# Pemecah Sudoku Interaktif Dengan Logika Proposisi

**Proposal Tugas Akhir**

**Kelas TA 1**

**Muhammad Zakaria Musa**
**NIM: 1103130047**


**Program Studi Sarjana Teknik Informatika**

**Fakultas Informatika**

**Universitas Telkom**

**Bandung**

**2017**

# Lembar Persetujuan

**Pemecah Sudoku Interaktif Dengan Logika Proposisi**

***Interactive Sudoku Solver Using Propositional Logic***

**Muhammad Zakaria Musa**
**NIM: 1103130047**

Proposal ini diajukan sebagai usulan pembuatan tugas akhir pada
Program Studi Sarjana Teknik Informatika
Fakultas Informatika Universitas Telkom

Bandung, 24 November 2017
Menyetujui

| Calon Pembimbing 1 | Calon Pembimbing 2 |
| --- | --- |
| Yanti Rusmawati, Ph.D. | Muhammad Arzaki, M.Kom. |
| NIP: 15711785-1 | NIP: 15871701-2 |

# Abstrak

Sudoku adalah sebuah permainan teka-teki yang biasanya dimainkan oleh satu orang. Dalam pengerjaannya banyak pemain sudoku yang terjebak dan tidak dapat menyelesaikan teka-teki. Hal ini disebabkan oleh pemain salah memasukkan angka atau sudoku tidak memiliki solusi. Oleh karena itu perlu dibuat sebuah pemecah sudoku yang dapat memberikan keterpenuhan dari sebuah sudoku. Sehingga pemain dapat mengetahui sebuah sudoku memiliki solusi atau tidak. Dalam pengerjaannya metode yang digunakan adalah SAT *solver* yang dibuat dalam bahasa python. Setelah pembuatan aplikasi selesai akan dilakukan pengujian untuk mengontrol kualitas aplikasi.Pengujian akan menggunakan metode pengujian kotak hitam

**Kata Kunci:** metode formal, SAT solver, Sudoku, pengujian kotak hitam

# Daftar Isi

# Bab I

# Pendahuluan

## 1.1 Latar Belakang

Pada Tahun 2015 *The Organization for Economic Co-operation and Development*(OECD) merilis nilai dari *Programme for International Student Assessment*(PISA) [1]. Nilai kemampuan siswa yang diukur adalah matematika, sains, dan membaca. Pada kategori matematika Indonesia menempati peringkat 65 dari 71 negara dengan nilai 386. Sedangkan nilai rata-rata OECD untuk matematika adalah 470 [2], hal ini sungguh memprihatinkan. Oleh karena itu diperlukan katalis yang dapat mengembangkan kemampuan *logical thinking* serta *problem solving* anak, salah satunya adalah dengan sudoku.

Sudoku berasal dari kata *Sūji wa dokushin ni kagiru* yang berarti angkanya harus tunggal [3] adalah suatu *puzzle* (teka-teki) yang direpresentasikan oleh sebuah matriks (*array* dua dimensi) berukuran $n^2 \times n^2$ yang dibangun dari $n^2$ dengan submatriks (atau blok) yang berukuran $n \times n$. Sudoku merupakan *puzzle* yang biasanya dimainkan oleh satu orang. Pada awal permainan, terdapat beberapa sel yang telah terisi yang disebut dengan pemberian (*givens*). Untuk menyelesaikan permainan ini, seorang pemain harus mengisi setiap sel yang belum terisi dengan angka di antara 1 sampai $n^2$ sedemikan sehingga setiap baris, setiap kolom, dan setiap blok (submatriks berukuran $n \times n$) memuat tepat satu bilangan di antara 1 sampai $n^2$. Biasanya suatu sudoku didesain agar tepat memiliki satu kemungkinan solusi. Hal ini juga mengakibatkan sudoku dapat diselesaikan hanya dengan mengandalkan penalaran yang sederhana. Pengisian suatu sel dapat dilakukan dengan meninjau kemungkinan dari isi sebuah sel.

Banyak pemain yang terjebak pada teka-teki sudoku dan tidak dapat melanjutkan permainan. Hal ini terjadi karena pemain mengisi nilai yang salah atau sudoku tidak memiliki solusi. Oleh karena itu banyak pemain yang membutuhkan pemecah sudoku yang dapat memberikan solusi dari sebuah sudoku. Sehingga pemain dapat mengetahui ada tidaknya solusi.

