Rozwiązywanie równań i układów równań nad skończonymi strukturami algebraicznymi z wykorzystaniem SAT-solvera.

Patryk Popek

Praca Licencjacka

Uniwersytet Marie Curie Skłodowskiej w Lublinie
Instytut Informatyki
2023

Spis treści

Wstęp

1 Opis Teoretyczny Problemu

Poniżej przedstawione są pojęcia związanie z pracą.

1.1 Równania i ich rozwiązywalność

1. **Struktura algebraiczna**: Jest to dwójka (A, B), gdzie A jest dziedziną algebry, B jest dozwolonymi operacjami na zbiorze.

2. **Funkcja:** Jest to operacja przekształcająca zbiór X w zbiór Y. Innymi słowami, każdemu elementowi $x \in X$ przypisuje dokładnie jeden element $y \in Y$. Zbiór X nazywamy dziedziną funkcji, a zbiór Y przeciwdziedziną. Najpopularniejszą formą przedstawienia funkcji jest podanie jej wzoru na przykład $f_x = x + 2$

3. **Arność:** Jest to liczba argumentów przyjmowana przez funkcję. Jeżeli arność wynosi 0 wtedy funkcja jest funkcją stałą

4. **Liczność(eng. Cardinality):** Jest to liczba elementów znajdujących się w zbiorze. Na potrzeby tej pracy określa dziedzinę wczytanej do aplikacji struktury. W dalszej części pracy będę to oznaczał jako 'C'

5. **Punkt przecięcia funkcji:** Mamy dwie funkcje F_1 oraz F_2. Punkt przecięcia oznacza miejsce, w którym $F_1 = F_2$. Posiadając wzory powyższych funkcji możemy ułożyć równanie, za pomocą którego umiemy wyliczyć wartość powyższego punktu.

6. **Równanie:** Jest to wyrażenie postaci $t_1 = t_2$, gdzie t_1, t_2 są dowolnymi wyrażeniami algebraicznymi zgodnymi z używaną algebrą. Równania dzielimy na 3 typy:

   a) Sprzeczne: Równanie nie posiada rozwiązania, czyli za zmienną równania nie da się podstawić żadnej wartości z dziedziny algebry.

   b) Oznaczone: Równanie posiada rozwiązania, czyli za zmienną równania da się podstawić co najmniej jedną wartość z dziedziny algebry.

   c) Tożsame: Równanie jest zawsze rozwiązywalne bez względu na to jaką wartość z dziedziny algebry podstawimy za zmienną równania.

7. **Notacja Polska:** Sposób zapisu wyrażeń, w którym operator/nazwa funkcji znajduję się przed argumentami. Na przykład $/ + xy - ab$ jest równoważne zapisowi $(x + y)/(a + b)$

1.2 Sat i spełnialność

Poniższe definicje odnoszą się do logiki zerowego rzędu.

1. **Zmienna zdaniowa:** jest to najmniejszy możliwy element w logice. Przyporządkowuje się jej wartość prawda(1) albo fałsz(0).

2. **Negacja:** Jest to zaprzeczenie zmiennej zdaniowej $x = 1$ $\neg x = 0$.

3. **Literał:** Jest to pojedyncze wystąpienie zmiennej x $\neg x$.

4. **Spełnialność funkcji:** Funkcja logiczna jest spełnialna gdy istnieje wartościowanie, które zwraca wartość logiczną funkcji 1.

5. **Prawdziwość funkcji:** Funkcja logiczna jest spełnialna przy każdym wartościowaniu, czyli zawsze zwraca wartość logiczną funkcji 1.

6. **Koniunkcja:** Funkcja logiczna postaci $x \wedge y$, zwraca 1 tylko i wyłącznie wtedy gdy zarówno x,y=1.

7. **Alternatywa:** Funkcja logiczna postaci $x \vee y$, zwraca 0 tylko i wyłącznie wtedy gdy zarówno x,y=0.

8. **Implikacja:** Funkcja logiczna postaci $x \rightarrow y$, zwraca 0 tylko i wyłącznie wtedy gdy x=1,y=0.

