liczb *i* z rozpatrywanego zakresu jej wartości przez ustaloną wartość *j*. Z kolei, w kolejnych wierszach tablicy znajdują się ilorazy wszystkich liczb *j* z rozpatrywanego zakresu jej wartości przez ustaloną wartość *i*.

Przy tym odwzorowaniu, istotne jest, że z każdą kolumną oraz z każdym wierszem związana jest konkretna wartość ilorazów z rozpatrywanych zakresów.

Możliwość odwzorowania zbioru *S* jako tablicy o opisanych własnościach, mocno sugeruje istnienie algorytmu logarytmicznego. Operacja wyszukiwania w konkretnym wierszu lub konkretnej kolumnie tablicy dowolnego elementu spełniającego relacje mniejszości, równości bądź większości w stosunku do znanego elementu, w przypadku uporządkowanej struktury (znane są różnice pomiędzy elementami, zarówno w układzie kolumnowym, jaki i wierszowym) wymaga znajomości punktu odniesienia, którym jest wartość elementu i jego usytuowanie w tablicy.

Operacja wyszukiwania zwraca:
- albo szukany element (spełniona relacja równości),
- albo gdy spełnione są relacje mniejszości i większości dwa sąsiednie elementy w wierszu (wyszukiwanie dla ustalonego *i*) lub kolumnie (wyszukiwanie dla ustalonego *j*),
- albo wynikiem jest stwierdzenie braku w przeszukiwanym wierszu lub kolumnie elementu spełniającego zadane kryterium.

Jeśli operacja wyszukiwania zwraca element, to znamy wiersz i kolumnę w której zwrócony element jest usytuowany, znamy tym samym liczby dające w ilorazie wartość tego elementu.

W obu pozostałych przypadkach, tj. gdy zwracana jest para sąsiednich elementów lub stwierdzany jest brak elementu, proces

wyszukiwania musi być kontynuowany w pewien ukierunkowany sposób.

Ponieważ wzajemnie pierwsze mogą być zarówno liczby nieparzyste, jak nieparzyste i parzyste (np. ilorazy 21/10 ale także 20/11), kryterium zawężania obszaru przeszukiwania może być odmienne dla każdego z trzech możliwych przypadków.

Spośród trzech możliwych przypadków, dwa charakteryzuje wspólna własność – dzielnikiem w ilorazach jest liczba nieparzysta, kiedy wzajemnie pierwsze mogą być liczby nieparzyste lub parzyste i nieparzyste. Wtedy w ciągu $s_{i,j}$ ($j=const$) dla danego $s=p/q$ istnieją dwie pary wyrazów takie, że dla $m>k$ $s_{k,l}<s-1<s_{k+2,l}$ oraz $s_{m,l}<s+1<s_{m+2,l}$, przy czym $s_{m,l}-s_{k,l}=2$ oraz $s_{m+2,l}-s_{k+2,l}=2$.

Jeśli $s_{k,l}$ jest pierwszym wyrazem ciągu arytmetycznego, a $s_{m+2,l}$ ostatnim, to w przedziale $<s_{k+2,l},s_{m,l}>$ jest nieparzysta ilość wyrazów tego ciągu. Wtedy albo istnieje wyraz o wartości $(s_{k,l}+s_{m+2,l})/2=(s_{k+2,l}+s_{m,l})/2$ taki, że jest równy s i jest równo oddalony od obu krańców przedziału, albo istnieje przedział określony przez dwa sąsiednie wyrazy $s_{r,l}$ oraz $s_{r+2,l}$ takie, że $s_{r,l}<s<s_{r+2,l}$, co oznacza wobec nieparzystej ilości wyrazów w rozpatrywanym zakresie, że wyraz $s_{r,l}$ jest oddalony od $s_{k,l}$ albo o nieparzystą, albo o parzystą ilość wyrazów i tym samym wyraz $s_{r+2,l}$ jest oddalony od $s_{m+2,l}$ albo o parzystą, albo nieparzystą ilość wyrazów, a stąd wynika, że te odległości w takim przypadku nie są sobie równe.

Ustalimy teraz charakter tych stwierdzonych nierówności. W tym celu rozpatrzymy otoczenie szukanego elementu. Zbadamy zachowanie się elementów ciągów $\{s_{i,q-2}\}$ i $\{s_{i,q+2}\}$ względem ciągu $\{s_{i,q}\}$, o którym zakładamy, że zawiera szukany element o wartości $s=p/q$.

W celu ułatwienia dyskusji, posługiwać się będziemy odwzorowaniem tych ciągów przy pomocy tablicy. Każdy z rozpatrywanych ciągów rozmieszczony jest w kolumnach, a usytuowanie elementów określać będziemy przy pomocy indeksów wierszy i kolumn.

Analizę rozpoczniemy od „lewej" strony, tj. ciągu rozmieszczonego w kolumnie $(q-2)$. W wierszu, w którym znajduje się element $s_{p,q}$, sąsiednim jest element $s_{p,q-2}>s_{p,q}$. Jeśli element ten będziemy traktować jako średnią arytmetyczną elementów s_{p+q-2} oraz s_{p-q+2} określających odpowiednio iloraz $(p+q-2)/(q-2)$ i iloraz $(p-q+2)/(q-2)$, to możemy z nim związać dwie pary elementów

spełniających nierówności $s_{p-q+1,q-2} < s_{p,q-2}-1 < s_{p-q+3,q-2}$ oraz $s_{p+q-3,q-2} < s_{p,q-2}+1 < s_{p+q-1,q-2}$ przy czym średnia arytmetyczna obu par elementów występujących w nierównościach jest równa odpowiednio $s_{p,q-2}-1$ oraz $s_{p,q-2}+1$. Element $s_{p,q-2}$ jest w rozpa-trywanej kolumnie poprzedzony elementami $s_{p-2,q-2}$; $s_{p-4,q-2}$, …. , które do pewnego momentu są większe od elementu $s_{p,q}$. Z każdym z tych elementów $s_{p-2,q-2}$; $s_{p-4,q-2}$, …., identycznie jak z elementem $s_{p,q-2}$ związane są dwie pary elementów.

W szczególności jeśli przejdziemy do elementu $s_{p-k,q-2}$ takiego, że jest większy od $s_{p,q}$, a poprzedzający go w kolumnie element $s_{p-k-2,q-2}$ jest mniejszy od $s_{p,q}$, to stosownie przemieszczą się pary elementów z nim związane i będą to para ($s_{p-q-k+1,q-2}$,$s_{p-q+k+3,q-2}$) oraz para ($s_{p+q-k-3,q-2}$,$s_{p+q-k-1,q-2}$). Znajomość elementów tych par pozwala określić odległość elementu $s_{p-q+3,q-2}$ od elementu $s_{p-k-2,q-2}$ z jednej strony i z drugiej strony odległość elementu $s_{p+q-k-3}$ od elementu $s_{p-k,q-2}$. Przyjmując, że miarą tych odległości może być różnica indeksów wierszy, w których są one usytuowane, otrzymujemy:

$$d_1=(p-k-2)-(p-q-k+3)=q-5 \quad \text{oraz} \quad d_2=(p+q-k-3)-(p-k)=q-3$$

co jednoznacznie pokazuje, że pierwsza z nich jest mniejsza od drugiej. Warto zauważyć, że tak określona miara odległości nie jest zależna od przesunięcia k.

Analizę „prawej" strony, tj. ciągu rozmieszczonego w kolumnie $(q+2)$ można przeprowadzić podobnie. W wierszu, w którym znajduje się element $s_{p,q}$, występuje element $s_{s,q+2}<s_{p,q}$. Jeśli element ten potraktujemy jako średnią arytmetyczną elementów $s_{p+q+2,q+2}$ oraz $s_{p-q-2,q+2}$ określających odpowiednio iloraz $(p+q+2)/(q+2)$ i iloraz $(p-q-2)/(q+2)$, to możemy z nim związać dwie pary elementów spełniających nierówności $s_{p+q+1,q+2}<s_{p,q}+1<s_{p+q+3,q+2}$ oraz $s_{p-q-3,q+2}<s_{p,q}-1<s_{p-q-1,q+2}$. Element $s_{p,q+2}$ poprzedza w rozpatrywanej kolumnie elementy $s_{p+2,q+2}$, $s_{p+4,q+2}$, … , które do pewnego momentu są mniejsze od elementu $s_{p,q}$. Z każdym z tych elementów $s_{p+2,q+2}$, $s_{p+4,q+2}$, …, identycznie jak z elementem $s_{p,q+2}$ związane są dwie pary elementów. W szczególności jeśli przejdziemy do elementu $s_{p+k,q+2}$ takiego, że jest mniejszy od $s_{p,q}$, a kolejny element $s_{p+k+2,q+2}$ jest większy od $s_{p,q}$, to stosownie przemieszczą się pary elementów z nim związane, tj. pary ($s_{p-q+k-3,q+2}$,$s_{p-q+k-1,q+2}$) oraz ($s_{p+q+k+1,q+2}$,$s_{p+q+k+3,q+2}$).

	$q-2$	q	$q+2$

	$s_{p-q-k+1,q-2} < s_{p,q} - 1$		
	$s_{p-q-k+3,q-2} > s_{p,q} - 1$		
			$s_{p-q-3,q+2}$
$p-q-1$		$s_{p-q-1,q} < s_{p,q} - 1$	$s_{p-q-1,q+2}$
$p-q+1$	$s_{p-q+1,q-2}$	$s_{p-q+1,q} > s_{p,q} - 1$	
$p-q+3$	$s_{p-q+3,q-2}$		
			$s_{p-q+k-3,q+2} < s_{p,q} - 1$
			$s_{p-q+k-1,q+2} > s_{p,q} - 1$
	$s_{p-q-k-2,q-2} < s_{p,q}$		
$p-k$	$s_{p-k,q-2} > s_{p,q}$		
p	$s_{p,q-2} > s_{p,q}$	$s_{p,q}$	$s_{p,q+2} < s_{p,q}$
$p+k$			$s_{p+k,q+2} < s_{p,q}$
			$s_{p+k+2,q+2} > s_{p,q}$
	$s_{p+q-k-3,q-2} < s_{p,q} + 1$		
	$s_{p+q-k-1,q-2} > s_{p,q} + 1$		
	$s_{p+q-3,q-2}$		
$p+q-1$	$s_{p+q-1,q-2}$	$s_{p+q-1,q} < s_{p,q} + 1$	
$p+q+1$		$s_{p+q+1,q} > s_{p,q} + 1$	$s_{p+q+1,q+2}$
$p+q+3$			$s_{p+q+3,q+2}$
			$s_{p+q+k+1,q+2} < s_{p,q} + 1$
			$s_{p+q+k+3,q+2} > s_{p,q} + 1$

Rys.1

Podobnie jak dla „lewej" strony, znajomość elementów tych par pozwala określić odległość elementu $s_{p-q+k-1,q+2}$ od elementu $s_{p+k,q+2}$ z

jednej strony i z drugiej strony odległość elementu $s_{p+q+k+1,q+2}$ od elementu $s_{p+k+2,q+2}$. W efekcie odległości mierzona różnicą indeksów wierszy, w których są one usytuowane, otrzymujemy:

$$d_1=(p+k)-(p-q+k-1)=q+1 \quad \text{oraz} \quad d_2=(p+q+k+1)-(p+k+2)=q-1$$

W odróżnieniu do „lewej" strony, pierwsza odległość jest większa od drugiej, pozostając nadal niezależną od przesunięcia k.

Uzyskany wynik dla „lewej" i „prawej" strony tablicy zawierającej analizowane ciągi ilorazów, określa kryterium sukcesywnego zawężania przeszukiwanego zakresu. Jeśli zastosować metodę połowienia, to proces wyszukiwania określonego ilorazu $s_{p,q}=p/q$ równego stosunkowi liczb wzajemnie pierwszych będzie realizowany w czasie logarytmicznym.

Przeprowadzona dyskusja dotyczy sytuacji, gdy iloraz s określają dwie liczby nieparzyste.

Przypadki, gdy iloraz s określany jest przez liczby parzystej i nieparzyste (np. 21/10 lub 20/11), wymagają podobnej analizy.

R o z d z i a ł s z ó s t y

Dyskusja problemu *mult(n)*

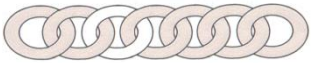
Marek Malinowski

Technologie Teleinformatyczne

Opracowanie zawiera opis koncepcji rozwiązania problemu **mult(n)** jako szczególnego przypadku problemu faktoryzacji. Przedstawiona została interpretacja geometryczna wprowadzonego modelu. Omówione zostały podstawowe własności modelu, w tym tzw. mechanizm bąbelkowy stanowiący główny element konstruowanych algorytmów. Analiza proponowanych algorytmów wskazuje, że charakteryzują się czasową złożonością logarytmiczną

Prezentowane wyniki upoważniają do stwierdzenia, że relacja zawierania klas $NC \subseteq P \subseteq NP$ przyjmuje postać równości tych klas. Pozwala to, o ile dla problemu faktoryzacji istnieje redukcja wielomianowa do problemu **3SAT,** zakwalifikować problem faktoryzacji do klasy problemów ***NP-zupełnych***.

6



byłoby dobrze wiedzieć, w którą stronę

otwierają się te bądź inne drzwi

[R. Ackoff], The Art of Problem Solving – tłum. wł.

W klasycznym ujęciu teorii liczb, problem faktoryzacji liczby oznacza jej rozkład na czynniki pierwsze. Znaczenie tego problemu nie wynika z faktu, że jest to jeden z najstarszych problemów dla którego do chwili obecnej nie jest znany ani nie udowodniono istnienia efektywnego algorytmu jego rozwiązania, ale przede wszystkim dlatego, że na założeniu jego trudnej rozwiązalności bazuje szereg rozwiązań współczesnej kryptografii cyfrowej, w tym szyfrowanie z kluczem publicznym.

Problem *mult(n)* może być traktowany jako szczególny przypadek problemu faktoryzacji.

6.1 Ogólny opis koncepcji

Istotę koncepcji stanowi skojarzenie ze sobą dwóch prostych modeli nie wymagających zaawansowanego aparatu matematycznego i odrzucenie założenia, że ponieważ szukane jest rozwiązanie całkowitoliczbowe, to wszelkie obliczenia prowadzone będą z wykorzystaniem arytmetyki całkowito-liczbowej.

Model pierwszy pozwala przedstawić iloczyn liczb pierwszych w postaci rodziny sum dwóch członów, z których jeden będziemy określać mianem multiplikatywnego, a drugi mianem addytywnego.

Drugi model stanowi hiperbola równoosiowa i przecinające ją proste. Ich wzajemne usytuowanie możemy opisywać przy pomocy dwóch komplementarnych funkcji kwadratowych, których miejsca zerowe wyzna-czane są przez punkty przecięcia prostych o współczynnikach *±1* z hiperbolą. Model ten będziemy nazywać modelem funkcji kwadratowej (**MFK**) ponieważ usytuowanie prostych względem hiperboli można opisać za pomocą funkcji kwadratowej.

Powiązanie ww. modeli ze sobą sprowadza się do odwzorowania członu multiplikatywnego przez hiperbolę , zaś członu addytywnego przez proste. Poprzez takie odwzorowanie, każdy element rodziny reprezentacji iloczynu dwóch liczb pierwszych ma jednoznaczny odpowiednik w modelu MFK. Jest oczywiste, że w sytuacji gdy zbiór takich odwzorowań jest duży, znalezienie elementu odwzorowania metodą prostego przeszukiwania może wymagać nieakceptowanego czasu. Używamy świadomie sformułowania **może**, bo w pewnych sytuacjach wynikających z opisanych dalej własności, rozwiązania uzyskujemy w czasie wymaganym dla obliczenia jednego tylko elementu zbioru odwzorowań (będzie on zdefiniowany jako bazowy).

Ponieważ każdą liczbę nieparzystą można przestawić jako $2 \cdot k+1$, gdzie $k \epsilon N$ i jeśli dwie liczby pierwsze $p=2 \cdot m+1$ oraz $q=2 \cdot k+1$ zapiszemy w takiej postaci, to ich iloczyn $n=p \cdot q$ przyjmie postać $n=4 \cdot m \cdot k+2 \cdot (m+k)+1$.

