



**UMCS**  
UNIwersytet Marii Curie-Skłodowskiej

UNIwersytet Marii Curie-Skłodowskiej  
w Lublinie

Wydział Matematyki, Fizyki i Informatyki

Kierunek: **Informatyka**

**Viktar Zhdanovich**

Nr albumu: 300187

## **Projekt i implementacja interfejsu graficznego SAT-solvera**

Design and implementation of a SAT-solver graphical interface

Praca licencjacka

napisana w Katedrze Systemów Inteligentnych  
pod kierunkiem dr. Jacka Krzaczkowskiego

**Lublin 2023**

# Spis treści

<b>Wstęp</b>	<b>3</b>
<b>1 Wprowadzenie</b>	<b>4</b>
1.1 CNF	4
1.2 Problem spełnialności formuł logicznych	4
1.3 SAT-solvery	5
1.4 Format DIMACS CNF	5
1.5 Cel i zakres pracy	6
<b>2 Projekt</b>	<b>7</b>
2.1 Definicja funkcjonalności	7
2.2 Prototypowanie interfejsu	9
2.2.1 Edytor plików w formacie DIMACS CNF	9
2.2.2 Wykrycie, wskazywanie i naprawienie błędów	11
2.2.3 Podstawowe funkcje SAT-solvera	13
2.2.4 Wyświetlanie wartościowań	13
2.2.5 Formuła w postaci CNF	14
2.2.6 Łączenie formuł	15
2.3 Architektura aplikacji	16
<b>3 Implementacja</b>	<b>17</b>
3.1 Narzędzia	17
3.1.1 PySAT	17
3.1.2 FastAPI	17
3.1.3 React	18
3.2 Implementacja edytora	19
3.3 Implementacja sprawdzarki	19
3.4 Optymalizacje	20
<b>Podsumowanie</b>	<b>22</b>
<b>Spis rysunków</b>	<b>23</b>
<b>Bibliografia</b>	<b>24</b>

# Wstęp

Graficzny interfejs użytkownika (ang. graphical user interface, GUI) jest typem interfejsu za pomocą którego użytkownik wchodzi w interakcję z komputerem, programem. Alternatywą do interfejsów graficznych są interfejsy konsolowe (ang. command line interface, CLI). Wadą takiego sposobu korzystania z programu jest to, że mamy pamiętać komendy na pamięć i je ręcznie wpisywać. Z tego powodu, że to jest uciążliwe dla większości ludzi, stały się popularne interfejsy graficzne, które są bardzo ważną częścią każdej współczesnej aplikacji. Wyobraźmy, na przykład, co by było gdyby systemy operacyjne nie miały interfejsów graficznych? Brak interfejsu bardzo podwyższa poziom przygotowania dla korzystania z takiej aplikacji. Interfejsy, w większości, czynią skomplikowane programy dostępnymi i zrozumiałymi dla wielu ludzi, niskopoziomowe stają się wysokopoziomymi. Dobrze przemyślony, atrakcyjny, niezawodny, pomagający użytkownikowi interfejs napewno poszerza grono użytkowników programu, co może m.in. dobrze wpłynąć na rozwój przedsiębiorstwa lub sukces twórcy. Dlatego jest bardzo ważne dbać o to, żeby aplikacja była nie tylko funkcjonalna, ale i jej interfejs graficzny nie miał opóźnień, był intuicyjny i porządnie wyglądał.

# Rozdział 1

## Wprowadzenie

### 1.1 CNF

**Definicja 1** (CNF). Koniunkcyjna postać normalna (ang. conjunctive normal form) danej formuły logicznej to równoważna jej formuła zapisana w postaci koniunkcji klauzul. Klauzula jest zbiorem literalów połączonych alternatywą. Literalem nazywamy pojedynczą zmienną zdaniową lub pojedynczą zanegowaną zmienną zdaniową. Zmienna zdaniowa to zmienna która może przyjmować wartość *true* lub *false*

Przykład klauzuli:

$$(x \vee y \vee z).$$

Przykłady literalów:

$$p, q, \neg p$$

**Definicja formalna.** Formuła  $\psi$  jest w koniunkcyjnej postaci normalnej jeśli jest ona koniunkcją klauzul, z których każda jest alternatywą *literalów*, tzn. ma następującą postać:

$$(p_{11} \vee p_{12} \vee \dots \vee p_{1k_1}) \wedge (p_{21} \vee p_{22} \vee \dots \vee p_{2k_2}) \wedge \dots \wedge (p_{n1} \vee p_{n2} \vee \dots \vee p_{nk_n})$$

gdzie każde  $p_{ij}$  jest literalem.

Każdą formułę logiczną można przedstawić równoważnie w postaci CNF. Dla jednej formuły logicznej może istnieć kilka równoważnych jej formuł w CNF.