9. **Koniunkcyjna postać normalna (CNF):** Funkcja logiczna, która jest koniunkcją alternatyw.

10. **Dysjunkcyjna postać normalna (DNF):** Funkcja logiczna, która jest alternatywą koniunkcji.

11. **Klauzula Horna:** Jest to alternatywa posiadająca co najwyżej jeden literał niezanegowany.

2 Redukcja do Sata

Głównym problemem związanym z moją pracą, był sposób przedstawienia równania matematycznego za pomocą formuły logicznej zdolnej do przetworzenia przez Sat-Solver. Finalna formuła, którą chciałem uzyskać, musiała być w postaci CNF.

2.1 Pliki .cnf

Wykonana przeze mnie redukcja nie tylko musiała być przedstawiona w odpowiednim typie formuły logicznej, ale też musiała mieć format akceptowany przez używany Sat-Solver. Takim formatem są pliki z rozszerzeniem .CNF [1] .

```
c 1=x_0 2=x_1 3=&0_0 4=&0_1
c 5=y_0 6=y_1 7=&1_0 8=&1_1
p cnf 8 16
1 2 0
-1 -2 0
3 4 0
-3 -4 0
-1 4 0
-2 3 0
5 6 0
-5 -6 0
7 8 0
-7 -8 0
-5 7 0
-6 8 0
-3 7 0
3 -7 0
-4 8 0
4 -8 0
```

Rysunek 1: Przykładowy plik reprezentujący redukcje CNF

Powyższy plik powstał poprzez redukcję wyrażenia $xor(1, x) = xor(y, 0)$. Pierwsze dwie linie, zaczynające się od znaku 'c' są komentarzami i nie są

rozpatrywane przez program. W tym przypadku są one legendą informującą o indeksach użytych zmiennych w czasie redukcji. Linia trzecia, czyli nie licząc komentarzy pierwsza, informuje o strukturze całego pliku. 'p cnf' oznacza, że plik przedstawia formułę w postaci CNF. Kolejne dwie liczby oznaczają odpowiednio ilość zmiennych oraz liczbę klauzul rozpatrywanej formuły. Dalsza część pliku jest zarezerwowana dla klauzul. '0' jest specjalnym symbolem oznaczającym koniec pojedynczej klauzuli, dlatego wszystkie zmienne indeksujemy od jedynki. Jeżeli indeks reprezentujący zmienną jest ujemny, to oznacza że ta zmienna w klauzuli jest zanegowana.

2.2 Tworzenie zmiennych

Na potrzeby redukcji musiałem stworzyć zmienne, na których podstawie mógłbym wyciągnąć wnioski po zakończeniu pracy Solvera. Zmienne zdaniowe mają postać x_n lub $\&_{m,n}$, gdzie $n \in \{0, 1..., C-1\}$ i $m \in \{0, 1..., k-1\}$ gdzie k jest liczbą operacji zdefiniowanych w algebrze, użytych w całym równaniu. x_n reprezentuje jakąkolwiek niewiadomą równania. Rozbicie powtarzamy wszystkich szukanych równania. $\&_{m,n}$ reprezentuje wynik zwrócony poprzez wykonanie pojedynczej funkcji. Wartość true zmiennej x_n oznacza, że dla wartości n niewiadomej x równanie ma rozwiązanie. Dlatego musimy wymusić, aby tylko tylko jedna zmienna z zakresu x_0 do x_n mogła mieć wartość true. Bez tego warunku zmienna mogłaby mieć kilka wartości na raz.

2.3 Klauzule użyte w redukcji

1. **Klauzula zapewniająca co najmniej jedną zmienną:**

   Pierwszym krokiem jest zapewnienie, aby pojedyncza zmienna na pewno miała jakąkolwiek wartość. Efekt ten uzyskujemy poprzez klauzule: $(x_0 \vee x_1 \vee ... \vee x_{C-1})$ gdzie x to dowolna niewiadoma równania. Aby powyższa formuła była spełniona, co najmniej jedna zmienna musi być prawdziwa.