Hingga saat ini, sudah banyak penelitian yang membahas penyelesaian sudoku secara matematis maupun komputasional. Salah satu metode yang cukup dikenal adalah penyelesaian sudoku dengan memanfaatkan masalah keter-

penuhan formula proposisional (*propositional satisfiability problem*). Dengan pendekataan ini, syarat-syarat yang harus dipenuhi oleh suatu sudoku dimodelkan dengan satu atau lebih formula logika proposisi [4, 5]. Keterpenuhan (*satisfiability*) dari himpunan formula yang memodelkan syarat-syarat ini akan menjamin bahwa suatu sudoku memiliki suatu solusi.

Pada tugas akhir ini, penulis akan membuat pemecah sudoku interaktif berbasis Python. Bahasa Python dipilih karena implementasi SAT solver dalam bahasa Python masih masih sedikit dibandingkan pada bahasa C dan C++ [6]. Kata interaktif adalah sebuah hubungan dua arah [7] sehingga penulis akan membuat aplikasi yang dengan dua arah sehingga pengguna dapat mendapatkan solusi sudoku menggunakan aplikasi dengan mudah. Aplikasi akan diuji menggunakan pengujian kotak hitam untuk mengetahui ada tidaknya solusi dari aplikasi [8].

## 1.2 Perumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah pada tugas akhir ini adalah bagaimana cara membuat SAT solver bebasis Python untuk pemecah sudoku yang interaktif.

## 1.3 Batasan Masalah

Batasan pada tugas akhir ini terbatas pada:

1. Penulis hanya membangun aplikasi sudoku solver berukuran $(4 \times 4)$, $(9 \times 9)$, $(16 \times 16)$. Ukuran tersebut diambil karena sesuai dengan tingkat kesulitan yang penulis akan buat yaitu mudah, menengah, sulit.

2. Untuk setiap ukuran sudoku memiliki empat tingkat kesulitan yaitu untuk mudah, menengah, dan sulit. Tingkat kesulitan tersebut di ambil dari database sudoku yang penulis ambil.

3. Program mengeluarkan kemungkinan solusi dari sudoku. Tingkat kesulitan *custom* akan mengambil dari sudoku yang dibuat oleh pemain.

4. Aplikasi akan diuji dengan metode pengujian kotak hitam (*black box testing*). Pengujian akan menggunakan *test plan* yang penulis buat.

## 1.4 Tujuan

Tujuan yang ingin dicapai pada tugas akhir ini adalah untuk membuat pemecah sudoku yang interaktif.

## 1.5 Rencana Kegiatan

Pada pengerjaan tugas akhir ini beberapa hal yang akan saya lakukan adalah sebagai berikut:

1. Studi literatur.
2. Spesifikasi aplikasi
3. Pembuatan aplikasi.
4. Pengujian aplikasi.
5. Analisis aplikasi.
6. Penulisan laporan.

## 1.6 Jadwal Kegiatan

Jadwal pengerjaan tugas akhir sesuai dengan alur yang telah dibuat.

Tabel 1.1: Jadwal Kegiatan.

| No | Jenis Kegiatan | Bulan | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | November 2017 | Desember 2017 | Januari 2018 | Februari 2018 | Maret 2018 | April 2018 | Mei 2018 |
| 1 | Studi literatur | | | | | | | |
| 2 | Spesifikasi aplikasi | | | | | | | |
| 3 | Pembuatan aplikasi | | | | | | | |
| 4 | Pengujian aplikasi | | | | | | | |
| 5 | Analisis aplikasi | | | | | | | |
| 6 | Penulisan Laporan | | | | | | | |

# Bab II

# Kajian Pustaka

Pada bab ini penulis akan menjelaskan teori yang digunakan selama pengerjaan tugas akhir.