### 1.2 Problem spełnialności formuł logicznych

**Problem spełnialności formuł logicznych** jest ważym problemem obliczeniowym w teorii złożoności. Wejściem problemu jest formuła logiczna w postaci CNF. Problem polega na znalezieniu wartościowania, tzn. wartości wszystkich zmiennych zdaniowych, takich, kiedy formuła staje się prawdziwa. Formułę, która posiada takie wartościowanie nazywamy formułą *spełnialną* (ang. satisfiable), a która nie, odpowiednio *niespełnialna* (ang. unsatisfiable).

Na przykład, formuła

$$(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1)$$

jest spełnialna, dlatego, że istnieje wartościowanie które ją spełnia, np.:

$$x_1 = false, x_2 = false, x_3 = false$$

## 1.3 SAT-solvery

**SAT-solver** jest programem mający na celu rozwiązanie problemu spełnialności. Na wejściu program przyjmuje formułę w postaci CNF, a na wyjściu zwraca odpowiedź, czy podana formuła jest lub nie jest spełnialna. Oryginalny SAT-solver wymaga spełnienia **wszystkich** klauzul.

Większość SAT-solverów zwraca nie tylko informację o spełnialności, ale też przykładowe wartościowanie, jeśli formuła była spełnialna, lub minimalny zbiór niespełnialnych klauzul, jeśli formuła była niespełnialna.

SAT-solvery mają wiele udanych zastosowań w różnych dziedzinach, takich jak np. sztuczna inteligencja, automatyzacja projektowania elektronicznego (ang. Electronic Design Automation), analiza programów.

Istnieją różne rodzaje SAT-solverów, które rozwiązują inne odmiany problemu spełnialności formuł logicznych:

- MAX-SAT - szuka maksymalnej liczby klauzul, które mogą być spełnione, tzn. zadajemy pytanie ile maksymalnie klauzul z tej formuły dadzą formułę spełnialną jeśli połączymy je razem.
- Częściowy MAX-SAT (ang. Partial MAX-SAT) - podczas gdy klasyczny problem SAT wymaga spełnienia wszystkich klauzul, PM-SAT jest rozluźniony spełnienia tego wymogu poprzez oznaczenie niektórych klauzul jako miękkie (ang. soft), a innych jako twarde (ang. hard). Celem jest znalezienie wartościowania, które spełnia wszystkie klauzule twarde i maksymalizuje liczbę spełnionych klauzul miękkich.

## 1.4 Format DIMACS CNF

DIMACS CNF to format tekstowy reprezentujący formułę w koniunkcyjnej postaci normalnej. Pliki z formułami mogą być w dowolnym formacie tekstowym, ale najczęściej stosują się \*.cnf lub \*.txt.

Reguły których trzeba pilnować, żeby poprawnie stworzyć plik z formułą:

1. Linie zaczynające się od znaku *c* przedstawiają komentarze.
2. Linia zaczynająca się od znaku *p* jest definicją formuły i wygląda następująco: *p cnf l k*. Gdzie *l* i *k* są liczbami dodatnimi, gdzie *l* reprezentuje liczbę zmiennych formuły, a *k* reprezentuje liczbę klauzul.
3. Klauzule są umieszczane dokładnie po definicji formuły. Każda klauzula jest zadowana jako sekwencja liczb dziesiętnych odseparowanymi spacjami.
4. Każda linia zawiera dokładnie jedną klauzulę i ma kończyć się symbolem 0.
5. Zostawianie pustych linii w formule nie jest dozwolone.

Niepilnowanie tych reguł może doprowadzić do błędnych wyników lub w ogóle do unieruchomienia SAT-solvera.

Na przykład formuła w CNF

$$(x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee \neg x_1) \wedge (x_2 \vee x_3 \vee \neg x_1)$$

może być zakodowana w DIMACS CNF jako:

```
p cnf 4 4
1 2 3 4 0
-1 2 0
3 4 -1 0
2 3 -1 0
```

## 1.5 Cel i zakres pracy

Celem pracy jest stworzenie intuicyjnego i dostępnego dla wszystkich interfejsu graficznego dla oryginalnego SAT-solvera. Dostępność zapewnimy tworząc aplikację webową dlatego, że nie wymaga od użytkownika żadnych niepotrzebnych instalacji, działa prawie wszędzie, a do uruchomienia potrzebny jest tylko komputer i internet. Intuicyjność zapewnimy przemyślnym interfejsem, który w razie problemów będzie pomagał użytkownikowi w ich rozwiązaniu za pomocą odpowiednich komunikatów i sugestii. W pracy zostanie omówiony proces projektowania interfejsu oraz szczegóły implementacji niektórych, najbardziej interesujących części aplikacji.

Frontend będzie stworzony za pomocą biblioteki **React**, a backend za pomocą frameworku **FastAPI**. Na backendzie wykorzystamy pakiet **PySAT**, który daje możliwość korzystania z wielu zaimplementowanych SAT-solverów. Komunikację pomiędzy backendem a frontendem zapewnimy poprzez HTTP zapytania.

# Rozdział 2

## Projekt

### 2.1 Definicja funkcjonalności

Wygląd interfejsu graficznego mocno zależy od tego, jak zdefiniujemy funkcjonalności, więc zanim zaczniemy, mamy je ściśle zdefiniować.