Jeśli wykonamy elementarne przekształcenia tak przedstawionego iloczynu dwóch liczb pierwszych, to uzyskamy postulowaną w pierwszym modelu postać uwidaczniająca człon multiplikatywny $m \cdot k$ oraz addytywny $(m+k)$

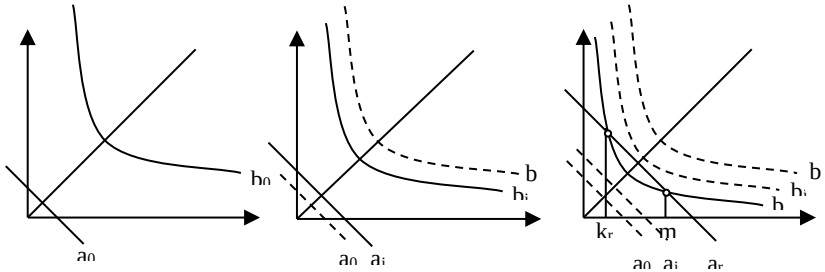
$$\frac{n-1}{2} = 2 \cdot m \cdot k + (m+k) = 2 \cdot b + a$$

Prawa strona reprezentuje całą rodzinę sum członów multiplikatywnych b_i i członów a_i addytywnych, ponieważ zmniejszenie członu multiplikatywnego dla zachowania relacji równości z lewą stroną zwiększa odpowiednio człon addytywny. Tak więc możemy zapisać następująco:

$$\frac{n-1}{2} = 2 \cdot b_i + a_i \quad \text{gdzie: } b_{i+1} = b_i - 1 \quad a_{i+1} = a_i + 2$$

$$\text{oraz } b_0 = \frac{n-1}{2} \operatorname{div} 2 \quad a_0 = \frac{n-1}{2} \bmod 2$$

Ciągi *{aᵢ}*, *{bᵢ}* tworzą ciągi arytmetyczne, a odpowiednie wyrazy $a_i=m_i+k_i$ oraz $b_i=m_i \cdot k_i$ mogą być interpretowane jako prosta i gałąź hiperboli równoosiowej. Wtedy w układzie *O(mk)* punkty przecięcia prostej z hiperbolą są rozwiązaniem układu równań $m+k=a_i$ oraz $m \cdot k=b_i$,które sprowadza się do równania kwadratowego $k^2-a_i \cdot k+b_i=0$.



W tym sensie zbiór par *{aᵢ,bᵢ}* reprezentuje rodzinę równań kwadratowych, w której zawiera się szukane rozwiązanie m_r i k_r określające liczby $p=2 \cdot m_r+1$ i $q=2 \cdot k_r+1$ będące czynnikami iloczynu $n=p \cdot q$. Możemy więc określić warunek istnienia rozwiązań dla tak zdefiniowanej rodziny równań. Pamiętając, że *{aᵢ}* oraz *{bᵢ}* tworzą ciągi arytmetyczne warunek ten przyjmie postać $\Delta_i=(a_0+2 \cdot i)^2-4 \cdot (b_0-i) \geq 0$, a po uporządkowaniu wyrazów lewej strony uzyskamy $4 \cdot i^2+4 \cdot (a_0+1) \cdot i+(a_0^2-4 \cdot b_0) \geq 0$.

Tak więc w oparciu o tą nierówność możemy ustalić wartości $i \geq \sqrt{4 \cdot b_0 + 2 \cdot a_0 + 1} - (a_0 + 1)$, dla której zwracane są rozwiązania m_i oraz k_i rozpatrywanej rodziny równań

$$m_i = \frac{a_i + \sqrt{\Delta_i}}{2} = \frac{a_i + \sqrt{{a_i}^2 - 4 \cdot b_i}}{2} = \frac{(a_0 + 2 \cdot i) + \sqrt{({a_0}^2 + 2 \cdot i)^2 - 4 \cdot (b_0 - i)}}{2}$$

$$k_i = \frac{a_i - \sqrt{\Delta_i}}{2} = \frac{a_i - \sqrt{{a_i}^2 - 4 \cdot b_i}}{2} = \frac{(a_0 + 2 \cdot i) - \sqrt{({a_0}^2 + 2 \cdot i)^2 - 4 \cdot (b_0 - i)}}{2}$$

Ponieważ interesują nas rozwiązania całkowite, najmniejszą wartość całkowitą i większą od określonej przez powyższą nierówność nazywać będziemy bazową i oznaczać będziemy przez i_b. Stosownie, wyznaczane przez nią i związane z nią wartości oznaczać będziemy m_b, k_b, $a_b=a_0+2\cdot i_b$, $b_b=b_0-i_b$ i $\Delta_b=(a_0+2\cdot i_b)^2-4\cdot(b_0-i_b)$. Wszystkie te wielkości w modelu **MFK** charakteryzują tzw. punkt bazowy.

6.2 Model MFK i jego własności

Model **MFK** skonstruowany w oparciu o opisaną w poprzednim punkcie koncepcję charakteryzuje szereg własności. Zaprezentowane zostaną tylko te, które odnoszą się do wprost do odwzorowań za pomocą rodziny równań $k^2-a_i\cdot k+b_i=0$. Nie będzie dyskutowane komplementarne odwzorowanie za pomocą prostej $c_i=m_i-k_i$ i dwóch gałęzi hiperboli $d_i=m_i\cdot k_i$ prowadzące do rodziny równań $k^2+c_i\cdot k-d_i=0$. Nie będą dyskutowane alternatywne reprezentacje liczb w postaci $(2\cdot k-1)$ i możliwa reprezentacja liczb pierwszych $6\cdot k\pm1$ lub $6\cdot k\pm5$ oraz ich kombinacje, pomimo iż te ostatnia reprezentacje stwarzają możliwości istotnego przyspieszenia uzyskania rozwiązań.

Podstawowe własności modelu można określić na podstawie analizy postaci rozwiązań dla m_i oraz k_i rozpatrywanej rodziny równań $k^2-a_i\cdot k+b_i=0$.

Przede wszystkim możemy stwierdzić, że ciąg rozwiązań $\{m_i\}$ jest ciągiem rosnącym (tj. $m_{i+1}>m_i$), zaś ciąg rozwiązań $\{k_i\}$ jest ciągiem malejącym (tj. $k_{i+1}>k_i$).

Ponadto analiza wyrażeń dla m_i i k_i jednoznacznie pokazuje, że ich różnica m_i-k_i określana jest przez pierwiastek kwadratowy z wyróżnika Δ_i.

Na tej podstawie można sformułować wniosek, że ponieważ dla i_b wyróżnik jest bliski zeru, to dla przypadku dwóch czynników iloczynu takich, że ich różnica jest mała, rozwiązanie uzyskujemy w punkcie bazowym (tj. dla i_b) lub nieznacznie tylko od niego odległego (tj. dla $i=i_b+j$, gdy j jest małe). W tej sytuacji interesujące może być, jak duża może być ta różnica, by rozwiązanie uzyskać w punkcie bazowym określonym przez i_b lub najbliższym i_b+1.

Jeśli $p=2\cdot m+1$ i $q=2\cdot k+1$, to ich różnica $\gamma = p - q = 2\cdot(m-k)$. Chcemy, aby rozwiązanie było nie dalej niż w punkcie i_b+2, więc możemy szacować, że różnica $\gamma=2\cdot(m_i-k_i)$ powinna być mniejsza od tej, którą określa pierwiastek z Δ_{ib+2}.

Ponieważ $\Delta_{ib+2}=(a_0+2\cdot(i_b+2))^2-4\cdot(b_0-(i_b+2))$, to po przekształceniach i podstawieniu wyznaczonego wcześniej wyrażenia określającego i_b otrzymamy wyrażenie szacujące wielkość γ w postaci

$$\gamma \le 2\cdot\sqrt{2\cdot\sqrt{4\cdot b_0+1}+4} \approx 4\cdot\sqrt{2\cdot\sqrt{n}}$$

Rezultaty obliczeń w modelu **MFK** dla trzech przykładów ilustrujących omówione wyżej sytuacje, tj. wartość obu czynników porównywalna $p \approx q$, różnica $p-q$ nie większa niż szacowana wielkość γ oraz różnica $p-q$ znacznie większa od γ, zawarto w formie tabelarycznych ilustracji w kolejnych przykładach.

Przykład 1: Dany jest iloczyn $n= 98722727$ szukanych dwóch liczb pierwszych (w przykładzie są to liczby $p=9973$ oraz $q=9899$). W tym przypadku $a_0=1$, $b_0=24680681$, a wyznaczony przez $i_b=4968$ punkt bazowy modelu **MFK** zwraca rozwiązanie $m=4986$ oraz $k=4949$. Poniżej w formie zestawienia tabelarycznego ilustracja obliczeń i rezultaty zwracanych wyników.

i	a_i	b_i	Δ_i	$\sqrt{\Delta_i} = m_i - k_i$	m_i	k_i
0	1	24680681	-98722723	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮
4967	9933	24675715	-38371	-	-	-
4968	**9935**	**24675714**	**1369**	**37**	**4986**	**4949**
4969	9937	24675713	41117	202,773272	5069,887	4867,113
4970	9939	24675712	80873	284,381786	5111,691	4827,309
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Przykład 2: Dany jest iloczyn $n= 94354553$ szukanych dwóch liczb pierwszych (w przykładzie są to liczby $p=9973$ oraz $q=9461$, których różnica spełnia oszacowanie γ). W tym przypadku $a_0=0$, $b_0=23588638$, a rozwiązanie $m=4986$ oraz $k=4730$ zwraca kolejny za punktem bazowym $i_b=4857$ punkt dla $i=4858$. Poniżej w formie

zestawienia tabelarycznego ilustracja obliczeń i rezultaty zwracanych wyników.

i	a_i	b_i	Δ_i	$\sqrt{\Delta_i}$	m_i	k_i
0	0	23588638	-94354552	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮
4856	9712	23583782	-12184	-	-	-
4857	9714	23583781	26672	163,315645	4938,658	4775,342
4858	**9716**	**23583780**	**65536**	**256**	**4986**	**4730**
4859	9718	23583779	104408	323,122268	5020,561	4697,439
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Przykład 3: Dany jest iloczyn *n= 32920873* szukanych dwóch liczb pierwszych (w przykładzie są to liczby *p=9973* oraz *q=3301*, których różnica znacznie przewyższa oszacowanie γ). W tym przypadku $a_0=0$, $b_0=8230218$, a rozwiązanie *m=4986* oraz *k=1650* zwracane jest dla *i=3318* istotnie oddalonego od punktu bazowego $i_b=2869$. Poniżej w formie zestawienia tabelarycznego ilustracja obliczeń i rezultaty zwracanych wyników.

i	a_i	b_i	Δ_i	$\sqrt{\Delta_i}$	m_i	k_i
0	0	8230218	-32920872	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮
2868	5736	8227350	-7704	-	-	-
2869	5738	8227349	15248	123,482792	2930,741	2807,259
2870	5740	8227348	38208	195,468668	2967,734	2772,266
⋮	⋮	⋮	⋮	⋮	⋮	⋮
3317	6634	8226901	11102352	3332,01921	4983,01	1650,99
3318	**6636**	**8226900**	**11128896**	**3336**	**4986**	**1650**
3319	6638	8226899	11155448	3339,97725	4988,989	1649,011
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Poza parametrem γ szacującym różnicę szukanych czynników iloczynu gwarantującą zwracanie wyniku w pobliżu punktu

bazowego, wprowadzimy jeszcze jeden parametr. Parametr ten, a w zasadzie związane ze sobą trzy parametry, będą określały ilorazy m_i/k_i, $(m_i-k_i)/k_i$ oraz $(m_i+k_i)/k_i$.

W każdym z rozpatrywanych przykładów, szukane liczby p i q pozostawały ze sobą w określonym stosunku $s=p/q$, który jest mniejszy od ilorazu m/k. W modelu MFK w punkcie bazowym stosunek zwracanych wartości m_b i k_b jest bliski wartości 1 i jeśli nie określa rozwiązania, to dla kolejnych i rośnie aż do wartości odpowiadającej $s=p/q$. Tak więc, jeśli iloraz liczb p i q jest duży, to punkt rozwiązania w modelu MFK jest znacznie oddalony od punktu bazowego.

Zanim sformułujemy wnioski wynikające z tego spostrze-żenia, rozpatrzymy zachowanie się modelu MFK w przypadku iloczynu n określonego przez czynniki, z których jeden jest liczbą pierwszą a drugi liczbą złożoną.

Przykład 4: Dany jest iloczyn $n= 98762619$. W przykładzie jest to iloczyn liczby pierwszej $p=9973$ oraz liczby złożonej $q=9903=3\cdot3301$ (różnica p i q jest mała – rozwiązanie więc powinno być zwrócone w punkcie bazowym i_b). W tym przypadku $a_0=1$, $b_0=24690654$, a rozwiązanie w punkcie bazowym $m=4986$ określa liczbę p natomiast $k=4951$ określa liczbę złożona q. Przeglądając rozwiązania dla kolejnych wartości i stwierdzamy, że dla $i=8304$ zwracane jest rozwiązanie $m=14959$ określające iloczyn $3\cdot p$ oraz $k=1650$ określające większy z czynników liczby złożonej q. Poniżej w formie zestawienia tabelarycznego ilustracja obliczeń i rezultaty zwracanych wyników.

i	a_i	b_i	Δ_i	$\sqrt{\Delta_i}$	m_i	k_i
0	1	24690654	-98762615	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮
4967	9935	24685687	-38523	-	-	-
4968	**9937**	**24685686**	**1225**	**35**	**4986**	**4951**
4969	9939	24685685	40981	202,4376	5070,719	4868,281
⋮	⋮	⋮	⋮	⋮	⋮	⋮
8303	16607	24682351	177063045	13306,503	14956,75	1650,248
8304	**16609**	**24682350**	**177129481**	**13309**	**14959**	**1650**
8305	16611	24682349	177195925	13311,496	14961,25	1649,752

Podstawowy wniosek wynikający z analizy ostatniego przykładu sprowadza się do stwierdzenia, że model **MFK** jest przydatny nie tylko przy rozwiązywaniu problemu *mult(n)* jako szczególnego przypadku faktoryzacji – jest modelem ogólnego problemu rozkładu liczb na czynniki pierwsze.

Jest tak, ponieważ cała przestrzeń rozwiązań zawiera w sobie wszystkie możliwe kombinacje iloczynów czynników pierwszych zadanej liczby n i zwraca je w sposób uporządkowany. Niech podobnie jak w przykładzie 4 liczba będzie określona przez trzy czynniki $n=p_1 \cdot p_2 \cdot p_3$ dla $p_1>p_2>p_3$, wtedy w modelu **MFK** zwracane będą rozwiązania dla (p_2p_3) i p_1, (p_1p_3) i p_2 oraz dla (p_1p_2) i p_3, ponieważ każdy z iloczynów (p_2p_3), (p_1p_3) i (p_1p_2) ma swoją reprezentację ogólnej postaci $2t+1$ (wartości t zawierają się w tym przypadku w ciągu $\{m_i\}$ modelu **MFK**).

W ogólnym przypadku dla $n=p_1p_2\ldots p_l$, model MFK zawiera rozwiązania określające wszystkie możliwe kombinacje iloczynów po $2, 3$ do $(l-1)$ czynników.

W tym miejscu wrócimy do wcześniej sygnalizowanego spostrzeżenia dotyczącego charakteryzowania oddalenia rozwiązania od punktu bazowego przy pomocy parametru s określającego iloraz szukanych dwóch czynników iloczynu n. Jest oczywiste, że jeśli w przypadku problemu *mult(n)* udałoby się odgadnąć iloraz $s=p/q$, to wymnażając n przez liczbę p_1 bliską s uzyskalibyśmy dla liczby $n_1=n \cdot p_1$ „efekt bąbelkowy" polegający na znacznym, jeśli nie zupełnym, usytuowaniu rozwiązania w pobliżu punktu bazowego modelu **MFK**. W istocie postąpiliśmy tak w przykładzie 4, jeśli porównać go z przykładem 3.

Niestety, naturalny iloraz szukanych dwóch czynników liczby $n=p \cdot q$ z reguły nie jest liczbą całkowitą, a przypadkowe błądzenie w zgadywaniu tego naturalnego ilorazu nie gwarantuje powodzenia.

Problem ilorazu określanego przez liczbę niecałkowitą, można rozwiązać poprzez dobór liczby n_1 będącej iloczynem pary niekoniecznie liczb pierwszych p_1 i q_1, których iloraz dobrze przybliża naturalny iloraz liczb p i q. Wtedy w modelu **MFK** oprócz rozwiązań dla p i q, które będą zwracane wprost, będziemy dysponowali w ciągach $\{m_i\}$ i $\{k_i\}$ rozwiązaniami określającymi

iloczyny $(p \cdot q_1)$ oraz $(q \cdot p_1)$, a znając te rozwiązania i znając liczby p_1 oraz q_1 łatwo ustalimy wartość szukanych czynników p i q. Przedstawiony sposób postępowania zilustrujemy poniższym przykładem.