2. **Klauzula zapewniająca co najwyżej jedną zmienną:**

   Punkt pierwszy nie zapewniał limitu górnego aktywnym zmiennych. W tym celu wprowadzamy dodatkowe klauzule: $(\neg x_a \vee \neg x_b)$ gdzie $a \neq b$ oraz $a, b \in \{0, 1, ..., C-1\}$. Przez co, jeżeli dwie zmienne będą miały wartość true to klauzula będzie nie spełniona czyli też cały Sat. Dlatego wymusza to pojedynczą wartość true dla całego zbioru zmiennych.

3. **Główna część redukcji:**

Posiadając odpowiednią liczbę zmiennych możemy przejść do głównej części redukcji. Polega ona na szukaniu wartości działania w tablicy wynikowej operatora wczytanej wraz z algebrą. Załóżmy, że mamy działanie d o arności 2 oraz, że część równania, które rozpatrujemy to d(x,y). Załóżmy dalej, że to działanie dla wartości 3 i 2 zwraca wartość 0. Nasza klauzula będzie miała wtedy wartość: $((x_3 \wedge y_2) \rightarrow \&_{0,0})$. Oznacza to, że z faktu iż argumentami funkcji są 3 i 2 to wynik wynosi 0. Postać CNF to: $(\neg x_3 \vee \neg y_2 \vee \&_{0,0})$. Jest to klauzula horna. Jej postać ogólna to : $(\neg x_a \vee ... \vee \neg y_b \vee \&_{m,n})$ gdzie $a, b, n \in \{0, 1..., C-1\}$ i $m \in \{0, 1..., k-1\}$. W powyższej formule zamieszcza się wszyskie niewiadome pojedynczego rozpatrywanego działania oraz jego symbol wynikowy, czyli długość takiej klauzuli wynosi co najwyżej arity+1. Jeżeli funkcja posiada stałe, to nie zamieszczamy ich w klauzuli. Klauzule są tworzone rekurencyjnie. Ilość klauzul stworzonych na podstawie jednej operacji to d^C gdzie d to ilość niewiadomych w funkcji. Jeżeli funkcja ma same stałe to wtedy wylicza się jej wartość i ma ona postać $\&_{x,y}$ gdzie x to index zmiennej a y to wyliczona wartość

4. **Przyrównanie stron**

   Równanie dzielimy na dwie części. Osobno rozpatrujemy lewą i prawą część. Dlatego jako ostatni krok musimy rozpatrzeć równowartość obu tych stron. Klauzula będzie miała postać $((\&_{a,n} \vee \neg\&_{b,n}) \wedge (\neg\&_{a,n} \vee \&_{b,n}))$ gdzie a, b są indeksami najbardziej zewnętrznych funkcji obu stron równań i $n \in \{0, 1..., C-1\}$

Jeżeli dowolna niewiadoma występuje w układzie więcej niż rat, to traktujemy ją jako tą samą zmienną. Wymuszenie pojedynczej wartości true przeprowadzamy na niej raz. Redukcja ta działa identycznie zarówno dla zwykłego równania jak i dla układu. Dla układów występuje więcej klauzul typu czwartego

3 Aplikacja i algorytmy w niej zastosowane

Aplikacja, którą napisałem na potrzeby tej pracy podzielona została na kilka kroków. Etapy te zostały ułożone liniowo i każdy kolejny można wykonać dopiero po ukończeniu wszystkich wcześniejszych. Wszystkie etapy są reprezentowane w interfejsie aplikacji za pomocą przycisków.