Mamy mieć edytor plików DIMACS CNF który będzie pomagał nam wykrywać błędy w formułach. Można wyróżnić wystąpienie następujących błędów, każdemu z nich z góry przepisujemy unikatowy kod błędu:

- *Kod 0*: Nie można zostawiać pustych linii
- *Kod 1*: Definicja formuły może być niepoprawna
- *Kod 2*: Każda klauzula ma kończyć się zerem
- *Kod 3*: Klauzula może być niepoprawna
- *Kod 4*: W pliku nie może być więcej niż jedna definicji formuły
- *Kod 5*: Zmienna w klauzuli może być nie w zakresie zdefiniowanym w definicji

Dla każdego z wymienionych powyżej błędów mamy dawać użytkownikowi indywidualne sugestie do poprawienia:

- *Kod 0*: usuwanie lub edycja linijki
- *Kod 1*: edycja linijki z definicją
- *Kod 2*: dopisywane zera na końcu linijki
- *Kod 3*: usuwanie lub edycja linijki z klauzulą
- *Kod 4*: wymiana istniejącej lub usuwanie linijki z definicją
- *Kod 5*: edycja linijki z klauzulą

Błędów może być dużo, więc damy też możliwość do automatycznego poprawiania formuły. Przy tym w formule następują takie zmiany:

1. Na końcach linii są dopisywane zera.

2. Liczby zmiennych i klauzul w nagłówku są poprawiane na odpowiednie.
3. Puste linie i niepoprawne klauzule są usuwane.
4. Komentarze są usuwane.

Ze względu na to, że formuła może być nieostrożnie uszkodzona, użytkownik będzie poproszony o potwierdzenie chęci zrobienia automatycznego poprawienia.

W edytorze damy też możliwość pozbyć się zduplikowanych klauzul z formuły. Fajne jest to, że będziemy usuwać nie tylko dokładnie takie same klauzule, a i równoważne, ze względu na to, że alternatywa jest przemienne.

W edytorze oprócz opcji wklejania zawartości pliku z formułą pojawi się możliwość do wczytania jej w postaci DIMACS CNF z pliku w formacie \*.cnf lub \*.txt. Zmodyfikowaną formułę można będzie zapisać do pliku w formacie \*.txt.

Formuła z edytora będzie parsowana i wyświetlana w innym miejscu w zwykłej postaci CNF. W tej postaci formułę można będzie modyfikować: usuwać, zmieniać lub dodawać klauzule. Dzięki temu użytkownik dostanie możliwość jednoczesnej edycji formuły w CNF i DIMACS, będzie mógł zanim wybrać wygodniejszy dla siebie sposób pracy z programem.

Zdefiniujemy podstawowe funkcję aplikacji dotyczące pracy z SAT-solverem. Użytkownik będzie mógł:

- dowiedzieć się czy formuła jest lub nie jest spełnialna
- znaleźć pojedyncze wartościowanie spełniające formułę
- jeśli formuła jest spełnialna pojawia się możliwość próby znalezienia kolejnego wartościowania
- uruchomić i w dowolnej chwili skończyć proces znalezienia wszystkich wartościowań spełniających
- wybrać SAT-solver z którego chce korzystać

Znalezione wartościowania i ich ilość mamy w czytelny i jasny sposób pokazywać użytkownikowi. Dodamy jeszcze możliwość zapisania do pliku wartościowania w postaci zero-jedynkowej, żeby można było go wrzucić np. do przetwarzania przez jakiś inny program.

W końcu dodamy możliwość łączenia formuł. Nie można będzie łączyć 100 formuł naraz, ale można będzie połączyć dwie, co z kolei da możliwość łączenia dowolnej ilości formuł. Rezultat można będzie od razu wkleić do edytora, jeśli łączenie się powiodło.