Przykład 5. Szukamy czynników liczby $n=7289$. Jest to iloczyn liczb $p=197$ i $q=37$, których iloraz jest równy $s \approx 5{,}32$. W oparciu o model **MFK** uzyskujemy rozwiązanie oddalone od punktu bazowego.

i	a_i	b_i	Δ_i	$\sqrt{\Delta_i}$	m_i	k_i
1	0	1822	-7288	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮
42	84	1780	-64	-	-	-
43	86	1779	280	16,7332005	51,3666	34,6334
44	88	1778	632	25,1396102	56,56981	31,43019
⋮	⋮	⋮	⋮	⋮	⋮	⋮
57	114	1765	5936	77,0454411	95,52272	18,47728
58	**116**	**1764**	**6400**	**80**	**98**	**18**
59	118	1763	6872	82,8975271	100,4488	17,55124
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Wybierzemy liczby $p_1=17$ i $q_1=3$, których iloraz jest równy $\approx 5{,}66$, a iloczyn jest równy $n_1=51$. Wtedy w oparciu o model **MFK** szukamy rozwiązania dla liczby określonej przez iloczyn liczb $n \cdot n_1=371739$ i uzyskujemy w punkcie bazowym rozwiązanie $m_b=314$ określające iloczyn $p_1 \cdot q=17 \cdot 37=629$ oraz $k_b=295$ określające iloczyn $p \cdot q_1=197 \cdot 3=591$. Zgodnie z oczekiwaniami w modelu **MFK** zwracane są także inne rozwiązania, np. dla $i=520$ zwracane jest rozwiązanie $m_i=943$ określające iloczyn $q \cdot p_1 \cdot q_1=37 \cdot 17 \cdot 3$ oraz $k_i=98$ określające $p=197$.

i	a_i	b_i	Δ_i	$\sqrt{\Delta_i}$	m_i	k_i
1	1	92934	-371735	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮
303	607	92631	-2075	-	-	-
304	**609**	**92630**	**361**	**19**	**314**	**295**
305	611	92629	2805	52,9622507	331,9811	279,0189

⋮	⋮	⋮	⋮	⋮	⋮	⋮
519	1039	92415	709861	842,532492	940,7662	98,23375
520	**1041**	**92414**	**714025**	**845**	**943**	**98**
521	1043	92413	718197	847,465044	945,2325	97,76748
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Podobny „efekt bąbelkowy" uzyskamy także wtedy, gdy para liczb p_1 i q_1 nie będzie dokładnie określać naturalny iloraz liczb p i q. Na przykład dla $p_1=13$ i $q_1=3$, których iloraz znacznie różni się od ilorazu p/q, w modelu **MFK** rozwiązanie otrzymamy w bezpośrednim sąsiedztwie punktu bazowego.

i	a_i	b_i	Δ_i	$\sqrt{\Delta_i}$	m_i	k_i
1	1	71067	-284267	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮
265	531	70802	-1247	-	-	-
266	533	70801	885	29,7489496	281,3745	251,6255
277	**535**	**70800**	**3025**	**55**	**295**	**240**
278	537	70799	5173	71,9235705	304,4618	232,5382
⋮	⋮	⋮	⋮	⋮	⋮	⋮

W tym przypadku swoje oddziaływanie zaznaczył parametr γ szacujący maksymalną różnicę dwóch iloczynów gwarantująca rozwiązanie w bliskim sąsiedztwie punktu bazowego. Tutaj ma to miejsce dla iloczynów $p \cdot q_1=197 \cdot 3=591$ oraz $p_1 \cdot q=37 \cdot 13=481$, których różnica $p \cdot q_1 - p_1 \cdot q=110$ i jest bliska wartości szacowania $\gamma \approx 4 \cdot \sqrt{2 \cdot \sqrt{n}} = 4 \cdot \sqrt{2 \cdot \sqrt{p \cdot q \cdot p_1 \cdot q_1}} \approx 130$.

Analiza ostatniego z rozpatrywanych przykładów pozwala sformułować pewne sugestie co do sposobu prowadzenia poszukiwania rozwiązań z wykorzystaniem „efektu bąbelkowego". Bardziej szczegółowo zostaną one omówione w kolejnym punkcie.

Przedstawione w tym punkcie własności modelu MFK ograniczono do tych, które zdaniem autora mogą być podstawą konstruowania efektywnego algorytmu dla rozpatrywanego problemu *mult(n)*. Nie wyklucza to wykorzystania w tym celu innych, nie omawianych szczegółowo w tym opracowaniu własności, takich jak np. własności wynikające ze związków

między rodzinami zasygnalizowanych komplementarnych równań $k^2-a_i \cdot k+b_i=0$ i $k^2+c_i \cdot k+d_i=0$.

6.3 Konstrukcja algorytmu i analiza złożoności czasowej

Własności modelu MFK przedstawione w poprzednim punkcie mogą posłużyć do konstruowania algorytmów na kilka różnych sposobów.

Jeden z nich może bazować na wykorzystaniu „efektu bąbelkowego" do zbudowania ciągu ilorazów prowadzących do rozwiązania w punkcie bazowym. Drugi z prezentowanych sposobów polega na iteracyjnym dopasowywaniu stron wyrażenia opisującego jeden z możliwych szczególnych punktów rozwiązania w modelu MFK.

W obu przedstawianych algorytmach ma miejsce „pozorne skomplikowanie" problemu pierwotnego. Polega ono na wymnażaniu znanego iloczynu n szukanych liczb p i q przez iloczyn n_1 ustalanych dwóch liczb p_1 i q_1.

Zauważmy, że gdyby p_1 i q_1 nie były znane, mielibyśmy do czynienia z problemem faktoryzacji iloczynu czterech czynników $n \cdot n_1=p \cdot q \cdot p_1 \cdot q_1$, ale p_1 i q_1 są znane i w tym zawiera się istota „pozornego skomplikowania".

6.3.1 Algorytm „bąbelkowy" w modelu MFK

Konstrukcja pierwszego sposobu zilustrowana będzie przykładem poszukiwania czynników iloczynu $n=7289$. Szukanymi czynnikami tego iloczynu są liczby $p=197$ i $q=37$. Przy realizacji algorytmu wykorzystywane będą schematy obliczeniowe modelu **MFK** - **MFK1** dla danego iloczynu n, **MFK2** dla ustalania liczb p_1, q_1 i ilorazów na kolejnych etapach algorytmu i **MFK3** dla weryfikowania punktu bazowego iloczynu $n \cdot p_1 \cdot q_1$. Rezultaty poszczególnych faz algorytmu ilustrujemy zestawieniem tabelarycznym.

Krok pierwszy polega na ustaleniu na podstawie punktu bazowego w **MFK1** wartości początkowych p_1 i q_1 (w przykładzie są to $p_1=53$ i $q_1=33$). Jako p_1 i q_1 ustalamy nieparzyste liczby

całkowite odpowiednio: najmniejszą liczbę większą od m_b oraz największą liczbę mniejszą od k_b.

Po ustaleniu p_1 i q_1 weryfikujemy otoczenie punktu bazowego **MFK3** dla $n \cdot p_1 \cdot q_1$ w celu stwierdzenia istnienia rozwiązania, w którym m_b i k_b (lub ich najbliższe otoczenie m_{b+j}, k_{b+j} dla małych j) określają iloczyny $q \cdot p_1$ lub $p \cdot q_1$ oraz $p \cdot q_1$ lub $p_1 \cdot q$ odpowiednio. Jeśli rozwiązanie nie istnieje, kontynuujemy procedurę następująco:

- w oparciu o **MFK2** ustalamy wartość ilorazu większego od ilorazu określającego ustalone poprzednio liczby p_1 i q_1 (w przykładzie w pierwszym cyklu będzie to wartość $s=2,21$),

- na podstawie tej wartości w **MFK1** szukamy takich m_i i k_i , których iloraz jest większy od przyjętej wartości s (w przykładzie będzie to $s=2,34$ określone przez $m_i=64,44$ i $k_i=27,56$),

- w oparciu o te wartości ustalamy nowe p_1 i q_1 do uwzględnienia w procesie weryfikacji punktu bazowego w **MFK3** (w przykładzie na tym etapie $p_1=65$ i $q_1=27$).

MFK1						MFK2					
a_i	b_i	m_i-k_i	m_i	k_i	s_i	a_i	b_i	m_i-k_i	m_i	k_i	s_i
84	1780	0	-	-	-	40	417	0	0	0	
a_b=86	1779	16,73	51,37	34,63	1,48	a_b=42	416	10	26	16	1,63
88	1778	25,14	56,57	31,43	1,8	44	415	16,61	30,31	13,7	2,21
90	1777	31,5	60,75	29,25	2,08	46	414	21,45	33,72	12,3	2,75
92	1776	36,88	64,44	27,56	2,34	48	413	25,53	36,77	11,2	3,27
94	1775	41,67	67,83	26,17	2,59	50	412	29,19	39,59	10,4	3,81
96	1774	46,04	71,02	24,98	2,84	52	411	32,56	42,28	9,72	4,35
98	1773	50,12	74,06	23,94	3,09	54	410	35,72	44,86	9,14	4,91
100	1772	53,96	76,98	23,02	3,34	56	409	38,73	47,36	8,64	5,49
102	1771	57,62	79,81	22,19	3,6	58	408	41,62	49,81	8,19	6,08
104	1770	61,12	82,56	21,44	3,85	60	407	44,41	52,2	7,8	6,7
106	1769	64,5	85,25	20,75	4,11	62	406	47,12	54,56	7,44	7,33
108	1768	67,76	87,88	20,12	4,37						
110	1767	70,94	90,47	19,53	4,63						
112	1766	74,03	93,01	18,99	4,9						
114	1765	77,05	95,52	18,48	5,17						
116	1764	80	98	18	5,44						
118	1763	82,9	100,4	17,55	5,72						

Opisane powyżej trzy kroki będą powtarzane są aż do momentu pozytywnej weryfikacji punktu bazowego w **MFK3**. Na każdym z cyklicznie wykonywanych etapów ustalania nowych wartości p_1 i q_1, stosunek ilorazów s z **MFK1** i **MKF2** jest w przybliżeniu stały i odpowiada wielkości wynikającej z

oszacowania γ gwarantującej przy poprawnym doborze p_1 i q_1 uzyskanie rozwiązania w punkcie bazowym.

MFK3						
cykl startowy	a_i	b_i	m_i-k_i	m_i	k_i	m_i/k_i
	3568	3185331	-	-	-	-
p_1=53 q_1=33	a_b=3570	3185330	59,8331012	1814,917	1755,083	1,03409
	3572	3185329	133,671238	1852,836	1719,164	1,07775
n_1=53·33=1749	...	...	...	...	...	...
	4228	3185001	2266,27006	3247,135	980,865	3,31048
$s=p_1/q_1$=	**a_r=4230**	**3185000**	**2270**	**3250**	**980**	**3,31633**
	4232	3184999	2273,72558	3252,863	979,1372	3,32217
	...	...	...	...	...	...
cykl 2	a_i	b_i	m_i-k_i	m_i	k_i	m_i/k_i
	3575	3196261	-	-	-	-
p_1=65 q_1=27	a_b=3577	3196260	99,4434513	1838,222	1738,778	1,05719
	3579	3196259	155,579562	1867,29	1711,71	1,09089
n_1=65·27=1755	...	...	...	...	...	...
	3859	3196119	1451,69039	2655,345	1203,655	2,20607
$s=p_1/q_1$=	**a_r=3861**	**3196118**	**1457**	**2659**	**1202**	**2,21215**
	3863	3196117	1462,29306	2662,647	1200,353	2,21822
	...	...	...	...	...	...
cykl 3	a_i	b_i	m_i-k_i	m_i	k_i	m_i/k_i
	3497	3057809	-	-	-	-
p_1=73 q_1=23	a_b=3499	3057808	108,485022	1803,743	1695,257	1,06399
	3501	3057807	160,539715	1830,77	1670,23	1,09612
n_1=73·23=1679	...	...	...	...	...	...
	3613	3057751	907,063945	2260,032	1352,968	1,67043
$s=p_1/q_1$=	**a_r=3615**	**3057750**	**915**	**2265**	**1350**	**1,67778**
	3617	3057749	922,872147	2269,936	1347,064	1,6851
	...	...	...	...	...	...
cykl 4	a_i	b_i	m_i-k_i	m_i	k_i	m_i/k_i
	3591	3225409	-	-	-	-
p_1=77 q_1=23	a_b=3593	3225408	89,5377016	1841,269	1751,731	1,05111
	3595	3225407	149,656273	1872,328	1722,672	1,08687
n_1=77·23=1771	...	...	...	...	...	...
	3687	3225361	832,18087	2259,59	1427,41	1,583
$s=p_1/q_1$=	**a_r=3689**	**3225360**	**841**	**2265**	**1424**	**1,59059**
	3691	3225359	849,732311	2270,366	1420,634	1,59814
	...	...	...	...	...	...
cykl 5	a_i	b_i	m_i-k_i	m_i	k_i	m_i/k_i
	3470	3010444	-	-	-	-
p_1=87 q_1=19	a_b=3472	3010443	114,070154	1793,035	1678,965	1,06794
	3474	3010442	164,036581	1819,018	1654,982	1,09912
n_1=87·19=1653	...	...	...	...	...	...
	3478	3010440	233,931614	1855,966	1622,034	1,14422
$s=p_1/q_1$=	**a_r=3480**	**3010439**	**262**	**1871**	**1609**	**1,16283**
	3482	3010438	287,353441	1884,677	1597,323	1,1799
	...	...	...	...	...	...
cykl 6	a_i	b_i	m_i-k_i	m_i	k_i	m_i/k_i
	3548	3148896	-	-	-	-
p_1=91 q_1=19	a_b=3550	3148895	83,1865374	1816,593	1733,407	1,04799
	3552	3148894	145,354738	1848,677	1703,323	1,08534

$n_1=91\cdot19=1729$	$\mathbf{a_r=3554}$	**3148893**	**188**	**1871**	**1683**	**1,11171**
$s=p_1/q_1=$	3556	3148892	222,638721	1889,319	1666,681	1,13358
	...	...	...	...	...	...

To oszacowanie w rozpatrywanym przykładzie jest równe $\gamma\approx340$ (pamiętamy, ze γ określa różnicę p-q, a my posługujemy się różnicą m-$k=\gamma/2$), co odpowiada dla **MFK3** ilorazowi m/k w punkcie bazy $s\approx1,07$.

W przytoczonym przykładzie w wyniku wykonania kolejnych cykli otrzymamy ciągi ilorazów dla **MFK1** *s=1,48, 2,34, 2,84, 3,34, 4,11, 4,63……* oraz dla **MFK2** *s=1,63, 2,21, 2,75, 3,27, 3,81, 4,35, …..* odpowiednio. W zestawieniu dla MK3 widoczny jest efekt zbliżania szukanego rozwiązania do punktu bazowego. Rozwiązania zawarte w ciągach $\{m_i\}$ i $\{k_i\}$ kolejnych cykli, określające odpowiednio $p\cdot q_1$ oraz $q\cdot p_1$ uzyskujemy dla punktów oznaczonych przez a_r. Przykładowo, w trzecim cyklu dla $p_1=73$ i $q_1=23$ schemat MFK3 zawiera dla $a_r=3615$ wyraz ciągu $m=2265$, który określa iloczyn $p\cdot q_1=(2\cdot m+1)$, a stąd liczbę p, bo $(2\cdot2265+1)/23=197$.

Przytoczony algorytm wydaje się jednak mało efektywny, a analiza jego złożoności obliczeniowej jest kłopotliwa. Na takie stwierdzenie pozwala analiza wyrazów ciągu s dla MKF2. Zasadniczo ciąg s w MFK2 tworzą kolejne ilorazy uzyskiwane dla kolejnych punktów rodziny rozwiązań schematu MFK2. Zauważamy także, że dwukrotnie użyliśmy w dwóch sąsiednich cyklach jako q_1 tej samej wartości. Ponadto, nie uwzględnialiśmy, że szukane p i q są liczbami pierwszymi.

Oceniając dyskutowany w tym miejscu algorytm, poprzestaniemy na stwierdzeniu, że proces iteracyjny zbiega jednostronnie do punktu rozwiązania, przy czym proces poszukiwania rozwiązania może być realizowany równolegle w ustalonych przedziałach wartości możliwych ilorazów s.