3.1 Wczytanie algebry do programu

3.1.1 Pliki .ua

Algebry, które są wykorzystywane w mojej aplikacji są reprezentowane poprzez pliki z rozszerzeniem "ua". Skrót ten pochodzi od wyrażenia "universal algebra". Są to pliki w formacie XML.

```
<?xml version='1.0' encoding='utf-8'?>
<algebra>
  <basicAlgebra>
    <algName>P4</algName>
    <cardinality>4</cardinality>
    <operations>
      <op>
        <opSymbol>
          <opName>d</opName>
          <arity>2</arity>
        </opSymbol>
        <opTable>
          <intArray>
            <row r="[0]">0,1,2,3</row>
            <row r="[1]">1,0,3,2</row>
            <row r="[2]">2,3,0,1</row>
            <row r="[3]">3,2,1,0</row>
          </intArray>
        </opTable>
      </op>
      <op>
        <opSymbol>
          <opName>m</opName>
          <arity>2</arity>
        </opSymbol>
        <opTable>
          <intArray>
            <row r="[0]">0,0,0,0</row>
            <row r="[1]">0,1,2,3</row>
            <row r="[2]">0,2,0,2</row>
            <row r="[3]">0,3,2,1</row>
          </intArray>
        </opTable>
      </op>
    </operations>
  </basicAlgebra>
</algebra>
```

Rysunek 2: Przykładowy plik reprezentujący algebrę uniwersalną

Powyższe zdjęcie przedstawia wygląd pliku opisującego algebrę. Sekcjami interesującymi nas najbardziej w tym pliku są cardinality oraz operations. Cardinality informuje o dziedzinie całej struktury. Jest ona niezmienna i ograniczona do zakresu $x \in \{0, 1, ..., cardinality - 1\} \wedge x \in \mathbb{N}$ Operations opisuje wszystkie dostępne operacje na strukturze. Sekcja ta jest podzielona na podsekcje op poświęcone poszczególnym działaniom. OpName informuje o nazwie działania, jest to identyfikator, po którym będzie można znaleźć działanie w programie. Arity określa arność funkcji reprezentowanej przez to działanie. Najważniejszą częścią opisu działania jest opTable. Jest to struktura przedstawiająca wyniki działania opisywanej funkcji na wszystkich możliwych wariancjach z powtórzeniami o długości arity dziedziny algebry. Tabele te są główną osią, dzięki której powstaje finalna formuła CNF używana w SatSolverze. Algebra pokazana stronę wyżej posiada dwa działania: Działanie d, które jest dwuargumentowym dodawaniem modulo 4 oraz działanie m, które jest dwuargumentowym mnożeniem modulo 4.

3.1.2 Algorytm w programie

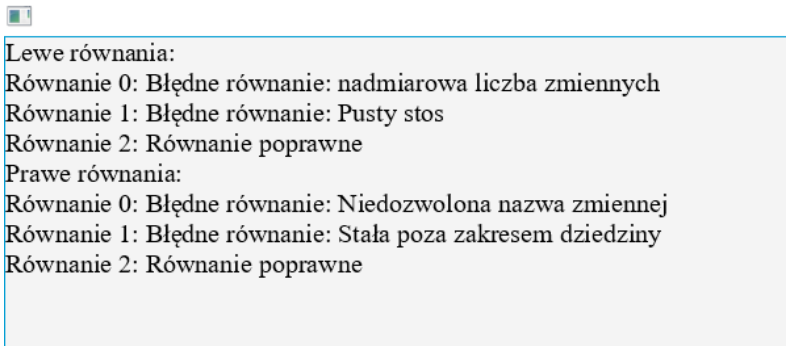
1. Wczytaj plik do programu.
2. Zapisz w programie nazwę oraz dziedzinę algebry.
3. Pobierz listę operacji.
4. Przejdź do kolejnej niezapisanej operacji.
5. Zapisz arność i nazwę operacji.
6. Zapisz tablicę wartości funkcji.
7. Jeżeli pozostała jakaś niezapisana operacja wróć do punktu 4.

3.2 Format i sprawdzenie poprawności równania

Równania, na których pracuje moja aplikacja są przyjmowane w formie funkcyjnej czyli $f_{(x,y)}$. Dzięki temu łatwo można zamienić powyższy zapis na notację polską, dla której znamy algorytm sprawdzania poprawności podanego wyrażenia, dla określonej dziedziny. Zamiana ta polega na usunięciu nawiasów i przecinków z podanego stringa. Jedno równianie jest podawane w dwóch częściach: jako lewa strona i prawa strona.