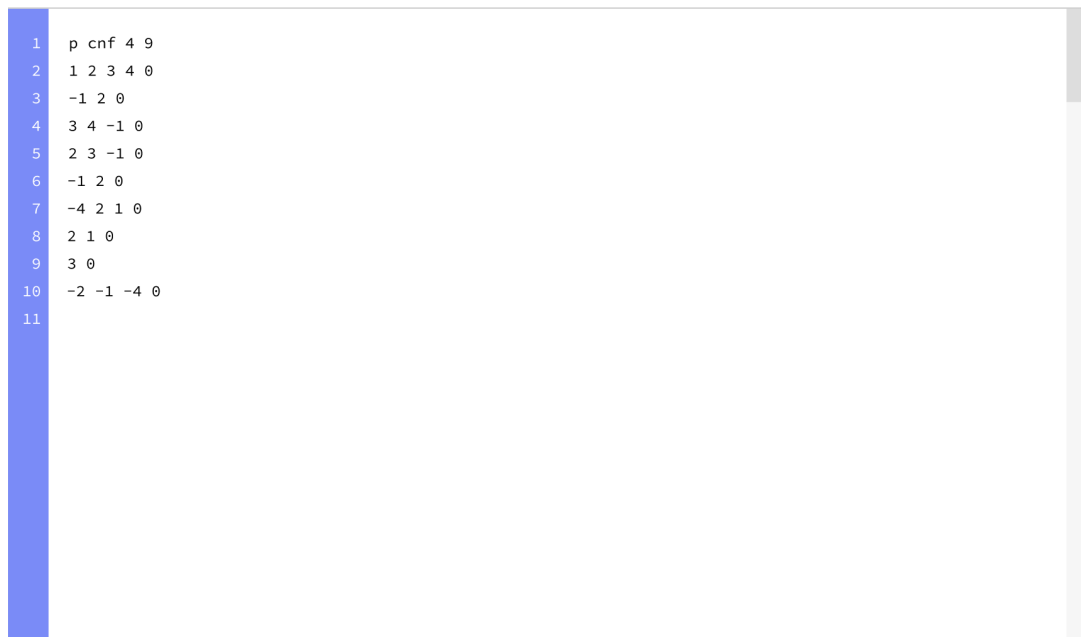


## 2.2 Prototypowanie interfejsu

W zakresie prototypowania komponent po komponencie stworzymy prototyp graficznego interfejsu SAT-solvera oraz rozważymy przyjęte decyzji projektowe.

### 2.2.1 Edytor plików w formacie DIMACS CNF

Korzystając z SAT-solverów jesteśmy zmuszeni korzystać z plików w formacie DIMACS CNF, więc przydałoby się mieć edytor do takich plików. W podstawowej wersji będzie wyglądał następująco:

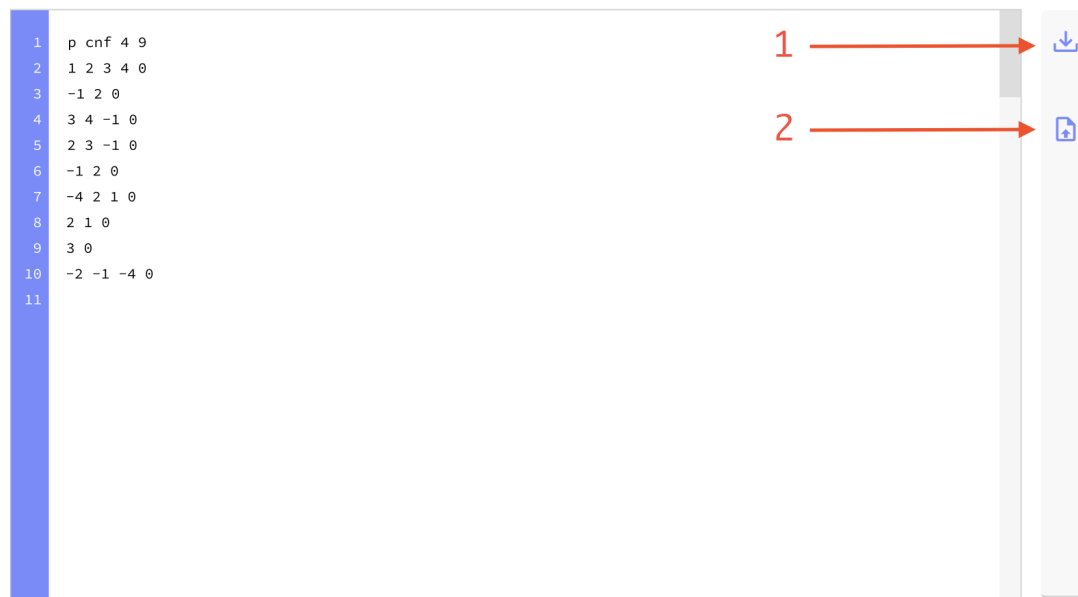


```
1 p cnf 4 9
2 1 2 3 4 0
3 -1 2 0
4 3 4 -1 0
5 2 3 -1 0
6 -1 2 0
7 -4 2 1 0
8 2 1 0
9 3 0
10 -2 -1 -4 0
11
```

Rysunek 2.1: Edytor podstawowy

W środku większą część zajmuje obszar w którym będziemy modyfikować naszą formułę, na razie istnieje możliwość tylko wklejenia jej do edytora. Po prawej stronie mamy scroll który będzie pomagał nam przemieszczać się po formułach z dużą liczbą klauzul. Po lewej stronie mamy słupek z informacją o numerach linii, w przyszłości nam się to przyda dla wykrycia i wskazywania błędów.