6.3.2 Algorytm „wagowy" w modelu MFK

Podobnie jak w przypadku algorytmu „bąbelkowego", w konstrukcji algorytmu „wagowego" w modelu MFK wykorzystana zostanie jego własność zawierania w rodzinie rozwiązań oprócz czynników pierwszych danego iloczynu n, także

rozwiązań dla wszystkich możliwych kombinacji iloczynów tych czynników, gdy jest ich więcej niż dwa.

Rozważając model MFK dla problemu $n= p_1 \cdot p_2 \cdot p_3$, przy założeniu $p_1>p_2>p_3$, problem iloczynu trzech liczb można sprowadzić do rozpatrzenia 3 przypadków iloczynu liczb złożonych $g_3=p_1 \cdot p_2$, $g_2=p_1 \cdot p_3$ oraz $g_1=p_2 \cdot p_3$ przez p_3, p_2 i p_1 odpowiednio.

$$n = p_1 \cdot p_2 \cdot p_3 = (p_2 \cdot p_3) \cdot p_1 = g_1 \cdot p_1$$

$$n = p_1 \cdot p_2 \cdot p_3 = (p_1 \cdot p_3) \cdot p_2 = g_2 \cdot p_2$$

$$n = p_1 \cdot p_2 \cdot p_3 = (p_1 \cdot p_2) \cdot p_3 = g_3 \cdot p_3$$

Każdy z iloczynów g_1, g_2, g_3 jest liczbą nieparzystą, więc daje się wyrazić jako:

$$g_1 = 2 \cdot t_{r1} + 1 \qquad g_2 = 2 \cdot t_{r2} + 1 \qquad g_3 = 2 \cdot t_{r3} + 1$$

a ponieważ z założenia $g_3>g_2>g_1$, więc $t_{r3}>t_{r2}>t_{r1}$.

Trzy rozpatrywane przypadki iloczynu złożonego $p_1 \cdot p_2 \cdot p_3$ zawieraja się w modelu MFK i ostatecznie są określane przez 3 rozwiązania:

para $\{t_{r1}, m_{r1}\}$ określa iloczyn $g_1=p_2 \cdot p_3=2 \cdot t_{r1}+1$ oraz $p_1=2 \cdot m_{r1}+1$

para $\{t_{r2}, m_{r2}\}$ określa iloczyn $g_2=p_1 \cdot p_3=2 \cdot t_{r2}+1$ oraz $p_2=2 \cdot m_{r2}+1$

para $\{t_{r3}, m_{r3}\}$ określa iloczyn $g_3=p_1 \cdot p_2=2 \cdot t_{r3}+1$ oraz $p_3=2 \cdot m_{r3}+1$

W podobny sposób można analizować problem iloczynu czterech liczb $p_1>p_2>p_3>p_4$, zakładając, że są to liczby pierwsze. Tu, oprócz iloczynów określonych przez pary liczb, należy uwzględnić także iloczyny trzech spośród czterech czynników i wtedy rozpatrywane będzie 7 przypadków:

$$n = p_1 \cdot p_2 \cdot p_3 \cdot p_4 = (p_1 \cdot p_4) \cdot (p_2 \cdot p_3)$$

$$n = p_1 \cdot p_2 \cdot p_3 \cdot p_4 = (p_1 \cdot p_3)(p_2 \cdot p_4)$$

$$n = p_1 \cdot p_2 \cdot p_3 \cdot p_4 = (p_1 \cdot p_2)(p_3 \cdot p_4)$$

$$n = p_1 \cdot p_2 \cdot p_3 \cdot p_4 = (p_2 \cdot p_3 \cdot p_4) \cdot p_1$$

$$n = p_1 \cdot p_2 \cdot p_3 \cdot p_4 = (p_1 \cdot p_3 \cdot p_4) \cdot p_2$$

$$n = p_1 \cdot p_2 \cdot p_3 \cdot p_4 = (p_1 \cdot p_2 \cdot p_4) \cdot p_3$$

$$n = p_1 \cdot p_2 \cdot p_3 \cdot p_4 = (p_1 \cdot p_2 \cdot p_3) \cdot p_4$$

Wszystkie rozpatrywane przypadki iloczynu złożonego $p_1 \cdot p_2 \cdot p_3 \cdot p_4$ zawierają się w modelu MFK i określane są przez 7 rozwiązań (przyjęliśmy $p_1 \neq p_2 \neq p_3 \neq p_4$). Dopuszczając, że $p_1=p_2=p_3=p_4=p$ w modelu MFK mamy tylko dwa rozwiązania – parę $\{p^2, p^2\}$ oraz $\{p^3, p\}$, przy czym pierwsze jest w punkcie bazowym.

Dotychczasowa dyskusja pozwala przyjąć, że model MFK ma zastosowanie w ogólnym przypadku problemu faktoryzacji, tj. faktoryzacji iloczynu różnych k czynników pierszych $n=p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_k$. Dodatkowo można także twierdzić, że każdy problem $n=p \cdot q$ (p, q – nieznane) można sprowadzić do problemu iloczynu czterech czynników, z których dwa p_1 i q_1 są znane i wszystkie cztery niekoniecznie muszą być liczbami pierwszymi. Wtedy

$$n = p \cdot q \cdot p_1 \cdot q_1$$

przy czym p_1 i q_1 można tak dobrać, żeby spełnione było uszeregowanie $p>p_1>q_1>q$ (pamiętamy własność ciągów $\{m_i\}$ oraz $\{k_i\}$ MFK).

Można łatwo sprawdzić, że rozwiązania względem punktu bazowego określane parami w ciągach $\{m_i\}$ i $\{k_i\}$ są uszeregowane następująco:

dla $i=i_b+j1$	- m_i określa $(p_1 \cdot q_1)$, zaś k_i określa $(p \cdot q)$	(*)
dla $i=i_b+j2$	- m_i określa $(p \cdot q_1)$, zaś k_i określa $(p_1 \cdot q)$	
dla $i=i_b+j3$	- m_i określa $(p \cdot p_1)$, zaś k_i określa $(q_1 \cdot q)$	
dla $i=i_b+j4$	- m_i określa $(p_1 \cdot q \cdot q_1)$, zaś k_i określa p	
dla $i=i_b+j5$	- m_i określa $(p \cdot q \cdot q_1)$, zaś k_i określa p_1	(*)
dla $i=i_b+j6$	- m_i określa $(p \cdot p_1 \cdot q)$, zaś k_i określa q_1	(*)
dla $i=i_b+j7$	- m_i określa $(p \cdot p_1 \cdot q_1)$, zaś k_i określa q	

gdzie: $j1<j2< \ldots < j7$.

Te rozwiązania, które można wskazać bezpośrednio zostały oznaczone (*). Rozwiązanie $(p_1 \cdot q_1) \cdot (p \cdot q)$ tak sformułowanego problemu iloczynu 4 czynników, w modelu MFK może być usytuowane w pewnej odległości od punktu bazowego. Analiza tej odległości pozwala wnioskować jak bardzo różnią się od siebie oba iloczyny (jaka jest ich różnica $p_1 \cdot q_1 - p \cdot q$; im większa tym p_1 będąc mniejszym różni się od p, zaś q_1 będąc większym różni się od q. (Uwaga: pamiętamy, że w wyjściowym modelu dla $n=p \cdot q$,

wyrazy ciągów $\{m_i\}$ rosną, a $\{k_i\}$ maleją przy oddalaniu się od punktu bazowego).

Jeśli proces doboru p_1 i q_1 będzie prowadzony w oparciu o wyjściowy model MFK dla $n=p\cdot q$, to wybrane p_1 i q_1 mogą spełniać uszeregowanie $p_1>p>q>q_1$.

Ponownie łatwo sprawdzić, że rozwiązania ujawniają się w ciągach $\{m_i\}$ i $\{k_i\}$, ale w nieco zmienionym usytuowaniu względem bunktu bazowego.

dla $i=i_b+j1$ - m_i określa $(p\cdot q)$, zaś k_i określa $(p_1\cdot q_1)$ (*)
dla $i=i_b+j2$ - m_i określa $(p_1\cdot q)$, zaś k_i określa $(p\cdot q_1)$
dla $i=i_b+j3$ - m_i określa $(p\cdot p_1)$, zaś k_i określa $(q_1\cdot q)$
dla $i=i_b+j4$ - m_i określa $(p\cdot q\cdot q_1)$, zaś k_i określa p_1 (*)
dla $i=i_b+j5$ - m_i określa $(p_1\cdot q\cdot q_1)$, zaś k_i określa p
dla $i=i_b+j6$ - m_i określa $(p\cdot p_1\cdot q_1)$, zaś k_i określa q
dla $i=i_b+j7$ - m_i określa $(p\cdot p_1\cdot q)$, zaś k_i określa q_1 (*)

Rozwiązania, które można ustalić wprost zaznaczono ponownie (*). Zauważmy, że teraz rozłożone są one nieco inaczej względem bazy poza pierwszym, który charakteryzuje inna wzajemna relacja iloczynów pq i p_1q_1.

6.3 Podsumowanie

Zaobserwowane zróżnicowane relacje iloczynów pq i p_1q_1 względem punktu bazowego w „komplikowanym"modelu MFK mogą stanowić kryterium dla systematycznego ustalenie metodą połowienia poszukiwanych czynników danego iloczynu n=pq.

Dla relacji $p>p_1>q_1>q$ iloczyn $p\cdot q$ określany jest przez wyraz ciągu $\{k_i\}$, natomiast dla $p_1>p>q>q_1$ iloczyn $p\cdot q$ określany jest przez wyraz ciągu $\{m_i\}$ modelu MFK. Podobnie jest z drugą parą czynników $(p_1\cdot q)$ oraz $(p\cdot q_1)$ – dla relacji $p>p_1>q_1>q$ iloczyn $p_1\cdot q$ określany jest przez wyraz ciągu $\{k_i\}$, natomiast dla $p_1>p>q>q_1$ iloczyn $p_1\cdot q$ określany jest przez wyraz ciągu $\{m_i\}$ modelu MFK.

R o z d z i a ł s i ó d m y

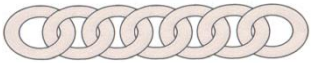
Zupełność problemu *mult(n)*

Marek Malinowski

Technologie Teleinformatyczne

Zasadniczą dyskusję poprzedza przypomnienie roli pojęcia zupełności w teorii złożoności. Wskazano przesłanki uzasadniające prawdziwość hipotezy P=NP i szerzej NC=P=NP. Przywołane zostały podstawowe stwierdzenia odnoszące się do sieci logicznych, w tym zasadniczy dla prowadzonych rozważań sposób redukcji sieci CIRCUIT SAT do SAT. Za punkt wyjścia dla pokazania takiej redukcji przyjęto sprzętową realizację operacji mnożenia w postaci układu kombinacyjnego. Pokazano, że rozszerzenie układu multiplikatora o dodatkowe elementy pozwala skonstruować sieć logiczną o wielomianowym rozmiarze i że jest to sieć jednostajna.

Prezentowane wyniki upoważniają do stwierdzenia, że dla problemu *mult(n)* istnieje redukcja wielomianowa do problemu **SAT**. Pozwala to zakwalifikować problem *mult(n)* do klasy problemów **NP-zupełnych**.

7 

to co stracono, może być znalezione
często nie tam, gdzie strata powstała

[R. Ackoff], The Art of Problem Solving – tłum. wł.*]*

Pojęcie zupełności stanowi bardzo ważne narzędzie metodologiczne teorii złożoności. Wykorzystywane jest do charakteryzowania złożoności problemów w kategoriach ich wzajemnej „równoważności". Zupełność problemów jest " … *więzią utrzymującą klasy złożoności przy życiu i łączą je z praktyką obliczeniową. …* " oraz " … *głównym narzędziem wykazywania równości klas złożoności.*" [2 s.182].

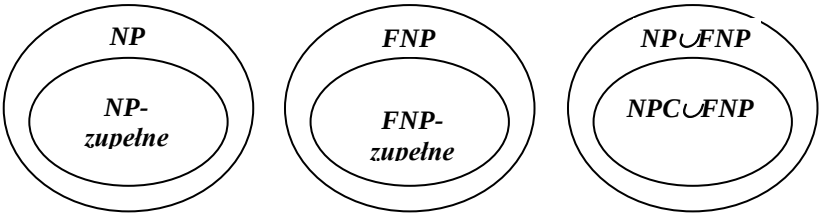
Zatem, jeśli dla jakiegoś problemu kwalifikowanego do pewnej klasy ***C*** udałoby się skonstruować algorytm o złożoności słabszej klasy ***C'*** w uszeregowaniu $C' \subseteq C$, to aby twierdzić o równości obu klas $C'=C$, musiałby to być problem ***C-zupełny***.

Wiedząc, że problem *mult(n)* można rozwiązać w sposób charakteryzujący klasę ***NC*** (klasa Nick'a), to brakującym ogniwem do pokazania relacji równości klas ***NC***, ***P*** i ***NP*** jest wykazanie, że problem *mult(n)*, jako szczególny przypadek faktoryzacji, jest problemem ***NP-zupełnym***. Bez tego, podobnie jak w przypadku problemu PRIMES, moglibyśmy tylko mówić o wyizolowanym problemie, dla którego zawężono lukę algorytmiczną. ***NP-zupełność*** problemu *mult(n)* zwiąże go ze wszystkimi problemami kwalifikowanymi do tej klasy i przesą-dzi o ich złożoności obliczeniowej.

W przypadku *mult(n)* należałoby mówić o ***FNP-zupełności***, gdzie ***FNP*** oznacza klasę wszystkich problemów funkcyjnych (obliczeniowych). Przykładem problemu funkcyjnego jest

problem **FSAT**. Polega on na tym, że dla danej formuły φ należy wskazać (obliczyć) wartościowanie(a) zmiennych spełniające jeśli φ jest spełniana, a nie tylko stwierdzić czy takie wartościowanie istnieje, jak w przypadku problemu decyzyjnego **SAT**.

Zakładając, że klasy problemów funkcyjnych ***FNP*** i decyzyjnych ***NP*** są rozłączne i odnosząc to także do problemów zupełnych w obu klasach, relacje zawierania klas można przedstawić graficznie (rys. 7.1).



Rys. 7.1 Relacje zawierania problemów decyzyjnych i funkcyjnych

Podobnie jak w przypadku problemów decyzyjnych, także dla problemów funkcyjnych można wskazać problemy zupełne – jest nim przywołany już wcześniej problem **FSAT** [2 s. 248]. Ponadto, podobnie jak w przypadku problemów decyzyjnych, zupełność problemów funkcyjnych pokazuje się z wykorzystaniem redukcji [2 s. 247].

Pokazanie algorytmów równoległych dla problemów *nwd(a,b)* i *mult(n)* bazowało na zupełnie nowych modelach. W przypadku dyskusji redukcji problemu *mult(n)* do **FSAT** i **SAT**, proces wnioskowania oparto na znanych już wynikach badań i stosowanych w teorii złożoności modelach.

Jako punkt wyjścia dla kolejnych etapów wnioskowania przyjęto model multiplikatora równoległego, zrealizowanego w postaci układu kombi-nacyjnego. Ostatecznie, ***FNP-*** i ***NP-****zupełność* problemu *mult(n)* jest wynikiem sekwencji wnioskowania, na które składają się stwierdzenia:, (i) że sieć logiczna układu multiplikatora jest wielowyjściową siecią wielomianową; (ii) że każde wyjście sieci logicznej multiplikatora oblicza inną funkcję logiczną; (iii) że funkcje logiczne opisujące wyjście sieci multiplikatora, każda oddzielnie, po tożsamościowej zamianie na wejściach multiplikatora stałych przez zmienne,

definiują odrębne problemy **FSAT**, a z racji dualizmu sieci także problemy **SAT**; (iv) że koniunkcja odrębnych problemów **FSAT** (i odpowiednio **SAT**) związanych z wyjściami sieci multi-plikatora definiuje problem **FSAT**.

Ponieważ w uzasadnianiu prawdziwości poszczególnych stwierdzeń, w toku wnioskowania wykorzystywane są elementy teorii układów kombinacyjnych, ekwiwalentność formuł logicznych i funkcji logicznych oraz sieci logiczne, naturalnym jest przedstawienie podstawowych zagadnień związanych z nimi.

7.1 Sieć logiczna – teoretyczny model wyrażeń logicznych

Teoria złożoności obliczeniowej dostarcza szeregu stwierdzeń odnoszących się do modelu sieci logicznych. Przedstawione poniżej w postaci faktów, przywoływane są w kolejnych etapach wnioskowania zupełności problemu *mult(n)*.