## 2.1 Sudoku

Sudoku berasal dari kata *Sūji wa dokushin ni kagiru* yang berarti angkanya harus tunggal [3]. Sebelum sudoku modern berkembang. *Puzzle latin square* terlebih dahulu berkembang yang di repretasikan oleh sebuah matriks berukuran $n \times n$ yang memiliki angka $1hinggan$ yang harus diisi oleh pemain. untuk menyelesakain *puzzle* setiap baris dan kolom pada *latin square* harus memiliki nilai angka yang berbeda. *Latin square* pertama kali diciptakan oleh Euler pada tahun 1783 [9]. Sudoku modern sendiri lahir dari Howard Garns yang dipublikasikan di *Dell Magazines* pada 1979 [10]. Sudoku dipopulerkan di jepang oleh *Nikoli in the paper Monthly Nikolist* pada April 1984. Sudoku adalah suatu *puzzle* (teka-teki) yang direpresentasikan oleh sebuah matriks (*array* dua dimensi) berukuran $n^2 \times n^2$ yang dibangun dari $n^2$ dengan submatriks (atau blok) yang berukuran $n \times n$. Sudoku merupakan *puzzle* yang biasanya dimainkan oleh satu orang. Pada awal permainan, terdapat beberapa sel yang telah terisi yang disebut dengan pemberian (*givens*). Untuk menyelesaikan permainan ini, seorang pemain harus mengisi setiap sel yang belum terisi dengan angka di antara 1 sampai $n^2$ sedemikan sehingga setiap baris, setiap kolom, dan setiap blok (submatriks berukuran $n \times n$) memuat tepat satu bilangan di antara 1 sampai $n^2$. Biasanya suatu sudoku didesain agar tepat memiliki satu kemungkinan solusi. Hal ini juga mengakibatkan sudoku dapat diselesaikan hanya dengan mengandalkan penalaran yang sederhana. Pengisian suatu sel dapat dilakukan dengan meninjau kemungkinan dari isi sebuah sel. Berikut adalah contoh sudoku $4 \times 4$ serta solusinya:

| | | | |
|---|---|---|---|
| | | | 2 |
| | 3 | | |
| 4 | | | |
| | | 2 | |

Gambar 2.1: Sudoku dengan *given*.Gambar di ambil dari [11].

Pada sel (4,3) terdapat angka 2 sehingga pada blok kiri bawah hanya sel (3,2) saja yang dapat diisi oleh angka 2. Lalu pada sel (1,4) terdapat angka 2 sehingga pada blok kiri atas hanya sel (2,1) saja yang dapat diisi oleh angka 2. Pada sel (2,2) terdapat nilai 3 maka blok kiri bawah hanya dapat mengisi angka 3 pada sel(4,1). Lalu Pada sel (2,2) terdapat nilai 3 maka blok kanan atas hanya dapat mengisi angka 3 pada sel(1,3). Pada blok kiri bawah hanya ada satu sel yang belum terisi maka sel (4,2) diisi dengan angka 1. Pada kolom 1 dan 2 hanya terdapat satu sel yang belum terisi maka kedua sel tersebut diisi dengan angka yang kurang, sel(1,1) angka 1 dan sel (1,2) angka 4. Pada sel(1,3) dan(4,1) terdapat angka 3 sehingga pada blok kanan bawah hanya sel(3,4) yang dapat diisi angka 3. Pada baris 4 hanya satu sel saja yang belum terisi maka sel(4,4) diisi dengan angka 4. Lalu kita dapat mengisi sisanya yaitu sel(2,3) dengan angka 4 lalu sel(2,4) serta (3,3) dengan angka 1.

| | | | |
|---|---|---|---|
| 1 | 4 | 3 | 2 |
| 2 | 3 | 4 | 1 |
| 4 | 2 | 1 | 3 |
| 3 | 1 | 2 | 4 |

Gambar 2.2: Sudoku yang sudah terselesaikan. Gambar di ambil dari [11].

## 2.2 *Conjunctive Normal Form*(CNF)

Sebuah formula memiliki bentuk CNF apabila formula tersebut konjungsi dari satu atau lebih klausa dimana klausa adalah disjungsi dari literal [12]. Operator yang bisa digunakan dalam CNF adalah $or(\vee)$, $and(\wedge)$, dan, $not(\neg)$. Operator *not* hanya dapat digunakan sebagai bagian dari literal, yang berarti *not* hanya dapat mendahului variabel proposisional.

Berikut adalah contoh formula CNF:

- $((x_1 \vee x_2) \wedge \neg x_3)$
- $(p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p)$

Berikut adalah contoh formula yang bukan CNF:

- $\neg\,(q \vee p)$, karena terdapat *or* didalam *not*
- $p \wedge (q \vee (r \wedge s))$, karena terdapat *and* didalam *or*

Setiap formula bisa dilakukan kesetaraan agar memiliki bentuk CNF. Berikut adalah bentuk CNF dari formula diatas:

- $\neg q \wedge p$
- $p \wedge (q \vee r) \wedge (q \vee s)$

## 2.3 *Satisfiability Problem*(SAT)

Masalah SAT (SAT *problem*) adalah salah satu masalah penting dalam logika komputasional [12]. Dalam pengerjaannya SAT berfokus pada membuktikan sebuah klausa *satisfiable* sehingga SAT berfokus pada menemukan sebuah model yang membuat klausa tersebut *satisfiable*. Jika tidak ditemukan sebuah model yang membuat klausa tersebut *satisfiable* maka klausa tersebut *unsatisfiable*, Untuk mengerjakan masalah tersebut dapat diselesaikan menggunakan tabel kebenaran namun hal tersebut kurang efektif, seperti contoh berikut. Tentukan formula berikut $(p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p)$ *satisfiable*, pada formula tersebut dibandingkan menggunkan tabel kebenaran kita bisa menggunakan *reasoning*. Dapat kita lihat formula $(p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p)$ akan bernilai benar jika ketiga variabel (p, q, dan r) memiliki nilai kebenaran yang sama, sehingga formula tersebut *satisfiable*. Masalah SAT memiliki bentuk CNF serta merupakan masalah NP-complete, hingga saat ini tidak terdapat algoritma yang efisien untuk memecahkan masalah tersebut.

## 2.4 Sudoku Sebagai Masalah SAT

Sudoku memiliki beberapa aturan contohnya untuk sudoku berukuran $9 \times 9$ yaitu :

1. Setiap baris memuat bilangan antara 1 hingga 9.
2. Setiap kolom memuat bilangan antara 1 hingga 9.
3. Setiap submatriks atau blok $3 \times 3$ memuat bilangan antara 1 hingga 9.
4. Setiap sel memuat paling banyak satu bilangan antara 1 hingga 9.

Definisi dari $p(r, c, n)$ adalah sel yang memiliki perpotongan baris $r$ dan kolom $c$ dengan nilai $n$, dengan $1 \leq r, c, n \leq 9$. Sebagai contoh :

- $p(1, 3, 2)$ berarti sel pada baris 1 dan kolom 3 memuat nilai 2
- $p(4, 2, 9)$ berarti sel pada baris 4 dan kolom 2 memuat nilai 9

Dari aturan-aturan tersebut akan ditranslasikan menjadi bentuk CNF lalu digunakan pada SAT *solver*. Dengan aturan yang telah ditranslasikan dalam CNF sebagai berikut:

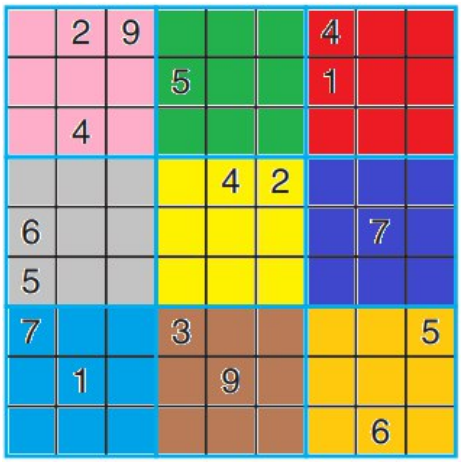
1. Setiap baris memuat bilangan antara 1 hingga 9 :
   $\bigwedge_{r=1}^{9} \bigwedge_{n=1}^{9} \bigvee_{c=1}^{9} p(r, c, n)$
2. Setiap kolom memuat bilangan antara 1 hingga 9 :
   $\bigwedge_{c=1}^{9} \bigwedge_{n=1}^{9} \bigvee_{r=1}^{9} p(r, c, n)$
3. Setiap submatriks atau blok $3 \times 3$ memuat bilangan antara 1 hingga 9 :
   $\bigwedge_{i=0}^{2} \bigwedge_{j=0}^{2} \bigwedge_{n=1}^{9} \bigvee_{r=3i+1}^{3i+3} \bigvee_{c=3j+1}^{3j+3} p(r, c, n)$
4. Setiap sel memuat paling banyak satu bilangan antara 1 hingga 9 :
   $\bigwedge_{r=1}^{9} \bigwedge_{c=1}^{9} \bigwedge_{n=1}^{8} \bigwedge_{i=n+1}^{9} (\neg p(r, c, n) \vee \neg p(r, c, i))$

Pada aturan 1 untuk menyatakan baris $r$ memuat bilangan antara 1 hingga 9, maka dapat ditulis dengan $\bigvee_{c=1}^{9} p(r, c, n)$. Untuk menyatakan bahwa baris $r$ memuat semua bilangan pada $\{1, ..., 9\}$ maka dapat ditulis dengan $\bigwedge_{n=1}^{9} \bigvee_{c=1}^{9} p(r, c, n)$. Untuk menyatakan bahwa setiap baris memuat semua bilangan pada $\{1, ..., 9\}$ dapat ditulis dengan $\bigwedge_{r=1}^{9} \bigwedge_{n=1}^{9} \bigvee_{c=1}^{9} p(r, c, n)$