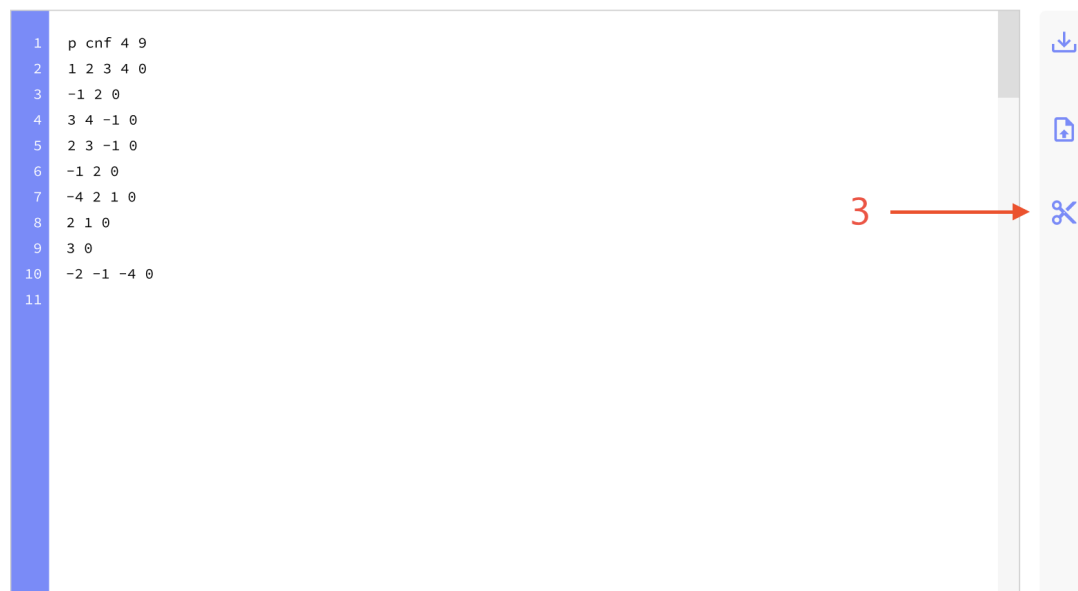
Nie za bardzo wygodne jest wklejanie zawartości pliku, więc damy użytkownikowi możliwość do załadowania pliku z formułą oraz możliwość do zapisania pliku z zmodyfikowaną formułą:



Rysunek 2.2: Edytor z możliwością załadowania i zapisania formuły

Po prawej stronie dodaliśmy obszar który będzie pomagał nam zarządzać edytorem, obecnie znajdują się tu dwa przyciski: nr. 1 daje możliwość wybrać i załadować plik z formułą, a nr.2 daje możliwość zapisać plik i dać mu nazwę.

W formule, przez przypadek, mogą pojawić się zduplikowane klauzule, chociaż to nie za bardzo wpływa na wydajność i szybkość działania SAT-solvera, damy użytkownikowi możliwość usunięcia takich klauzul:



Rysunek 2.3: Edytor z możliwością usunięcia zduplikowanych klauzul

Z obrazów przycisków z obszaru do zarządzania edytorem może nie być jasne, co one robią, więc dodana jest możliwość zobaczyć w postaci tekstowej tę informację. Przy najechnaniu myszką na każdy z przycisków pod nimi będzie pojawiała się podpowiedź (ang. tooltip) z tekstem. Na przykład pod przyciskiem nr.1 pojawi się komunikat "Upload formula".

### 2.2.2 Wykrycie, wskazywanie i naprawienie błędów

Dla poprawnego i przewidywanego działania SAT-solvera konieczne jest poprawne stworzenie pliku z formułą w formacie DIMACS CNF. Jako twórcy interfejsu graficznego mamy pomoc użytkownikowi poradzić z wykryciem, wskazywaniem i naprawieniem błędów w razie ich wystąpienia.

Kiedy formuła w edytorze będzie się zmieniała, będzie wywoływana funkcja wykrywająca błędy. Wykryte błędy mamy w jakiś sposób wyświetlić. Proponujemy podświetlenie odpowiednich linijek. Tak to będzie wyglądało:

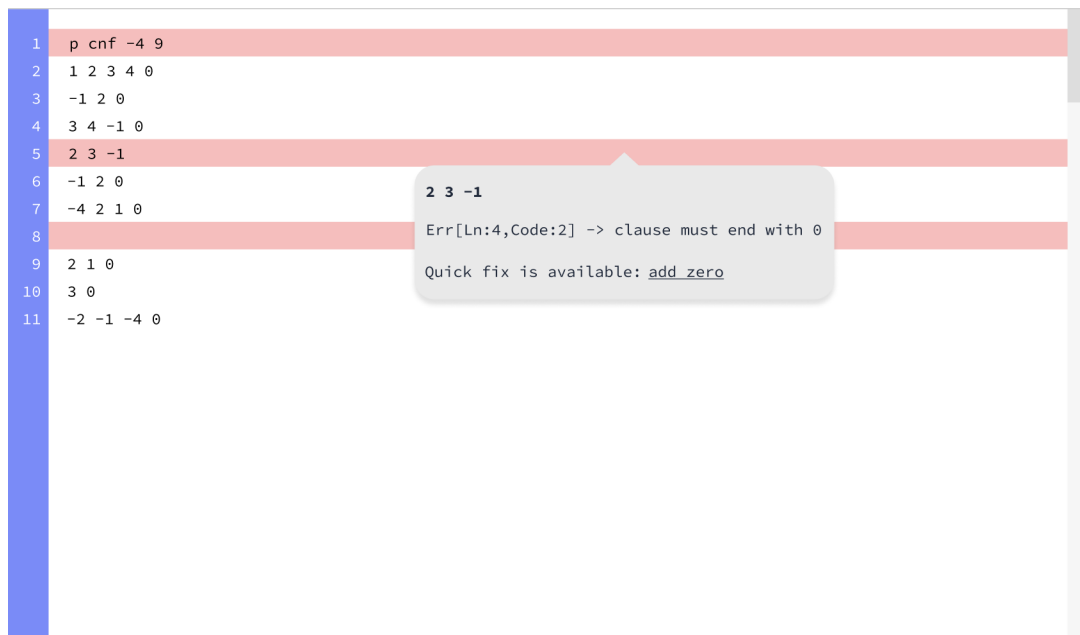


Rysunek 2.4: Edytor wraz z tabelą z błędami

Widzimy jeszcze, że w obszar do zarządzania formułą został dodany nowy przycisk pod numerem 4. Będzie on odpowiadał za automatyczne poprawienie błędów. Kiedy użytkownik nacisnie na ten przycisk, zanim poprawienie się zacznie, będzie wyświetlony komunikat mówiący o bezzwrotności działań użytkownika i prośbą o potwierdzenie chęci zrobienia popraw. Dodatkowo w komunikacie będzie napisano, że:

- puste linie, komentarze oraz uszkodzone klauzule zostaną usunięte
- liczba zmiennych i klauzul będzie przeliczona

W tym momencie, jeśli użytkownik będzie miał błędy w formule, będzie je dobrze widział, ale ma je też naprawić. Żeby to teraz zrobić, ma z góry wiedzieć w czym jest problem. Ułatwmy zrozumienie dodaniem odpowiednich podpowiedzi przy najechaniu myszką na czerwoną linię. Na rysunku 2.5 pojawiła się podpowiedź, gdzie w pierwszej linijce znajduje się treść uszkodzonej linii, w drugiej opis błędu, a w trzeciej propozycja do naprawienia w postaci przycisku. W tym przykładzie widzimy, że zaproponowano dodanie zera na końcu linii.



Rysunek 2.5: Edytor wyświetlający podpowiedź przy najechaniu myszką

Zostały nam jeszcze dwa problemy do rozwiązania. Po pierwsze, w razie błędów nie widzimy od razu ile ich jest. Po drugie, ze względu na to, że pliki z formułami mogą zawierać dużo klauzul, a zatem potencjalnie dużo błędów, nie mamy sposobu na wyświetlanie wszystkich błędów naraz w jednym miejscu. W drugim przypadku przydało by się też zostawić możliwość szybkiej naprawy.

Te dwa problemy można rozwiązać wprowadzając kolejny komponent, który będzie znajdował się od razu pod edytorem:

There are 3 errors in a formula !!!			Hide errors list -
Description	Line	Fix proposition	
Invalid formula definition [Ln:1, Cd:1]	p cnf -4 9	EDIT	
Clause must end with 0 [Ln:5, Cd:2]	2 3 -1	ADD ZERO	
No empty lines allowed [Ln:8, Cd:0]	empty line here 🙄	EDIT	DELETE

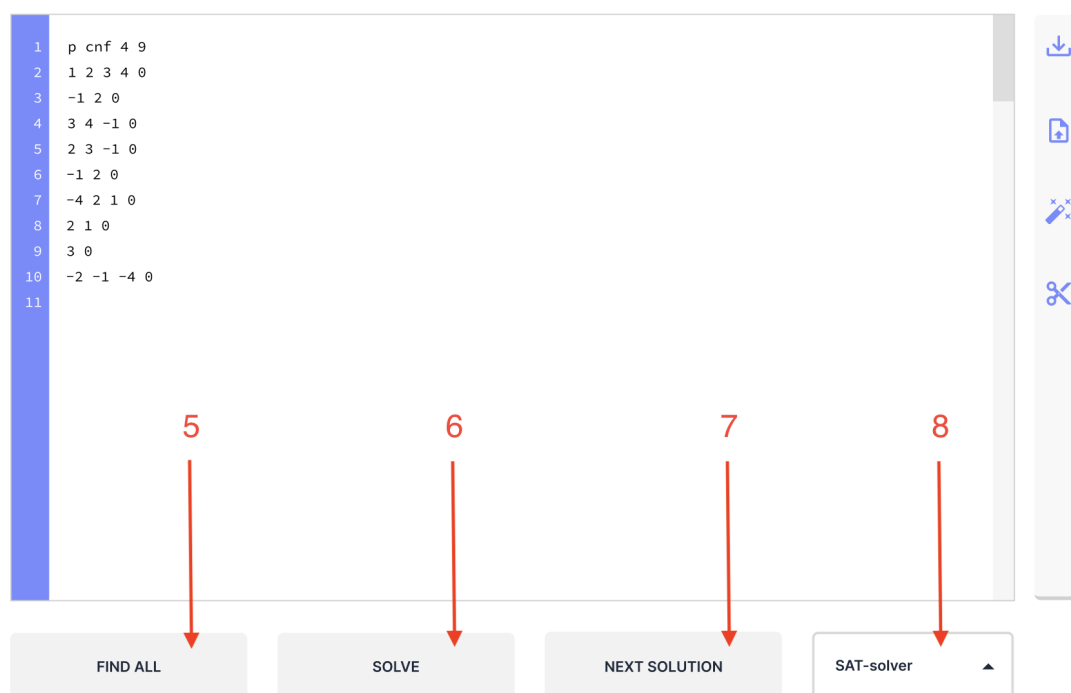
Rysunek 2.6: Tabela z wykrytymi błędami

Jest ukryty, kiedy formuła nie zawiera błędów. Ma dwa stany: otwarty i zamknięty. Domyślnie ma stan zamknięty. Na górze widzimy nagłówek w którym jest napisano ile mamy