Fakt 7.1: Ekwiwalentność formuł logicznych i funkcji logicznych [2 s. 95].

Każda formuła φ ze zmiennymi $x_1, \ldots, x_n$ wyraża n-argumentową funkcję logiczną f i odwrotnie, dowolna n-argumentowa funkcja logiczna f może być wyrażona jako formuła φ zawierająca zmienne $x_1, \ldots, x_n$. ■

Fakt 7.2: Reprezentacja funkcji logicznych przez sieci logiczne [2 s. 95-96].

Każda funkcja logiczna może być przedstawiona jako sieć logiczna, zdefiniowaną jako graf $C=(V,E)$, w którym $V=\{1, \ldots, n\}$ są bramkami grafu C, zaś E zbiorem krawędzi (i,j) przy $i<j$. ■

Składnia sieci ponadto wiąże z każdą bramką $i \in V$ rodzaj $s(i) \in \{true,false\} \cup \{x_1, \ldots, x_n\} \cup \{\vee, \wedge, \neg\}$ i charakteryzuje je stopniem wejściowym (liczba wchodzących krawędzi) równym *0*, *1* lub *2*. Wierzchołek o numerze n, który nie ma krawędzi wychodzących, nazywany bramką wyjściową sieci oblicza funkcję logiczną.

Jeśli będziemy rozważać sieci, które mają kilka wyjść, to obliczają one kilka funkcji jednocześnie.

Kolejne stwierdzenia wiążą sieci logiczne z problemami różnych klas złożoności obliczeniowej, i tak:

Fakt 7.3: Sieć logiczna jako problem **CIRCUIT VALUE** [2 s. 97].

Sieć *C* bez zmiennych (tj. sieć z bramkami $V(i) \in \{true,false\} \cup \{\vee, \wedge, \neg\}$) definiuje problem **CIRCUIT VALUE** $\in P$.■

Fakt 7.4: Sieć logiczna jako problem **CIRCUIT SAT** [2 s. 97].

Sieć *C* ze zmiennymi, kiedy należy stwierdzić czy istnieje takie wartościowanie zmiennych, że bramka wyjściowa zwraca wartość *true*, definiuje problem **CIRCUIT SAT** $\in$ ***NP-****zupełnych*.■

Fakt7. 5: Redukcja problemu **CIRCUIT SAT** do **SAT** [2 s. 179-180].

Problem **CIRCUIT SAT** redukuje się w czasie wielomianowym do **SAT**. Redukcja wykorzystuje fakt, że formuły i sieci są różnymi sposobami przedstawienia funkcji logicznych i tłumaczenie jednej reprezentacji na drugą i odwrotnie jest proste.■

W formule $\varphi(C)$ występować będą zmienne z *C* oraz zmienne *g* ozna-czające bramki w sieci. Formułę $\varphi(C)$ konstruujemy następująco:

- jeżeli *g* jest bramką odpowiadającą zmiennej *x*, to dodajemy dwie klauzule $(\neg g \vee x)$ i $(g \vee \neg x)$,
- jeżeli *g* jest bramką o wartości *true*, to dodajemy klauzulę (g),
- jeżeli *g* jest bramką o wartości *false*, to dodajemy klauzulę $(\neg g)$,
- jeżeli *g* jest bramką NOT, a jej poprzednikiem w sieci jest bramka *h*, to dodajemy dwie klauzule $(\neg g \vee h)$ i $(g \vee \neg h)$,
- jeżeli *g* jest bramką OR o poprzednikach w postaci bramek *h* i *h'*, to dodajemy klauzule $(\neg h \vee g)$, $(\neg h' \vee g)$ i $(h \vee h' \vee \neg g)$,
- jeżeli *g* jest bramką AND o poprzednikach *h* i *h'*, to dodajemy klauzule $(\neg g \vee h)$, $(\neg g \vee h')$ i $(g \vee \neg h \vee \neg h')$,

i ostatecznie, jeżeli *g* jest bramką wyjściową, to do $\varphi(C)$ dodajemy klauzulę (g).

Kolejne przytaczane stwierdzenia są wynikiem prac wiążących złożoność obliczeniową ze złożonością sieci logicznych. Złożoność sieci jest określana jako rozmiar sieci wyrażany liczbą bramek w tej sieci.

Fakt 7.6: Sieci wielomianowe [2 s. 285-286].

Sieć ma rozmiar wielomianowy, jeśli istnieje rodzina sieci $C=\{C_0, C_1, \dots\}$ dla której prawdą jest po pierwsze, że rozmiar C_n jest równy co najwyżej $p(n)$ dla pewnego ustalonego wielomianu p i po drugie, że dla każdej kombinacji zmiennych $x_i \in \{0,1\}$ $i=1,2, \dots, n$ spełniających formułę reprezentowaną przez tą sieć, wartością sieci jest *true* (w kategoriach języka maszyn Turinga powiedzielibyśmy, że dla słów wejściowych, które maszyna akceptuje – wartością sieci jest *true*).■

Fakt 7.7: Każdy problem z klasy *P* ma sieć wielomianową [2 s.286].■

Niestety, stwierdzenie odwrotne nie jest prawdziwe. Nie każda sieć wielomianowa jest reprezentacją problemu z klasy ***P*** (np. pokazano, że istnieją problemy nierozstrzygalne, które mają sieci wielomianowe).

Związanie sieci wielomianowych z obliczeniami wielomianowymi dokonuje się poprzez wprowadzenie pojęcia jednostajności sieci.

Fakt 7.8: Wielomianowe sieci jednostajne [2 s.287].

Rodzina sieci $C=\{C_0, C_1, \dots\}$ jest jednostajna, jeśli dla zmiennych $x_i \in \{1\}$ $i=1, 2, \dots, n$ możliwe jest skonstruowanie sieci C_n w pamięci $\log n$.■

Tak zdefiniowana jednostajna sieć wielomianowa prowadzi do stwierdzenia, że dla reprezentowanego przez nią problemu jednostajna rodzina sieci wielomianowych istnieje wtedy i tylko wtedy, gdy problem ten należy do klasy ***P***.

Ostatnie stwierdzenie otwiera także drogę do sformułowania równoważnej dla hipotezy $\boldsymbol{P \neq NP}$ hipotezy, że problemy ***NP-****zupełne* nie mają jednostajnych sieci wielomianowych ani niejednostajnych sieci wielomianowych.

Z jednej strony, jako mocną przesłankę przemawiająca za tą hipotezą uważa się twierdzenie Razborowa, mimo że odnosi się ono do słabszego modelu sieci – sieci monotonicznych (czyli sieci bez bramek *NOT*) oraz określane przez nie wykładnicze dolne ograniczenie rozmiaru sieci dotyczy konkretnego problemu z klasy ***NP-****zupełnych* (*CLIQUE(n,k)*) [2 s.362-363].

Z drugiej strony, należy zachować pewną ostrożność w uogólnianiu tego wyniku na całą klasę problemów ***NP-****zupełnych*. Tym

bardziej, że sieci monotoniczne mogą obliczać wyłącznie funkcje monotoniczne (tj. takie funkcje logiczne, które nie zmieniają wyniku z *true* na *false* przy zmianie jednego dowolnego argumentu z *false* na *true*), a nie wszystkie problemy ***NP-****zupełne* (np. *BISECTION WIDTH*, *NODE COVER*, *KNAPSACK*) są monotoniczne [2 s. 362].

Poczynione wyżej spostrzeżenia dotyczące badań nad złożonością sieci zmierzających do potwierdzenia przy ich pomocy hipotezy $\boldsymbol{P \neq NP}$, skłaniają do poważnego potraktowania przeciwnej hipotezy $\boldsymbol{P = NP}$. Tym bardziej, że w stosunku do przedstawionego przez Papadimitriou [2 s. 98] dowodu twierdzenia pochodzącego z pracy Shannona

> *„Dla dowolnego $n \geq 2$ istnieje n-argumentowa funkcja logiczna f, taka że nie istnieje sieć logiczna o co najwyżej $2^n/(2 \cdot n)$ bramkach, która oblicza f."*

można mieć pewne zastrzeżenia.

W dowodzie zakłada się, że dla $n \geq 2$ wszystkie n-argumentowe funkcje logiczne mają sieci o najwyżej $m=2^n/(2 \cdot n)$ bramkach. Dalej szacuje się ilość wszystkich możliwych do zbudowania sieci o co najwyżej m bramkach, określając tą ilość wyrażeniem $((n+5) \cdot m^2)^m$.

Porównując otrzymane oszacowanie ilości sieci z ilością funkcji logicznych ograniczoną do 2^{2n} zauważamy, że ilość możliwych sieci jest mniejsza od ilości funkcji. W związku z tym wnioskuje się następująco:

> *„założywszy, że każda funkcja logiczna o n zmiennych może być obliczona przez sieć o najwyżej m bramkach, doszliśmy do wniosku, że dwie różne n-argumentowe funkcje logiczne f i f' są obliczane przez tę samą sieć, co oczywiście jest niemożliwe".*

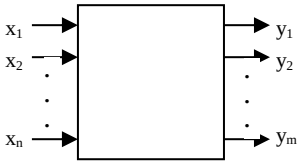
Sam sposób szacowania ilości sieci nie budzi zastrzeżeń. W końcowym etapie wnioskowania nie uwzględnia się jednak faktu, że w dopuszczonym przy szacowaniu ilości sposobie konstruowania sieci są także sieci, w których część bramek nie jest ze sobą połączona. Wtedy w istocie mamy do czynienia z sieciami o wielu wyjściach, a każde z nich może obliczać inną funkcję logiczną, w tym funkcje n-argumentowe.

Kolejne zastrzeżenie wynika z analizy sieci dla funkcji $parity_n$ (zwraca wartość *true*, gdy na wejściach jest dowolna nieparzysta liczba jedynek). Taką sieć dla n zmiennych łatwo skonstruować przy pomocy $(n-1)$ bramek *EXOR*, przy czym ta sieć oblicza także funkcje $parity_2$ (gdy dokładnie 3 są nieparzyste), $parity_4$ (gdy dokładnie 5 jest nieparzystych) itd., łącznie oblicza więc $2^{(n-1)}$ funkcji logicznych.

7.2 Układy kombinacyjne – techniczna realizacja funkcji logicznych

Zdefiniowane w poprzednim punkcie sieci logiczne są dogodnym modelem teoretycznym. W wymiarze praktycznym, reprezentantem sieci logicznych są układy kombinacyjne wykorzystywane do przetwarzania cyfrowego.

Jeśli przyjmiemy, że układ kombinacyjny jest wielobiegunnikiem (rys. 7.2), w którym stan m wyjść jest jedno-znacznie określany przez aktualny stan n wejść, to możemy zapisać $y_i = f_i(x_1, x_2, \ldots, x_n)$, gdzie f_i jest funkcją logiczną związaną z jednym z i-tym wyjściem $(i=1, 2, \ldots, m)$.



Rys. 7.2 Układ kombinacyjny – schemat wielobiegunnika

W teorii układów kombinacyjnych, funkcję f która wartościom binarnym zmiennych x przyporządkowuje wartość binarną zmiennych y, nazywamy funkcją przełączającą.

Każda funkcja przełączająca może być opisana za pomocą zmiennych i operacji ze zbioru stanowiącego system funkcjonalnie pełny [3 s.23].

Podstawowy system funkcjonalnie pełny (**SFP)** stanowią operacje $O \in \{\vee, \wedge, \neg\}$ realizowane przez bramki *OR*, *AND* i *NOT*. Inne **SFP**, mające praktyczne znaczenie ze względu na łatwość wykonania elementu realizującego operacje, to:

$\{\vee, \neg\}$ - bo w oparciu o $a \wedge b = \neg(\neg a \vee \neg b)$ iloczyn można wyrugować,
$\{\wedge, \neg\}$ - bo w oparciu o $(a \vee b) = \neg(\neg a \wedge \neg b)$ sumę można wyrugować,
$\{\downarrow\}$ - bo $a \downarrow a = \neg a$ oraz $\neg(a \downarrow b) = a \vee b$ (bramka *NOR* – funkcja Peirce'a),
$\{|\}$ - bo $a|a = \neg a$ oraz $\neg(a|b) = a \wedge b$ (bramka *NAND* – funkcja Sheffera),
$\{\oplus, \wedge, 1\}$ - bo $a \oplus 1 = \neg a$.

Z zestawienia wynika, że do zbudowania układu kombinacyjnego wystarcza jeden typ uniwersalnego elementu realizującego operator *NOR* albo *NAND*.

W procesie projektowania układów kombinacyjnych, ważnym elementem jest upraszczanie postaci funkcji przełączającej (dla układów wielowyjściowych funkcji przełączających) tak, aby realizacja układu była prosta i tania tj., by stosować standardowe bramki oraz by układ charakteryzował się minimalną ilością bramek.

Optymalizacja rozwiązania polega więc na sprowadzeniu opisu funkcji do postaci o najmniejszej liczbie składników. W tym celu wykorzystywane są różnorodne metody minimalizacji, a wśród nich najbardziej typowe - metoda tablic Karnaugha oraz metoda Quinea-McCluskeya.

W przypadku układów kombinacyjnych o kilku wyjściach, każde z nich opisywane jest przez funkcję przełączającą. Możliwe jest wtedy, że parami funkcje przełączające y_i i y_j mają wspólne wyrażenia. Odpowiada to sytuacji $y_i \wedge y_j \neq \varnothing$. Wyszczególnienie wspólnych wyrażeń możliwe jest także dla sytuacji $y_i \wedge y_j = \varnothing$, bo wtedy zachodzi $(\neg y_i \wedge y_j) \neq \varnothing$ i $(y_i \wedge \neg y_j) \neq \varnothing$. Wspólne wyrażenia odpowiadają wspólnym kombinacjom sygnałów wejściowych (zmienne x).

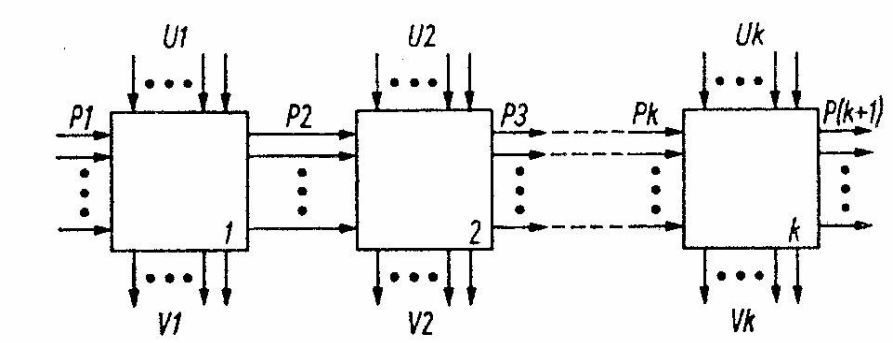
W praktyce projektowania układów kombinacyjnych mamy do czynienia z występowaniem sytuacji, w których wykonywane są wielokrotnie te same czynności. Pozwala to na budowanie układu z jednakowych członów, które są ze sobą łączone tworząc tzw. układy iteracyjne (rys. 7.3).

Przykładem realizacji układu kombinacyjnego, zbudowanego z jednakowych członów iteracyjnych może być sumator liczb binarnych.

Dla każdej *i-tej* pary cyfr dwóch liczb binarnych a i b trzeba wygenerować:

wynik $s_i=(\neg a_i\wedge\neg b_i\wedge p_i)\vee(\neg a_i\wedge b_i\wedge\neg p_i)\vee(a_i\wedge b_i\wedge p_i)\vee(a_i\wedge\neg b_i\wedge\neg p_i)$

oraz przeniesienie $p_{i+1}=(a_i\wedge b_i)\vee(a_i\wedge p_i)\vee(b_i\wedge p_i)$.