Pada aturan 2 untuk menyatakan kolom $c$ memuat bilangan antara 1 hingga 9, maka dapat ditulis dengan $\bigvee_{r=1}^{9} p(r, c, n)$ Untuk menyatakan bahwa kolom $c$ memuat semua bilangan pada $\{1, ..., 9\}$ maka dapat ditulis dengan

$\bigwedge_{n=1}^{9}\bigvee_{r=1}^{9}p\,(r,c,n)$. Untuk menyatakan bahwa setiap kolom memuat semua bilangan pada $\{1,...,9\}$ dapat ditulis dengan $\bigwedge_{c=1}^{9}\bigwedge_{n=1}^{9}\bigvee_{r=1}^{9}p\,(r,c,n)$

Pada aturan 3 karena sudoku dengan ukuran $9 \times 9$ memiliki sel dengan ukuran $3 \times 3$ maka terdapat 9 blok pada sudoku sehingga :




Gambar 2.3: Gambar sudoku di ambil dari [13].

- blok pertama(merah muda) akan dimulai pada sel $(1,1)$ dan di akhiri pada sel $(3,3)$
- blok kedua(hijau) akan dimulai pada sel $(4,1)$ dan di akhiri pada sel $(6,3)$
- blok ketiga(merah) akan dimulai pada sel $(7,1)$ dan di akhiri pada sel $(9,3)$
- blok keempat(abu-abu) akan dimulai pada sel $(1,4)$ dan di akhiri pada sel $(3,6)$
- blok kelima(kuning) akan dimulai pada sel $(4,4)$ dan di akhiri pada sel $(6,6)$
- blok keenam(biru) akan dimulai pada sel $(7,4)$ dan di akhiri pada sel $(9,6)$
- blok ketujuh(biru muda) akan dimulai pada sel $(1,7)$ dan di akhiri pada sel $(3,9)$
- blok kedelapan(coklat) akan dimulai pada sel $(4,7)$ dan di akhiri pada sel $(6,9)$

- blok kesembilan(jingga) akan dimulai pada sel $(7, 7)$ dan di akhiri pada sel $(9, 9)$

Sehingga sebuah blok memuat semua sel $(r, c)$ dengan $3i + 1 \leq r \leq 3i + 3$ dan $3j + 1 \leq c \leq 3j + 3$, dengan $0 \leq i, j \leq 2$. Oleh karena itu aturan 3 dapat ditulis dengan $\bigwedge_{i=0}^{2} \bigwedge_{j=0}^{2} \bigwedge_{n=1}^{9} \bigvee_{r=3i+1}^{3i+3} \bigvee_{c=3j+1}^{3j+3} p(r, c, n)$

Pada aturan 4 agar setiap sel memuat angka 1 hingga 9 dapat ditulis dengan $\bigwedge_{r=1}^{9} \bigwedge_{c=1}^{9} \bigvee_{n=1}^{9} p(r, c, n)$. Lalu agar setiap selnya memiliki paling banyak satu nilai maka nilai pada sel tersebut akan dibandingkan dengan nilai lainnya pada sel yang sama dan jika terdapat dua atau lebih nilai pada sel yang sama maka akan mengeluarkan nilai *false* hal tersebut dapat ditulis dengan $\bigwedge_{r=1}^{9} \bigwedge_{c=1}^{9} \bigwedge_{n=1}^{8} \bigwedge_{i=n+1}^{9} (\neg p(r, c, n) \vee \neg p(r, c, i))$.

## 2.5 SAT *Solver*

SAT *solver* adalah sebuah aplikasi yang dibuat untuk menyelesaikan masalah SAT. Salah satu SAT *solver* yang cepat adalah MiniSat. SAT *solver* sendiri menerima masukan formula dalam bentuk CNF dan mengeluarkan dua buah keluaran yaitu 'sat' ditambah sebuah model jika formula *satisfiable*, dan 'unsat' jika formula *unsatisfiable*. Format masukan pada SAT *solver* atau yang disebut DIMACS adalah sebagai berikut :

1. Pertama baris komentar dapat ditulis dengan: `c <komentar>`
2. Lalu banyak variabel dan klausa`p cnf <banyak-variabel> <banyak-klausa>`
3. Lalu klausa:
    - Setiap variabel di representasikan dengan sebuah integer $\geq 1$.
    - Integer dengan nilai negatif mengartikan negasi literal.
    - Literal pada sebuah klausa dipisahkan oleh spasi.
    - Akhir dari klausa ditulis dengan 0

Contoh masukan DIMACS:
Formula : $(p \vee q) \wedge (\neg q \vee r \vee \neg s) \wedge (s \vee \neg r)$
Ditulis:

```
c Baris komentar
p cnf 4 3
1 2 0
-2 3 -4 0
4 -3 0
```