błędów w formule. Klikając na nagłówek zmieniamy stan otwarcia. Tabela ma trzy kolumny: w pierwszej znajduje się opis błędu, w drugiej uszkodzona linia, a w trzeciej znajdują się przyciski, dzięki którym można szybko naprawić błąd. Błędy w tabeli są sortowane rosnąco w zależności od indexu linii wystąpienie.

### 2.2.3 Podstawowe funkcje SAT-solvera

Kiedy umiemy już wykrywać i naprawiać błędy, co zapewnia nam bezpieczne i przewidywane działania SAT-solvera, możemy dodać przyciski kontrolujące i dać użytkownikowi możliwość do uruchomienia go:



Rysunek 2.7: Edytor ze wszystkimi przyciskami

Przycisk nr.6 pozwala dowiedzieć czy formuła jest lub nie jest spełnialna. Jeśli jest spełnialna zwracane jest pierwsze wartościowanie.

Przycisk nr.7 pozwala znaleźć następne wartościowanie, jeśli takie istnieje. Jest nieaktywny jeśli formuła jest niespełnialna.

Przycisk nr.5 uruchamia wyszukiwanie wszystkich wartościowań. Wyszukiwanie można będzie przerwać w dowolnej chwili kiedy będzie znalezione pierwsze wartościowanie. Po przerwaniu wyszukiwanie można ponownie zacząć od momentu skończenia, w tym przypadku tekstem przycisku będzie nie "FIND ALL", a "FIND OTHER SOLUTIONS".

Przycisk nr.8 pozwala wybrać dowolny z dostępnych w pakiecie PySAT SAT-solver.

Jeśli formuła w edytorze zawiera błędy - uruchomienie SAT-solvera nie jest dozwolone. Wszystkie wymienione przyciski oprócz nr.8 będą nieaktywne.

### 2.2.4 Wyświetlanie wartościowań

Chcemy, żeby komponent pozwalał w jasny sposób widzieć jakie wartości zostały przepisane zmiennym formuły.

Mamy poradzić z problemem wyświetlania wartościowań dla formuł zawierających dużą liczbę zmiennych. Trudno będzie wyświetlić takie wartościowanie wprost w słupku lub w jednej linii, doprowadzi to do przekroczenia granic ekranu dowolnego rozmiaru wzdłuż osi X lub Y.

### 2.2.5 Formuła w postaci CNF

Komponent, który stworzymy, będzie dawał możliwość zoboczenia i edytowania formuły inaczej, niż w edytorze DIMACS, a mianowicie w zwykłej postaci CNF. Ważne tu jest umieszczenie wszystkiego w wygodny sposób dlatego, że formuły często zawierają dużo klauzul i to też może zająć dużo miejsca na ekranie, a im więcej umieszczamy elementów na ekranie, tym wolniejszy staje się rendering tego wszystkiego w przeglądarce. Wolne działanie aplikacji może źle wpłynąć na doświadczenie użytkownika przy korzystaniu z niej, czemu chcemy zapobiec.

Proponujemy stworzenie następnego komponentu:



Rysunek 2.8: Komponent pozwalający pracować z formułą w postaci CNF

To wszystko to jest ten sam komponent, ale na górze znajduje się w stanie, kiedy formuła jeszcze nie została dodana. Klikając na nagółwek komponentu możemy go zamknąć lub otworzyć. Domyślnie znajduje się w stanie otwartym.

Klikając na plus możemy ręcznie dodawać klauzule do formuły. Pojawi się do tego odpowiednie pole z miejscem do stworzenia wraz z instrukcją, w której jest wytłumaczone jak to prawidłowo zrobić.

W stanie nr.2, nad drugą klauzulą widzimy przyciski pozwalające ją modyfikować. Takie przyciski będą pojawiać się nad każdą klauzulą przy najechaniu na nią myszką. Przy naciśnięciu na śmietnik użytkownik zostanie zapytany, czy napewno chce usunąć klauzulę. W razie pozytywnej odpowiedzi klauzula będzie usunięta. Przy naciśnięciu na długopis, użytkownik dostaje możliwość do modyfikacji zawartości klauzuli. Kiedy modyfikowanie jest skończone, naciska na przysick "Zapisz", który stanie się dostępny w interfejsie graficznym lub "Enter" na klawiaturze.