Rys. 7.3 Sygnały wejściowe $x_1, x_2, \ldots, x_n$ są składnikiem wektorów U_i, sygnały wyjściowe $y_1, y_2, \ldots y_m$ są elementami wektorów V_i i/lub P_i (i=1, 2, … , k)

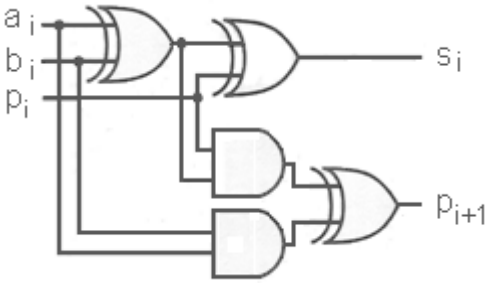
Wyrażenie s_i można przekształcić, wyłączając wspólne czynniki poza nawias i wtedy przyjmie postać (dla zwiększenia czytelności zapisu, operator iloczynu logicznego $\wedge$ zastąpimy zwykłym operatorem mnożenia w postaci kropki):

$$s_i=\neg a_i\cdot(\neg b_i\cdot p_i\vee b_i\cdot\neg p_i)\vee a_i\cdot(b_i\cdot p_i\vee\neg b_i\cdot\neg p_i)=\neg a_i\cdot(b_i\oplus p_i)\vee a_i\cdot(b_i\equiv p_i)=a_i\oplus b_i\oplus p_i$$

W zapisie operatorowym stosowanym w układach kombinacyjnych, otrzymana postać s_i wyraża się następująco $s_i=EXOR(p_i,EXOR(a_i,b_i))$, natomiast postać wyrażenia p_{i+1} umożliwiającego użycie bramek *EXOR*, możemy uzyskać wykorzystując zależność $a\vee b=a\oplus b\vee a\cdot b$ i wtedy:

$$p_{i+1}=a_i\cdot b_i\vee a_i\cdot p_i\vee b_i\cdot p_i=a_i\cdot b_i\vee p_i\cdot(a_i\vee b_i)=a_i\cdot b_i\vee((a_i\oplus b_i)\vee a_i\cdot b_i)\cdot p_i=a_i\cdot b_i\vee(a_i\oplus b_i)\cdot p_i=$$
$$=a_i\cdot b_i\oplus(a_i\oplus b_i)\cdot p_i\vee a_i\cdot b_i\cdot(a_i\oplus b_i)\cdot p_i=a_i\cdot b_i\oplus(a_i\oplus b_i)\cdot p_i\cdot(1\vee a_i\cdot b_i)=a_i\cdot b_i\oplus(a_i\oplus b_i)\cdot p_i.$$

W ten sposób wynikowy układ, przedstawiony na rysunku 7.4, może być prostszy w realizacji.



Rys. 7.4 Schemat sumatora jednopozycyjnego

7.3 Dyskusja operacji mnożenia

7.3.1 Dualizm mnożenia

Efektywność algorytmów wielu problemów w znacznym stopniu zależy od efektywności wykonania elementarnych operacji arytmetycznych dodawania i mnożenia dwóch liczb binarnych. W technice obliczeniowej operacje te wykonywane są przez specjalizowane układy, w tym głównie układy kombinacyjne. Operacja dodawania realizowana jest przez sumatory, zaś operacja mnożenia przez multiplikatory.

Obie operacje są także przedmiotem badań w teorii złożoności oblicze-niowej. Dotyczy to szczególnie operacji mnożenia, a wynika to roli jaką odgrywa w zastosowaniach współczesnej informatyki. Ta szczególna rola polega na związkach mnożenia z problemem faktoryzacji **FACTORING** i traktowaniu mnożenia jako funkcji jednokierunkowej $f_{MULT}(p, q)$ [1] w zastosowaniach kryptografii cyfrowej (złamanie szyfrowania z kluczem jawnym jest problemem z klasy ***FNP***).

Mnożenie dwóch liczb binarnych *a* i *b n-bitowych* jest równoważne dodawaniu najwyżej *n* liczb (iloczynów częściowych), z których każda z nich jest liczbą *a* pomnożoną przez potęgę dwójki, dla której odpowiedni bit w *b* jest równy 1.

[1] Powszechnie przyjmuje się, że mnożenie całkowitoliczbowe licz pierwszych jest funkcją jednokierunkową. Wtedy $f_{MULT}(p,q)$ jest różnowartościowa i może być obliczona w czasie wielomianowym, a jej funkcja odwrotna f^{-1} należy do klasy ***FNP***.

Schemat takiego działania na przykładzie liczb *4-bitowych* ilustruje rysunek 7.5

				p_3	p_2	p_1	p_0
				q_3	q_2	q_1	q_0
				p_3q_0	p_2q_0	p_1q_0	p_0q_0
			p_3q_1	p_2q_1	p_1q_1	p_0q_1	
		p_3q_2	p_2q_2	p_1q_2	p_0q_2		
	p_3q_3	p_2q_3	p_1q_3	p_0q_3			
n_7	n_6	n_5	n_4	n_3	n_2	n_1	n_0

Rys.7.5 Schemat mnożenia

Mnożenie, będące problemem obliczeniowym (w literaturze określane jest także mianem problemu funkcyjnego), może być postrzegane przez pryzmat problemu decyzyjnego. Nieformalne określenie problemu decyzyjnego mówi, że jest to taki problem algorytmiczny, który polega na ustaleniu, czy dla konkretnych danych wejściowych zachodzi pewna własność [1 s.169]. Wynikiem więc może być odpowiedź *«tak»* potwierdzająca taką własność lub *«nie»* stwierdzająca brak takiej własności. W przypadku problemów kwalifikowanych do klasy ***NP-zupełnych***, uzyskanie odpowiedzi potwierdzającej istnienie pewnej własności jest trudne, jednak sprawdzenie, że jest to odpowiedź poprawna jest łatwe i nie wymaga czasu wykładniczego [1 s. 183].

Traktowanie mnożenia jako problemu decyzyjnego mogłoby polegać na sprawdzaniu czy uzyskiwany na wyjściach multiplikatora wynik jest poprawny, tj. czy iloczyn dwóch liczb binarnych *a* i *b* podanych na wejścia układu jest równy pewnej znanej zapisanej binarnie wartości ich iloczynu. Rolę badanej własności pełniłaby więc właśnie ta wartość znanego iloczynu.

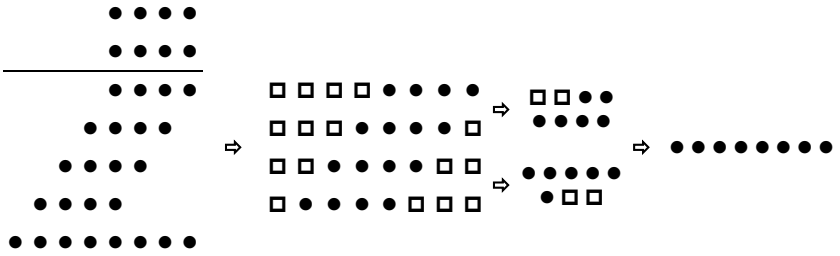
Traktowany w ten sposób układ multiplikatora realizujący normalnie mnożenie dwóch czynników, może być użyty do sprawdzania, jakie czynniki na wejściach układu pozwalają uzyskać pewną zadaną wartość iloczynu. Określony tak problem staje się w oczywisty sposób problemem znalezienia odwrotności operacji mnożenia i jest wtedy problemem funkcyjnym **FSAT**. Jednak gdy chcemy tylko stwierdzić, czy w ogóle istnieją czynniki pozwalające uzyskać zadaną wartość iloczynu (intere-sują nas

odpowiedzi tak nie) to mamy do czynienia z problemem decyzyjnym **SAT**.

7.3.2 Model multiplikatora

Opracowano szereg algorytmów mnożenia równoległego, dla których istnieje możliwość sprzętowej realizacji w postaci układu kombinacyjnego. Warto w tym miejscu wskazać równoległe algorytmy drzewa Wallace'a i algorytm Booth'a.

W dalszej dyskusji wykorzystany będzie nadmiarowy model algorytmu, w którym uzyskanie iloczynu dwóch liczb *n-bitowych* jest możliwe przez równolegle wykonywane sumowanie najwyżej *n* liczb *2n-bitowych* za pomocą co najwyżej *(2·n-1)* sumatorów tworzących drzewo o głębokości $\log_2 n$. Przykładowy model multiplikatora przedstawia rysunek 7.6.



Rys. 7.6 Przykładowy model multiplikatora dla *n=4*

Schemat rozpatrywanego multiplikatora daje się przedstawić jako w pełni uporządkowana struktura. Sumatory na każdym poziomie drzewa mają charakter układów iteracyjnych (rys. 7.3). Zbudowane są z suma-torów jednopozycyjnych (rys. 7.4) o identycznej ilości elementarnych bramek i identycznym sposobie połączeń między nimi. Wejścia sumatorów pierwszego poziomu określane są przez wyjścia bramek generujących iloczyny częściowe.

Zdefiniowany schemat układu multiplikatora pozwala ustalić jego rozmiar wyrażony ilością elementarnych bramek. Analizowany w po-przednim punkcie sumator jednopozycyjny zbudowany jest z 3 bramek *EXOR* i 2 bramek *AND*. Uwzględniając fakt, że bramkę *EXOR* można zastąpić przez 2 bramki *NOT*, 2 bramki *AND* i 1 bramkę *OR*, to odpowiadająca temu sumatorowi jednopozycyjnemu sieć logiczna ma rozmiar 17 bramek (6 bramek *NOT*, 8 bramek *AND* i 3 bramki *OR*).

W rezultacie jeden sumator *2n-bitowy* zbudowany jest z $17 \cdot (2 \cdot n) = 34 \cdot n$ bramek.

Ostatecznie, rozpatrywany układ multiplikatora zawiera $34 \cdot n \cdot (n-1)$ bramek sumatorów (w układzie równoległym jest $n-1$ sumatorów) i n^2 bramek *AND* służących wygenerowaniu sumowanych równolegle liczb. W ten sposób, układ takiego multiplikatora charakteryzuje się wielomianową ilością bramek logicznych, którą możemy dokładnie określić. Łączna ilość bramek *NOT, AND i OR* wyraża się wielomianem drugiego stopnia.

Regularna struktura elementarnych bloków funkcjonalnych układu multiplikatora, jednoznacznie określone ilość i sposób połączeń, pozwalają w uporządkowany sposób ponumerować wszystkie bramki i ich wejścia oraz wyjścia.

Sprzętowa realizacja multiplikatora ma postać układu kombinacyjnego, którego działanie można opisać przy pomocy funkcji przełączających. Prezentowany wcześniej (rys. 7.5) schemat mnożenia dwóch liczb binarnych, niezależnie od tego jak będzie realizowane sumowanie iloczynów częściowych, pozwala określić dla każdego wyjścia n_j układu multiplikatora odpowiednią funkcję przełączającą f_j $(j=1, 2, \ldots, 2n)$.

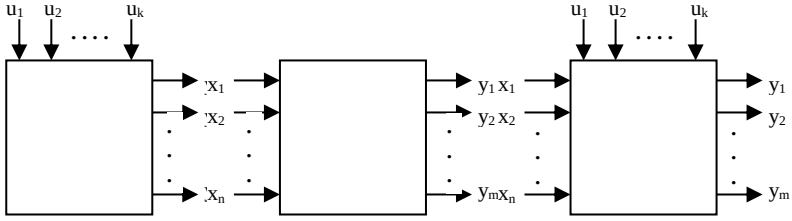
Uwzględniając ekwiwalentność formuł logicznych i funkcji logicznych, układ multiplikatora może być rozpatrywany w kategoriach sieci logicznych.

7.4 Dualizm sieci logicznych

Wiemy (fakt 7.2), że każda funkcja logiczna może być przedstawiona jako sieć logiczna, a ta w wymiarze praktycznym może być zrealizowana jako układ kombinacyjny z bramkami podstawowego **SFP** (bramki *AND*, *OR* i *NOT*). Określona w definicji sieci logicznej składnia sieci pozwala na stwierdzenie, że każdy układ kombinacyjny, nawet wtedy, gdy jest zrealizowany przy pomocy bramek niekoniecznie podstawowego **SFP**, można przedstawić w postaci sieci logicznej.

Omawiając elementy teorii układów kombinacyjnych, stwierdziliśmy, że układy kombinacyjne można przedstawić w postaci wielobiegunnika. Poza schematem prezentowanym na

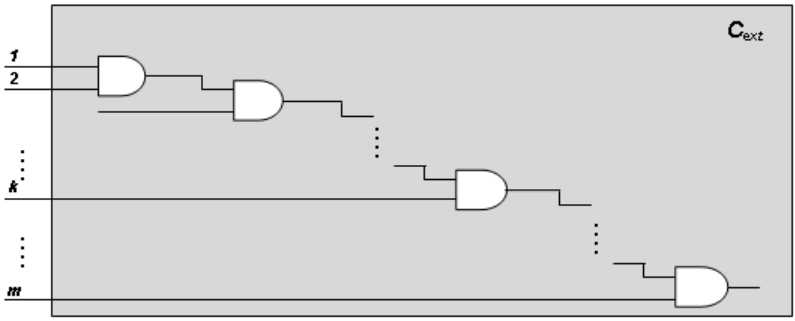
rysunku 7.3, układy kombinacyjne mogą przyjmować jeszcze inne postaci. Na przykład zamiast zmiennych logicznych $x_1, x_2, \ldots, x_n$ na wejściach układu występują stałe logiczne $u_i \in \{true, false\}$ (rys. 7.7 a). Możliwe jest także, że wejścia układu określane są zarówno przez zmienne jaki i stałe (rys. 7.7 c). Wszystkie trzy możliwe postaci ilustruje rys. 7.8.



Rys. 7.7 Schematy ideowe typów sieci logicznych, gdzie:
X – wektor sygnałów wejściowych (zmienne $x_1, x_2, \ldots, x_n$);
U – wektor stałych $u_i \in \{true, false\}$ $i=1, 2, \ldots, k$;
Y – wektor sygnałów wyjściowych, $y_j \in \{true, false\}$ $j=1, 2, \ldots, m$
jest obliczoną wartością pewnej funkcji (niekoniecznie funkcji wszystkich n zmiennych).

Prezentowane schematy układów kombinacyjnych charakteryzują się wieloma wyjściami. Tak więc, odpowiadające im sieci logiczne są sieciami wielowyjściowymi (każde wyjście sieci może obliczać inną funkcję logiczną).

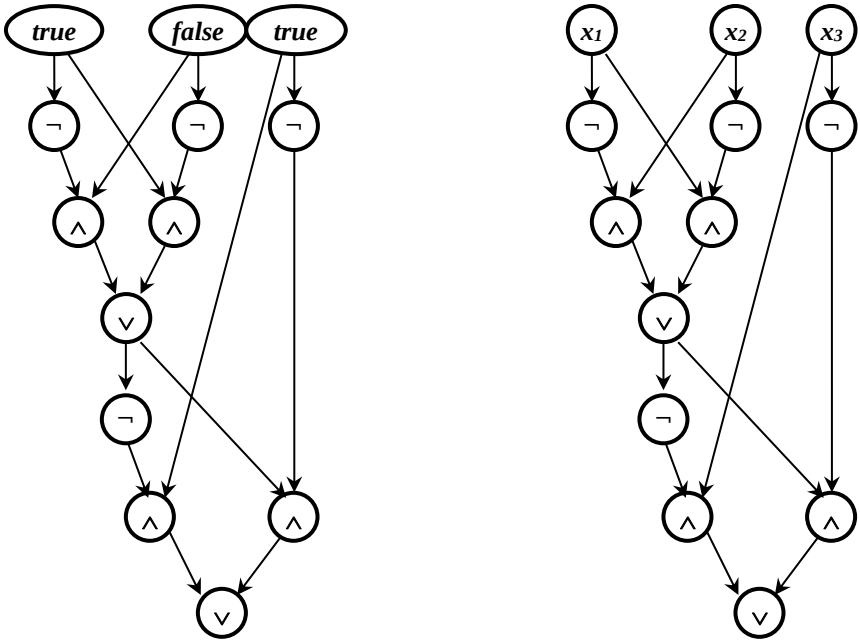
Uwaga 7.1 Każda m-wyjściowa sieć może być sprowadzona do postaci sieci z jednym wyjściem. W szczególności może to być sieć rozszerzająca C_{ext} o rozmiarze $(m-1)$, zbudowana z 2-wejściowych bramek *AND* w układzie równoległym, wtedy jej głębokość jest równa $\log_2 m$ lub w układzie sekwencyjnym (rys. 7.8),



Rys. 7.8 Schemat ideowy sieci C_{ext} w układzie sekwencyjnym.

wtedy jej głębokość jest równa *(m-1)*. Ogólnie, na wejściach bramek *AND* sieci C_{ext} mogą występować inne bramki poprzedzające, niekoniecznie elementarne.

Teoria złożoności wiąże sieci logiczne z dwa problemami – sieć bez zmiennych z problemem **CIRCUIT VALUE** (fakt 7.3) oraz sieć ze zmiennymi z problemem **CIRCUIT SAT** (fakt 7.4).