Contoh keluaran SAT *solver*:

Formula: $(p \vee q) \wedge (\neg q \vee r \vee \neg s) \wedge (s \vee \neg r)$

Keluaran:

```
SAT
-1 2 -3 -4 0
```

Untuk penggunaan minisat dapat ditulis dengan: `./minisat namaFileMasukan namaFileKeluaran`. Kekurangan pada SAT *solver* adalah solusi *satisfiable* yang dikeluarkan hanyalah satu. Kekurangan ini dapat diatasi dengan memasukkan solusi sebelumnya kedalam salah satu klausa pada masukan DIMACS dalam bentuk negasinya. Proses tersebut akan diulangi sampai keluaran dari SAT *solver* adalah *unsatisfiable*.

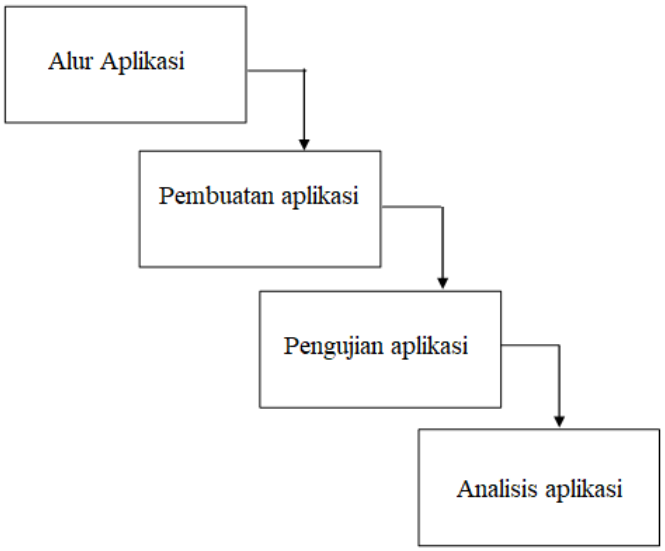
**Skrip NuSMV 1** Prosedur menemukan semua solusi.

```
1: procedure SOLUTION(in, out, sol)
2:     Start
3:     RunSAT(in,out)                                  ▷ Menjalankan SAT solver
4:     while out[0] ≠′ unsat′ do          ▷ Melihat baris pertama pada file out
5:         sol ← out[1]                                ▷ Menambahkan solusi baru
6:         For n = 0 : len(out[1]) do
7:             temp ← out[1][n] ∗ −1                   ▷ Negasi dari solusi
8:             in append(temp)          ▷ Menambahkan negasi solusi sebagai
   klausa baru
9:         RunSAT(in,out)                              ▷ Menjalankan SAT solver
10:    end while
11:    End
12:    Return sol                                      ▷ Mengeluarkan solusi
13: end procedure
```

# Bab III

# Metodologi

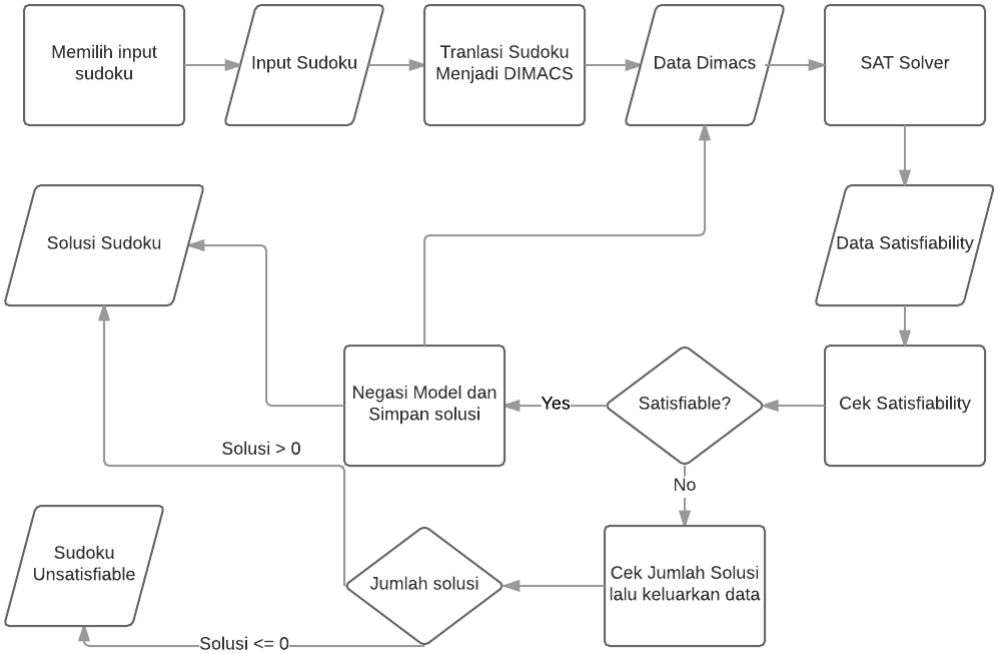
Pada bab ini penulis menjelaskan mengenai metodologi yang digunakan selama pengerjaan tugas akhir.