3.2.1 Algorytm sprawdzania poprawności wyrażenia w notacji polskiej

1. Przejdź do ostatniego znaku równania.
2. Pobierz kolejny znak równania, jeżeli jest dostępny.
3. Sprawdź, czy rozpatrywany znak jest nazwą dozwolonej operacji.
4. Jeżeli nie, sprawdź czy jest dozwoloną stałą lub zmienną.
5. Jeżeli nie przejdź do kroku 11.
6. jeżeli znak został określony jako operacja ściągnij ze stosu ilość znaków równej arności operacji i dodaj na stos znak wynikowy.
7. Jeżeli na stosie zabrakło symboli lub nastąpił ich nadmiar przejdź do kroku 11.
8. Jeżeli znak został określony jako stała lub zmienna dodaj go do stosu.
9. Jeżeli pozostały nierozpatrzone znaki przejdź do kroku 2
10. Jeżeli nie ma już nierozpatrzonych znaków równania, zakończ działanie i zwróć prawdę
11. Zwróć fałsz



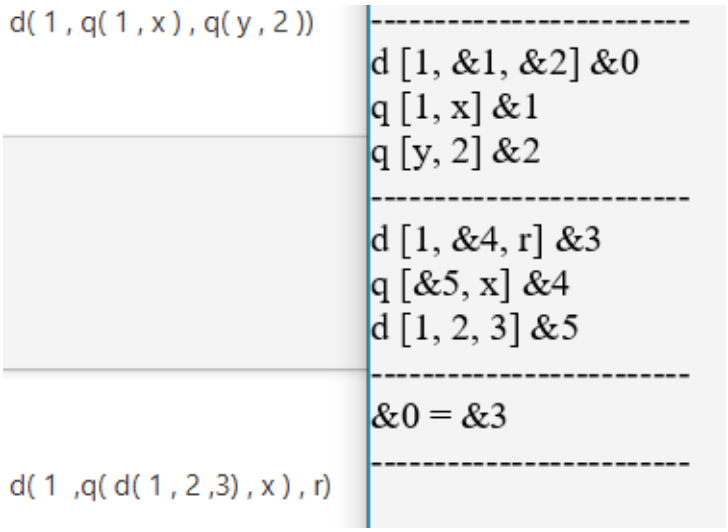
Rysunek 3: Przykładowe wyjście algorytmu

Algorytm sprawdzający zwraca 5 różnych wynikowych komunikatów dotyczących poprawności równania względem wczytanej algebry. Pierwszy komunikat od góry występuje, gdy w momencie napotkania operacji ilość elementów na stosie przekracza jej arność. Mamy wtedy za dużo argumentów do

obsłużenia dlatego równanie jest niepoprawne. Komunikat drugi występuje w sytuacji odwrotnej do poprzedniej, czyli na stosie mamy niewystarczającą ilość elementów do zastosowania napotkanej operacji. Komunikat trzeci dotyczy sytuacji gdy zmienna którą napotkamy nie jest jednoliterowa. Ostatni komunikat sprawdza przynależność stałych do dziedziny. Zapobiega to podawaniu liczb ujemnych do programu.

3.3 Krok pomocniczy: zbudowanie listy równań

Aplikacja dopuszcza możliwość, w której działanie algebry jest argumentem innego działania. Dzięki temu istnieje możliwość stworzenia nieskończonego drzewa zagnieżdżeń funkcji. Z perspektywy programu zapis $f(1, x)$ oraz $f(1, f(1, x))$ to ten sam zapis gdyż funkcja najbardziej zewnętrzna ma te same argumenty: stałą 1 na pierwszym miejscu i niewiadomą na drugim. Chcielibyśmy jednak móc rozróżnić powyższe równania. Biorąc pod uwagę powyższy problem stworzyłem dodatkową strukturę reprezentowaną poprzez listę trójek (X,Y,Z), gdzie X oznacza nazwę operacji, Y jest listą argumentów operacji, a Z to symbol reprezentujący wynik działania



Rysunek 4: Przykładowe wyjście algorytmu

W ten sposób uzyskujemy dostęp do zagnieżdżonych operacji poprzez ich symbol wynikowy.

Podsumowanie

Literatura

[1] https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html