Rys.7.9 Przykładowe sieci logiczne: (a) bez zmiennych, (b) ze zmiennymi

Ostatecznie, w odniesieniu do sieci logicznych możemy mówić o sieciach obliczających i sieciach rozstrzygających. Przykłady obu rodza-jów sieci ilustruje rys. 7.9. Obie sieci są odwzorowaniem części układu sumatora jednopozycyjnego.

W przypadku sieci bez bramek zmiennych (problem **CIRCUIT VALUE**), aby uzyskać wartość na wyjściu (wyjściach) sieci, wystarczy obliczać wartości bramek stosownie do kolejności określonej przez ich połączenia. Obliczana wartość wyjścia zależy tylko od konkretnych wartości stałych na wejściach sieci.

Zastąpienie na wejściach sieci stałych logicznych przez zmienne, powoduje, że ta sama sieć (taka sama ilość, rodzaj

i sposób połączeń bramek) zmienia swój charakter. Staje się siecią CIRCUIT SAT , ale także siecią pozwalającą ustalać wartość wyjścia (wyjść) w zależności od wartościo-wania zmiennych wejściowych – tak traktowaną sieć nazwiemy siecią CIRCUIT FSAT.

Uwaga 7.2 Możliwość wartościowania zmiennych na wejściach sieci, pozwala ustalić jakie wartościowanie zmiennych odpowiada określonej wartości wyjścia w przypadku sieci z jednym wyjściem lub określonym wartościom wyjść w przypadku sieci z wieloma wyjściami. Tak określony problem jest problemem funkcyjnym i jest pokrewny z FSAT.

Uwaga 7.3 W przypadku sieci z wieloma wyjściami możemy zawsze tak skonstruować sieć rozszerzającą, by móc rozpatrywać sieć z jednym wyjściem (uwaga 7.1) i wtedy sieć może być Rozpatrywana zarówno w kategoriach problemu SAT jak i problemu FSAT.

7.5 Sieć logiczna multiplikatora

7.5.1 Kwestia FNP i NP w modelu multiplikatora

We wstępie sygnalizowaliśmy, że ponieważ problem *mult(n)* jest problemem funkcyjnym, to w zasadzie powinna nas interesować ***FNP-****zupełność*. Stwierdzenie ***FNP-****zupełności* problemu pozwalałoby kwalifikować go do klasy ***FNP*** i dalej w oparciu o twierdzenie, że ***FP = FNP*** wtedy i tylko wtedy, gdy ***P = NP***, rozstrzygać kwestię ***P*** *versus* ***NP*** [2 s.248].

Dotychczasowa dyskusja daje podstawy, by model ultiplikatora mógł być rozpatrywany w kategoriach sieci logicznych jako problem ***FNP***.

Gdy rozpatrujemy multiplikator w kategoriach sieci z wielo-ma wyjścia-mi i dodatkowo uwzględnimy dualny charakter mnożenia, w odniesieniu do ***FNP- NP-****zupełności* mamy do czynienia z dwoma sytuacjami.

Po pierwsze, niezależnie od charakteru mnożenia, wszystkie wyjścia sieci multiplikatora dają się oddzielnie opisać funkcją zmiennych wejściowych (uwaga 7.2). Każda z nich oddzielnie stanowi problem FSAT, a ponieważ jest on wynikiem redukcji, to jest ***FNP-****zupełnym*.

Po drugie, przy rozpatrywaniu multiplikatora jako sieci logicznej zwykłej operacji mnożenia, wszystkie funkcje wyjść (każda oddzielnie) są spełniane, więc nie stanowią problemu **SAT**. Dzieje się tak, ponieważ w operacji mnożenia, oba czynniki mogą przyjmować wartości z zakresu $\langle 1, 2^n-1 \rangle$, co sprawia, że dla każdej wartości iloczynu z przedziału $\langle 1, 2^{2n}-2 \cdot 2^n-1 \rangle$ istnieje przynajmniej jedna para czynników takiego iloczynu.

Z kolei, w przypadku rozpatrywania multiplikatora jako sieci logicznej mnożenia całkowitoliczbowego dwóch liczb pierwszych (funkcja $f_{mult}(p, q)$ jest różnowartościowa), każda funkcja wyjść multiplikatora będąc oddzielnie problem **FSAT** (i ***FNP-zupełnym***), niekoniecznie będzie funkcją spełnialną, więc może stanowić problem **SAT**.

Kwestie ***FNP***- i ***NP-zupełności*** będą wyglądały całkowicie odmiennie w przypadku sieci z jednym wyjściem, a wiemy (uwaga 7.1), że każda sieć z wieloma wyjściami może być sprowadzona przez sieć rozszerzającą C_{ext} do sieci z jednym wyjściem.

Jednak, przy sprowadzaniu sieci multiplikatora przy pomocy sieci rozszerzającej C_{ext} do sieci z jednym wyjściem pojawia się pewna trudność. Polega ona na tym, że ponieważ sieć multiplikatora dla określonej pary czynników oblicza z reguły inną wartość na każdym ze swoich $2 \cdot n$ wyjść, to obliczane dla tych czynników wartości na wyjściach sieci niekoniecznie wszystkie muszą przyjmować wartość *true*. Z tych powodów, sieć rozszerzająca C_{ext} nie może być siecią zbudowaną z samych bramek *AND*, tak jak to przedstawiono na rys. 7.8

Takie sytuacje ilustrują zamieszczone niżej przykłady mnożenia czynników o różnych wartościach (8·3), (6·4) i (15·7), których iloczyn dwóch pierwszych jest taki sam (w przykładach użyto zapisu binarnego).

czynniki	1000 x 0011	0110 x 0100	1111 x 0111
	---	---	---
Iloczyny cząstkowe	1000 1000 0000 0000	0000 0000 0110 0000	0111 0111 0111 0000
	---	---	---
Wyjścia sieci *C*	00011000	00011000	00110001

Jeśli ogólnie oznaczymy przez f_j, $j=0, 1, \ldots, 2n-1$ funkcje odpowiadające wyjściom multiplikatora (a w konsekwencji wyjściom sieci reprezentującej układ multiplikatora), to w zależności od kombinacji zmiennych na jego wejściach, traktowanych jako *n-bitowe* czynniki wynikowego iloczynu, będziemy mogli zapisać formułę, którą ta wejściowa kombinacja zmiennych spełnia. Zapisując formułę, funkcje odpowiadające wyjściom
z wartościami *false*, zapisujemy jako zanegowane. Nie zapisujemy *explicite* wyrażeń dla funkcji f_i określających wyjścia układu multiplikatora, bo są one „ukryte" w sieci reprezentującej układ multiplikatora.

Ostatecznie formuła dla pierwszego i drugiego przykładu przyjmuje postać $(\neg f_7) \wedge (\neg f_6) \wedge (\neg f_5) \wedge f_4 \wedge f_3 \wedge (\neg f_2) \wedge (\neg f_1) \wedge (\neg f_0)$ natomiast dla trzeciego: $(\neg f_7) \wedge (\neg f_6) \wedge f_5 \wedge f_4 \wedge (\neg f_3 \wedge) (\neg f_2) \wedge (\neg f_1) \wedge f_0)$

Uzyskane postaci korespondują z problemem badania spełnialności formuły będącej koniunkcją formuł różnych kombinacji zmiennych spośród $2 \cdot n$ zmiennych (uwzględniamy dwa *n-bitowe* czynniki).

Istotne jest także to, że w kategoriach sieci logicznych cały czas sieć C_{ext} rozszerzająca układ multiplikatora ma charakter sieci jednostajnej (wszystkie wejścia C_{ext} przyjmują wartość *true*). Ponadto sieć C multiplikatora z rozszerzeniem C_{ext} staje się siecią z jednym wyjściem. Tym samym przyjmuje postać, którą można wykorzystać jako sieć rozstrzygającą problem decyzyjny albo obliczeniowy (uwagi 7.2 i 7.3).

Zauważmy, że w rozpatrywanych przykładach, przy zapisywaniu formuł, funkcje f_i wyjść multiplikatora nie były negowane, gdy odpowiadały pozycjom wyjść multiplikatora o wartości *„1" (true)* ciągu zero-jedynkowego traktowanego jako liczba binarna.

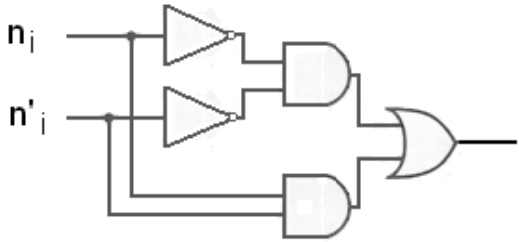
7.5.2 Konstrukcja sieci logicznej multiplikatora

Poczynione spostrzeżenie, nasuwa prosty sposób konstruowania sieci logicznej dla dowolnego przypadku problemu *mult (n)* traktowanego jako **SAT** lub **FSAT**.

Aby nie konstruować sieci C_{ext} z użyciem bramek *NOT* na wejściach określonych przez postać binarną iloczynu (w *mult(n)* postać binarna iloczynu jest znana) odrębnie dla różnych

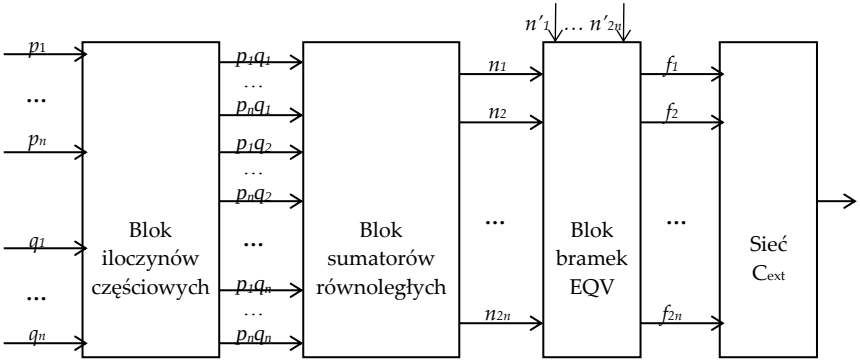
możliwych wartości, użyjemy na wszystkich wejściach sieci C_{ext} 2-wejściowe bramki realizujące funkcję *EQV* (funkcja równoważności). Sposób funkcjonowania *j-tej* bramki EQV opisanej w postaci tablicowej i jej schemat przedstawia rys. 7.10.

n_i'	n_i	EQV_i
0	0	1
0	1	0
1	0	0
1	1	1



Rys. 7.10 Tablica funkcji EQV i schemat bramki

Jeśli n_j będzie wartością *j-tego* wyjścia sieci *C* multiplikatora, a n_j' stałą logiczną określoną przez cyfrę *j-tej* pozycji iloczynu zapisanego binarnie, to w ten sposób na wejściach sieci C_{ext} zapewnimy negacje wartości *false* obliczanych na wyjściach sieci *C*, zaś obliczone wartości *true* na wyjściach sieci *C* pozostaną niezmienione. Ideowy schemat funkcjonalny sieci rozpatrywanego multiplikatora przedstawia rysunek 7.11.



Rys. 7.11 Schemat ideowy sieci multiplikatora

Tak skonstruowana sieć rozszerzająca C_{ext} zachowuje rozmiar wielomianowy całego multiplikatora i jest odpowiednikiem schematu układu kombinacyjnego ze zmiennymi i stałymi na wejściach (rys. 7.7 c). Ponadto charakteryzuje się własnością jednostajności.

Uwaga 7.5 Jeśli każde *j-te (j=1, 2, …, 2n)* wyjście sieci multiplikatora będzie połączone z oboma wejściami *j-tej* bramki *EQV* sieci C_{ext} oraz na wejściach multiplikatora jako czynniki przyjęte zostaną liczby 2^n*-1*, których wszystkie pozycje postaci binarnej traktowane jako zmienne mają wartość *true*, to w ten prosty sposób otrzymujemy sieć spełniającą wymogi definicji wielomianowej sieci jednostajnej.

7.5.3 Redukcja sieci logicznej multiplikatora

Rozpatrywany model multiplikatora scharakteryzowaliśmy wcześniej jako wielowyjściowy układ kombinacyjny. Po pierwsze, dla konkretnego modelu multiplikatora dysponujemy schematem połączeń bramek *AND*, *OR* i *NOT*. Po drugie, każde *j-te* wyjście odpowiadające *j-tej* pozycji iloczynu można traktować jako niezależne i każde można opisać funkcją przełączającą.

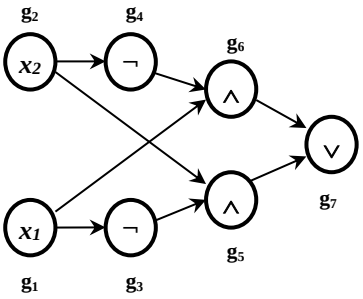
Zatem, układ multiplikatora można traktować jako sieć logiczną: - raz jako sieć bez zmiennych **CIRCUIT VALUE**, jeśli na wejściach układu czynnikami są konkretne liczby zapisane binarnie (co odpowiada schematowi wielobiegunnika z rys. 7.7a); - dwa jako sieć ze zmiennymi **CIRCUIT SAT** , gdy wartość czynników na wejściach określana jest ogólnie poprzez zmienne (odpowiednik schematu wielobiegunnika z rys. 7.7 b), a także jako sieć **CIRCUIT FSAT**, gdy wejścia dodatkowo określane są przez stałe (odpowiednik schematu wielobiegunnika z rys. 7.7 c).

W obu przypadkach, w oparciu o fakt 7.5, każda funkcja logiczna (tu odpowiednik funkcji przełączających f_j wyjść multiplikatora) może być odwzorowana w czasie wielomianowym $\boldsymbol{O}(n^2)$ i w pamięci logarytmicznej *log n* przez wyrażenie mające postać *SAT CNF*. Na łączny czas wielomianowy redukcji składa się liniowy $\boldsymbol{O}(n)$ czas redukcji sieci pojedynczego sumatora liczb *n-bitowych*, a takich sumatorów w układzie równoległym multiplikatora jest *n-1*. O tym, że taką redukcję można wykonać w pamięci logarytmicznej decyduje jednorodna budowa i struktura połączeń bramek układu sumatorów i całego multiplikatora, które sprowadzają prowadzenie procesu redukcji do operowania na indeksach.

Ponieważ takie redukcje trzeba wykonać dla wszystkich *2n* funkcji wyjść rozpatrywanego modelu multiplikatora, to całkowity czas redukcji wyraża się czasem wielomianowym nie gorszym $\boldsymbol{O}(n^3)$. Przy szacowaniu przyjęliśmy *2n* wyjść

multiplikatora i ilość n^2 bramek *AND, OR i NOT* całego układu multiplikatora.

Prosty przykład takiego odwzorowania ilustruje redukcja sieci bramki *EXOR* (rys. 7.12).



$$SAT\ CNF = (g_7 \vee \bar{g}_6)(g_7 \vee \bar{g}_5)(g_5 \vee g_6 \vee \bar{g}_7)$$
$$\wedge(\bar{g}_6 \vee g_4)(\bar{g}_6 \vee g_1)(g_6 \vee \bar{g}_4 \vee \bar{g}_1)(\bar{g}_5 \vee g_3)$$
$$\wedge(\bar{g}_5 \vee g_2)(g_5 \vee \bar{g}_3 \vee \bar{g}_2)$$
$$\wedge(\bar{g}_4 \vee \bar{g}_2)(g_4 \vee g_2)(\bar{g}_3 \vee \bar{g}_1)(g_3 \vee g_1)$$
$$\wedge(g_1 \vee \bar{x}_1)(\bar{g}_1 \vee x_1)(g_2 \vee \bar{x}_2)(\bar{g}_2 \vee x_2)$$

Rys.7.12 Przykład redukcji sieci bramki EXOR do wyrażenia SAT CNF

Możliwa wielomianowa redukcja sieci logicznej rozpatrywanego multiplikatora w zasadzie rozstrzyga ***FNP-*** i ***NP -*** *zupełność* problemu *mult(n)*. Wystarczyłoby powołać się na dyskutowany wcześniej dualizm sieci logicznych i możliwość konstruowania sieci rozszerzającej C_{ext} w przypadku sieci z wieloma wyjściami.

Istnieją jednak powody, by uzasadnienie ***FNP-*** i ***NP*** *-zupełności* problemu *mult(n)* pogłębić. Po pierwsze dlatego, że chociaż rozpatrywany model multiplikatora i jego sieci może być przy odpowiednich założeniach wystarczający (np. poprzez dopuszczenie uzupełniania i późniejszej analizy nieznaczących zer w iloczynie i obu czynnikach), to w istocie powinniśmy posługiwać się rodziną multiplikatorów. Po drugie, wobec pewnych związków problemu **FACTORING** z problemem **PRIMES**, pogłębiona dyskusja może powiązać zupełność *mult(n)* z zupełnością problemu **PRIMES**.