Gambar 3.1: Metodologi.

## 3.1 Alur Aplikasi

Aplikasi akan memiliki diagram alir seperti berikut:

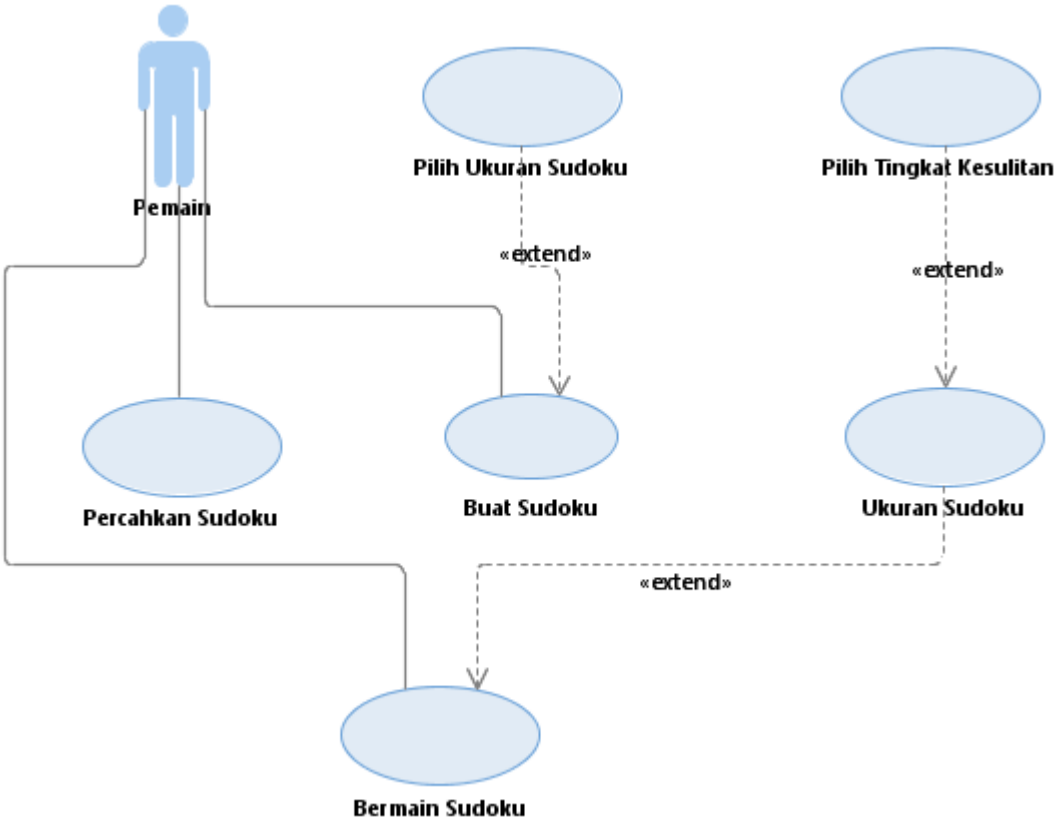



Gambar 3.2: Diagram alir.

## 3.2 Pembuatan aplikasi

Pembuatan aplikasi pemecah sudoku interaktif berbasis Python.

### 3.2.1 Use Case




Gambar 3.3: *Use Case Diagram.*

Fungsionalitas aplikasi adalah sebagai berikut:

1. Bermain sudoku.
2. Membuat sudoku.
3. Memecahkan sudoku.
4. Pilih ukuran sudoku.
5. Pilih kesulitan sudoku.

### 3.2.2 Aturan Sudoku Dalam Python

Aturan-aturan yang ada dalam bentuk CNF di ubah kedalam bahasa Python serta dibuatkan aturan DIMACSnya.