Zostało nam jeszcze poradzić z obsługą formuł zawierających dużo liczbę klauzul. Zrobimy to jak w przypadku dużej ilości wartościowań formuły, czyli rozdzielimy dane

na strony. Ilość klauzul wyświetlanych na jednej stronie będzie stała i znajdowała się w kodzie programu, użytkownik nie będzie w stanie jej zmieniać. Domyślnie na jednej stronie będziemy pokazywali 115 klauzul. Takie rozdzielanie dobrze wpłynie na wydajność aplikacji dlatego, że przeglądarka nie będzie miała renderować całej formuły naraz, a tylko odpowiednią jej część. Wyglądać to będzie następująco:



Rysunek 2.9: Duża formuła rozdzielona na kilka stron

### 2.2.6 Łączenie formuł

Został nam do zaprojektowania ostatni komponent, taki, który będzie pozwalał łączyć formuły w formacie DIMACS CNF, zaprojektujemy go w następujący sposób:



Rysunek 2.10: Komponent pozwalający łączyć formuły

Widzimy tu najpierw dwa edytora i dwa przyciski. Edytory nie mają możliwości sprawdzenia błędów. Do edytorów można wkleić lub załadować formułę z pliku, klikając na przycisk nr.1 lub nr.2. Kiedy dwie formuły zostały załadowane, użytkownik może nacisnąć przycisk "LINK FORMULAS". Jeśli nie wystąpiło błędów podczas łączenia, pojawi się odpowiedni komunikat, że formuły zostały pomyślnie połączone oraz odblokuje się przycisk "PASTE RESULT IN THE EDITOR". Kiedy użytkownik kliknie na przycisk nr.4, to zostanie przekierowany na stronę z edytorem w który już będzie wklejony rezultat łączenia.

## 2.3 Architektura aplikacji

Na frontendzie będziemy robili: wykrycie, wskazywanie i wyświetlenie błędów formuły, synchronizacja zmian formuły w edytorze i w formacie CNF.

Część zadań rozwiążemy na backendzie: stwierdzenie czy formuła jest spełniala i zwrócenie pierwszego wartościowania, znalezienie następnego wartościowania, automatyczne poprawienie formuły, usuwanie zduplikowanych klauzul, łączenie formuł.

Na backendzie stworzymy następujące endpointy:

- post /fix ciało dimacs: string
- post /remove-duplicate ciało dimacs: string
- post /link ciało firstDimacs: string and secondDimacs: string
- post /solve ciało dimacs: string and solver: string
- post /next-solution ciało formula: string and solver: string



# Rozdział 3

## Implementacja

### 3.1 Narzędzia

#### 3.1.1 PySAT

Darmowa i otwarta biblioteka przeznaczona dla języka Python.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

#### 3.1.2 FastAPI

Darmowa i otwarta biblioteka przeznaczona dla języka Python.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

### 3.1.3 React

Darmowa i otwarta biblioteka przeznaczona dla języka JavaScript.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt

tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

## 3.2 Implementacja edytora

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

## 3.3 Implementacja sprawdzarki

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum.

Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

## 3.4 Optymalizacje

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum

pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

# Podsumowanie

Chcemy, żeby nasz program stał się początkiem tworzenia interfejsów graficznych dla różnych rodzajów SAT-solverów. Ze względu na brak takich programów. W tej pracy dotknęliśmy tylko małą część, dużo rzeczy można jeszcze do tego dodać.

Będę [2] zawsze otwarty na takie propozycje na GitHubie. Chciałbym, żeby [1] studenci naszej uczelni w [1] swoich pracach licencjackich lub [3] nawet magisterskich [4] kontynuowali rozwijanie i polepszenie naszego programu.

# Spis rysunków

2.1	Edytor podstawowy . . . . .	9
2.2	Edytor z możliwością załadowania i zapisania formuły . . . . .	10
2.3	Edytor z możliwością usunięcia zduplikowanych klauzul . . . . .	10
2.4	Edytor wraz z tabelą z błędami . . . . .	11
2.5	Edytor wyświetlający odpowiedź przy najechniu myszką . . . . .	12
2.6	Tabela z wykrytymi błędami . . . . .	12
2.7	Edytor ze wszystkimi przyciskami . . . . .	13
2.8	Komponent pozwalający pracować z formułą w postaci CNF . . . . .	14
2.9	Duża formuła rozdzielona na kilka stron . . . . .	15
2.10	Komponent pozwalający łączyć formuły . . . . .	15

# Bibliografia

- [1] Paul Adrien Maurice Dirac. *The Principles of Quantum Mechanics*. International series of monographs on physics. Clarendon Press, 1981. ISBN: 9780198520115.
- [2] Albert Einstein. “Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]”. In: *Annalen der Physik* 322.10 (1905), pp. 891–921. DOI: <http://dx.doi.org/10.1002/andp.19053221004>.
- [3] Donald Knuth. *Knuth: Computers and Typesetting*. URL: <http://www-cs-faculty.stanford.edu/~uno/abcde.html>. (accessed: 01.09.2016).
- [4] Donald E. Knuth. “Fundamental Algorithms”. In: Addison-Wesley, 1973. Chap. 1.2.