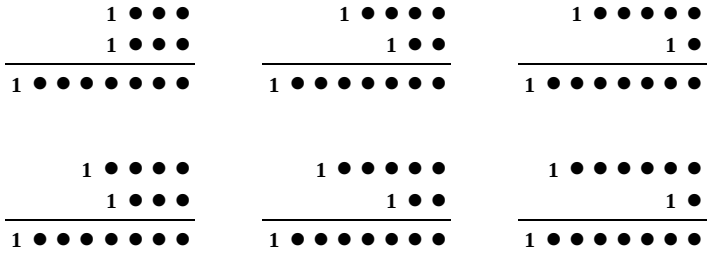
7.6 Rodzina multiplikatorów

7.6.1 Rozmiar sieci rodziny multiplikatorów

Stwierdziliśmy, że w kontekście problemu *mult(n)*, gdy znany jest *2n-bitowy* iloczyn, pojedyncza sieć multiplikatora rozpatry-

wanego do tej pory (tj. z oboma *n-bitowymi* czynnikami) może być niewystarczająca. Wygodniej będzie, jeśli posłużymy się rodziną sieci multiplikatorów. Wynika to z faktu, że jeśli dwie liczby całkowite *a* i *b* zapisane binarnie mają długości w_1 i w_2, to długość ich iloczynu zapisanego binarnie jest równa $w=w_1+w_2$ lub $w=w_1+w_2-1$.

Zatem, ponieważ nasz problem *mult(n)* polega na ustaleniu czynników danego iloczynu o długości *2n*, aby uwzględnić możliwe zróżnicowanie długości czynników tego iloczynu, koniecznym jest rozpatrywanie możliwych 2(*n-1)* przypadków. Na przykład, rys. 7.13 ilustruje wszystkie możliwe przypadki dla *8-bitowego* iloczynu, z pominięciem trywialnego, gdy drugi czynnik jest równy 1.



Rys. 7.13 Przykładowe możliwe przypadki czynników iloczynu *8-bitowego*

Każdemu z możliwych 2(*n-1)* przypadków iloczynu czynników o długości *2n* (z najbardziej znaczącym bitem równym *1*), odpowiada stosowny układ multiplikatora. Spośród wszystkich *2(n-1)* multiplikatorów wyodrębnimy grupy. Pierwszą grupę będą stanowiły multiplikatory M'_i ($i = n, n-1, \ldots, 2$), w których długość drugiego czynnika jest równa *i*, natomiast długość pierwszego określona jest przez *2n-i*. Z kolei drugą grupę będą stanowiły multiplikatory M''_i, w których długość drugiego czynnika jest także równa *i*, natomiast długość pierwszego określona jest przez *2n+1-i*. Multiplikatory M'_i *i* M''_i będą się różnić więc tylko długością pierwszego czynnika.

W każdym multiplikatorze obu grup, sumowanie iloczynów częścio-wych realizowane jest przez sumatory w układzie równoległym, przy czym począwszy od pary M'_n *i* M''_n do M'_2 *i* M''_2 w co drugiej ich ilość będzie sukcesywnie malała od *n-1* do *1* (oba multiplikatory M'_2 i M''_2 wykorzystują jeden sumator). Podobnie jest w odniesieniu do ilości bramek *AND* ustalających

iloczyny częściowe. Począwszy od pary M'_n *i* M''_n do M'_2 *i* M''_2 ilość bramek *AND* maleje od wielomianowej do liniowej.

Ponieważ multiplikatory M'_i i M''_i $(i=n, n-1, \ldots, 2)$ parami różnią się tylko długością pierwszego czynnika, co nie wpływa na charakter zależności opisującej ich rozmiar wyrażony ilości bramek. Tak więc, w szacowaniu rozmiaru sieci logicznych układów całej rodziny multiplikatorów można ograniczyć się do dyskusji dowolnej grupy.

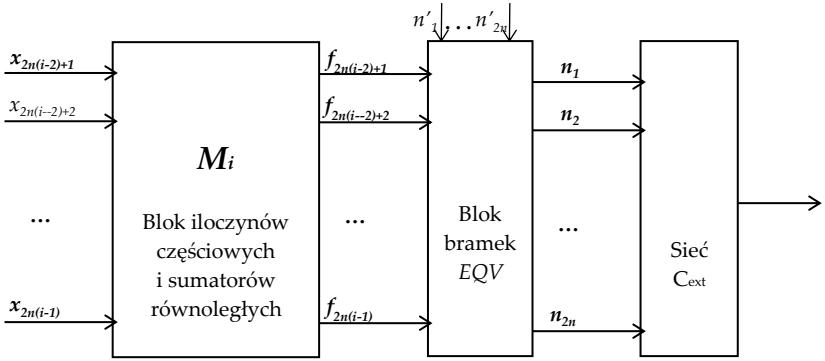
Rozmiar sieci logicznej układu multiplikatora o największej ilości sumatorów oraz ilości bramek *AND* do ustalenia iloczynów częściowych, odpowiadający układowi multiplikatora M'_n oszacowaliśmy wcześniej (punkt 7.3.2) - wyrażał się on wielomianem drugiego stopnia. Ustaliliśmy także tam, że przy rozmiarze sieci multiplikatora określonym przez wielomian drugiego stopnia, ale przy konieczności uwzględniania wszystkich jego $2n$ wyjść, redukcja tej sieci do **SAT** wymaga czasu wielomianowego $\boldsymbol{O}(n^3)$. Tak więc, wobec malejącego rozmiaru sieci logicznych kolejnych par multiplikatorów M'_i oraz M''_i, wymagany czas redukcji każdej z tych sieci do **SAT** nie będzie gorszy niż $\boldsymbol{O}(n^3)$ (w przypadku sieci pary multiplikatorów M'_2 oraz M''_2 będzie liniowy).

Ostatecznie, w odniesieniu do całej rodziny $2(n-1)$ multiplikatorów, całkowity czas redukcji ich sieci logicznych do **SAT** będzie charakter-ryzował się wielomianem czwartego stopnia $\boldsymbol{O}(n^4)$.

W oszacowaniu czasu redukcji uwzględniono: (i) po pierwsze, że sprzętowa realizacja każdego multiplikatora rodziny ma postać układu kombinacyjnego. Tak jak w modelu multiplikatora (pkt 7.3.2), każdy multiplikator rodziny oddzielnie cechuje regularna budowa i sposób połączeń jego bramek. Wyjścia tych multiplikatorów można opisać przy pomocy funkcji przełączających, każdorazowo innych $2n$ lub $2n+1$ zmiennych wejściowych; (ii) po drugie, że w przypadku dyskutowanego wcześniej modelu multiplikatora wystarczyło posłużyć się liniową liczbą zmiennych. Natomiast w przypadku rozpatrywanej rodziny multiplika-torów, dla opisania funkcji wyjść wszystkich multiplikatorów musimy posłużyć się wielomianową równą $(4n+1)(n-1)$ ilością zmiennych. Jednak w odniesieniu do każdego multiplikatora oddzielnie, ich wyjścia będą opisywane przez funkcje przełączające liniowej równej

$2n$ ilości zmiennych; (iii) po trzecie, rozszerzenie sieci każdego multiplikatora rodziny o sieć C_{ext} z bramkami *EQV* (patrz schemat ideowy – rys. 7.11) sprawiające , że sieć ma charakter **CIRCUIT SAT**, nie wpływa na oszacowanie rozmiaru sieci i czasu redukcji sieci poszczególnych multiplikatorów i całej rodziny.

W przykładowym schemacie ideowym multiplikatora M'_i (rys. 7.14), zmienne $x_{2n(i-2)+1}, \ldots, x_{2n(i-2)+2n-i}$ na wejściach multiplikatora, mogą być interpretowane jako pozycje binarne pierwszego *(2n-i)-bitowego* czynnika, a pozostałe $x_{2n(i-2)+2n-i+1}, \ldots, x_{2n(i-1)}$ jako pozycje binarne drugiego *i-bitowego* czynnika iloczynu.



Rys. 7.14 Schemat ideowy multiplikatora M_i rodziny multiplikatorów.

Sieci rozszerzające C_{ext} z bramkami *EQV*, na wejścia której podawane są wyjścia multiplikatorów oraz wartości stałych $n'_1, \ldots, n'_{2n}$ interpretowane jako binarna postać weryfikowanego iloczynu, sprawiają, że uzyskujemy 2(n-1) schematów problemu decyzyjnego.

7.6.2 Zupełność problemów *mult(n)* i *primes*

Dyskutując operację mnożenia, zwróciliśmy uwagę, że mnożenie charakteryzuje się swoistym dualizmem. Będąc problemem oblicze-niowym, może być także postrzegane jako problem decyzyjny. Przekłada się to na dualizm multiplikatora jako układu wykonawczego i jego sieci logiczne. Wielomianowa sieć układu multiplikatora może być rozpatry-wana raz jako problem **CIRCUIT FSAT**, a z siecią rozszerzającą C_{ext} jako **CIRCUIT SAT**.

Dualizm w szerszym ujęciu jest przedmiotem badań w teorii złożoności. Znajduje to odbicie w rozpatrywaniu dualnych klas

złożoności, np. klasy ***coNP*** jako dopełnienia klasy ***NP***[2]. Jeśli więc w klasie ***NP***, w odniesieniu do problemów decyzyjnych, interesują nas potwierdzenia czy problemy klasyfikowane do niej mają rozwiązanie (odpowiedzi *«tak»*), to w klasie ***coNP*** sytuujemy problemy w stosunku do których interesują nas potwierdzenia braku rozwiązania (odpowiedzi *«nie»*). Inaczej można to sformułować następująco: *„Problemy należące do klasy NP mają zwięzłe dowody, podczas gdy dla problemów z coNP istnieją szybkie sposoby dyskwalifikacji."* [2 s. 239].

Stosownie do powyższych określeń, klasę ***coNP*** tworzą problemy będące dopełnieniem problemów z klasy ***NP***, w tym reprezentatywne problemy **SAT COMPLEMENT** (definiowany przez zanegowanie formuły **SAT** i tym samym równoważny problemowi **VALIDITY** –prawdziwość), bądź problem **HAMILTON PATH COMPLEMENT**. Oba wymienione problemy są dodatkowo przykładami problemów ***coNP-zupełnych***[3] (wiemy, że jeśli problem ***A*** jest ***NP-zupełny***, to jego dopełnienie jest ***coNP-zupełne*** [2 s. 238].

Innym ważnym problemem zaliczanym do ***coNP***, jest problem **PRIMES**. Pytamy w nim, czy *n-bitowa* liczba jest liczbą pierwszą. Jego dualną wersją jest zatem problem, w którym pytamy, czy dana *n-bitowa* liczba nie jest liczbą pierwszą. Pozytywna odpowiedź w tym przypadku, tzn. potwierdzenie, że dana liczba nie jest liczbą pierwszą, jest tym samym stwierdzeniem, że jest to liczba złożona. To z kolei jest równoważne odpowiedzi pozytywnej w decyzyjnym problemie **FACTORING**, gdy chcemy wiedzieć czy dana *n-bitowa* liczba ma czynniki pierwsze (bez określania ich wartości, co jest celem problemu **FACTORING** zaliczanego w wersji obliczeniowej do klasy ***FNP***).

Więc jeśli dla danej liczby *n-bitowej* w problemie decyzyjnym **FACTORING** rolę zwięzłego dowodu pełni wskazanie przynajmniej jednego czynnika (niekoniecznie pierwszego) różnego od 1, to w problemie **PRIMES** dla tej samej danej liczby *n-bitowej*, rolę dowodu

[2] Badanie relacji klasy ***coNP*** z innymi klasami, a w szczególności z klasą ***NP*** jest obok kluczowej kwestii ***P versus NP***, kolejnym istotnym pytaniem stawianym w teorii złożoności.

[3] Tak jak w klasie ***NP*** wyodrębniono problemy ***NP-zupełne***, tak w klasie ***coNP*** również wyodrębnia się problemy ***coNP-zupełne*** (tj. takie, które dają się wzajemnie odwzorowywać przy pomocy redukcji wielomianowej).

dyskwalifikującego może spełniać wskazanie tego samego czynnika.

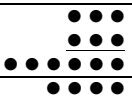
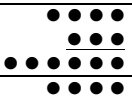
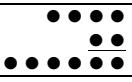
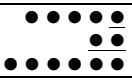
Opisany tak sposób rozstrzygania obu problemów można uzyskać wykorzystując dyskutowaną w poprzednim punkcie rodzinę multiplikatorów.

Każdy multiplikator rodziny, gdy rozpatrywany jest ze stałymi na swoich wejściach bez sieci rozszerzającej C_{ext}, po redukcji jego sieci jest problemem **CIRCUIT VALUE** – jego wyjścia określają pozycje binarne iloczynu czynników o ustalonych długościach. Jednak, gdy będzie rozpatrywany ze zmiennymi na wejściach oraz z siecią rozszerzającą C_{ext} i jej bramkami *EQV* ze stałymi n_i reprezentującymi pozycje danej liczby N (o której chcemy wiedzieć czy jest pierwsza, czy nie jest pierwsza), po redukcji jego sieci reprezentuje problem **SAT**.

Spełnialność formuł *SAT* M'_2 i M''_2 uzyskanych w wyniku redukcji sieci logicznych każdego multiplikatora rodziny rozstrzyga, czy dla danej liczby $N=(n_{2n}, n_{2n-1}, \ldots, n_2, n_1)$ istnieją czynniki zdeterminowane długością wejściowych słów. Na przykład, multiplikatory M'_2 i M''_2 pozwalają stwierdzić, czy liczba *3* jest dzielnikiem danej liczby N. Z kolei, multiplikator M'_3 pozwala stwierdzić, czy podzielnikiem danej liczby N są liczby *5* i *7*.

Wykorzystanie rodziny multiplikatorów do rozstrzygania czy dana liczba jest pierwsza, czy nie jest, można zilustrować przykładem badania liczb *6-bitowych* z zakresu *32-63* (tabela 7.1 poniżej).

Tabela 7.1 Spełnialność formuły ***SAT*** M'_i oraz M''_i

schemat multiplikatora		**spełnialność formuł *SAT*** M'_i i M''_i				
		N=35	***N=45***	***N=47***	***N=54***	***N=55***
M'_3		***tak (7·5)***	***nie***	***nie***	***nie***	***nie***
M''_3		***nie***	***tak (9·5)***	***nie***	***tak (9·6)***	***tak***
M'_2		***nie***	***tak (15·5)***	***nie***	***nie***	***nie***
M''_2		***nie***	***nie***	***nie***	***tak (27·2),(18·3)***	***nie***

Z analizy tabeli 7.1 jednoznacznie wynika, że dla problemu, czy dana liczba *N* nie jest pierwsza, odpowiedzią potwierdzającą jest spełnialność przynajmniej jednej formuły *SAT* M_i lub M''_i całej rodziny multipoli-katorów. Z kolei, dla problemu, czy liczba jest pierwsza, odpowiedzią potwierdzająca jest brak spełnialności wszystkich formuł całej rodziny multiplikatorów.

Interpretację rozstrzygania złożoności bądź pierwszości liczb $N \in <32,63>$ można uogólnić na dowolny zakres liczb $N \in <2^{2n}, 2^{2n+1}-1>$. Ponadto, dyskusja potwierdzania złożoności bądź pierwszości zawartych tam przykładów, nasuwa prosty sposób skonstruowania problemu **SAT**$_{MULT(n)}$ i **SAT**$_{PRIMES}$. Wystarczy by wyjścia sieci rozszerzeń C_{ext} każdego multiplikatora rodziny stanowiły wejścia bramek *OR* w układzie szeregowym lub równoległym.

W obu przypadkach, wejściami bramek *OR* są wyjścia bramek sieci C_{ext} poszczególnych multiplikatorów całej rodziny M'_i i M''_i, $i=2, \ldots, n$ i każdemu odpowiada określona formuła *SAT* M'_i lub M''_i.

BIBLIOGRAFIA

[1] Harel David, Rzecz o istocie informatyki. Algorytmika, Warszawa, WNT, 1992

[2] Papadimitriou Christos, Złożoność obliczeniowa, Warszawa, WNT, 2007

[3] Traczyk Wiesław, Układy cyfrowe. Podstawy teoretyczne i metody syntezy, Warszawa, WNT, 1986