1. Setiap baris memuat bilangan antara 1 hingga 9 :
   $\bigwedge_{r=1}^{9}\bigwedge_{n=1}^{9}\bigvee_{c=1}^{9}p\left(r,c,n\right)$

   Skrip Python :

```
def aturan1(dimacs):
    for r in range(1, 10):
      for n in range(1, 10):
        for c in range(1, 10):
          dimacs += str(r) + str(c) + str(n) + ' '
        dimacs += '0\n'
        klausa += 1
    return dimacs
```

2. Setiap kolom memuat bilangan antara 1 hingga 9 :
   $\bigwedge_{c=1}^{9}\bigwedge_{n=1}^{9}\bigvee_{r=1}^{9}p\left(r,c,n\right)$

   Skrip Python :

```
def aturan2(dimacs):
    for c in range(1, 10):
      for n in range(1, 10):
        for r in range(1, 10):
          dimacs += str(r) + str(c) + str(n) + ' '
        dimacs += '0\n'
        klausa += 1
    return dimacs
```

3. Setiap submatriks atau blok $3 \times 3$ memuat bilangan antara 1 hingga 9 :
   $\bigwedge_{i=0}^{2}\bigwedge_{j=0}^{2}\bigwedge_{n=1}^{9}\bigvee_{r=3i+1}^{3i+3}\bigvee_{c=3j+1}^{3j+3}p\left(r,c,n\right)$

   Skrip Python :

```
def aturan3(dimacs):
    for i in range(0, 3):
```

```
    for j in range(0, 3):
      for n in range(1, 10):
        for r in range(3*i+1, 3*i+4):
          for c in range(3*j+1, 3*j+4):
            dimacs += str(r) + str(c) + str(n) + ' '
        dimacs += '0\n'
        klausa += 1
  return dimacs
```

4. Setiap sel memuat paling banyak satu bilangan antara 1 hingga 9 : $\bigwedge_{r=1}^{9}\bigwedge_{c=1}^{9}\bigwedge_{n=1}^{8}\bigwedge_{i=n+1}^{9}(\neg p(r,c,n) \vee \neg p(r,c,i))$

   Skrip Python :

```
def aturan4(dimacs):
    for e in range(0, 3):
      for c in range(0, 3):
        for n in range(1, 9):
          for i in range(n+1, 10):
            dimacs += '-' + str(r) + str(c) + str(n) + ' ' + '-'
            + str(r) + str(c) + str(i) + ' 0\n'
          klausa += 1
    return dimacs
```

## 3.3 Pengujian aplikasi

Pada tahap ini aplikasi akan diujikan kebeberapa penguji dengan diberikan lembar *test plan* yang meliputi beberapa aspek kualitas pada aplikasi. Teknik yang digunakan pada pengujian adalah pengujian kotak hitam [8].

## 3.4 Analisis aplikasi

Menganalisa kesusaian keluaran dari aplikasi berdasarkan hasil dari pengujian.

# Daftar Pustaka


[1] "Pisa 2015 result," http://www.keepeek.com/Digital-Asset-Management/oecd/education/pisa-2015-results-volume-i9789264266490-enpage1.

[2] "The latest ranking of top countries in math, reading, and science is out," http://www.businessinsider.sg/pisa-worldwide-ranking-of-math-science-reading-skills-2016-12/?r=US&IR=T.

[3] "Ed pegg jr.'s math games: Sudoku variations," http://www.mathpuzzle.com/MAA/41-Sudoku

[4] G. Kwon and H. Jain, "Optimized CNF encoding for sudoku puzzles," in *Proc. 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR2006)*, 2006, pp. 1–5.

[5] I. Lynce and J. Ouaknine, "Sudoku as a SAT Problem." in *ISAIM*, 2006.

[6] "Understanding SAT by Implementing a Simple SAT Solver in Python," http://sahandsaba.com/understanding-sat-by-implementing-a-simple-sat-solver-in-python.html.

[7] "Arti kata interaktif-kbbi," https://kbbi.web.id/interaktif.

[8] B. Beizer, *Black-box testing: techniques for functional testing of software and systems.* John Wiley & Sons, Inc., 1995.

[9] S. Jones, P. Roach, and S. Perkins, "Properties of sudoku puzzles," in *Proceedings of the 9th international conference on Software Engineering.* IEEE Computer Society Press, 11 2007, pp. 7–11.

[10] "The answer men," http://content.time.com/time/magazine/article/0,9171,2137423,00.htr

[11] "Easy sudoku puzzles 001," http://www.printablesudoku99.com/.

[12] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and reasoning about systems.* Cambridge University Press, 2004.

[13] K. Rosen, *Discrete Mathematics and Its Applications.* McGraw-Hill, 2007.

# Lampiran

Pada bab ini akan berisi lampiran dari pengerjaan tugas akhir.