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, 1 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

Abstrak **i**

Daftar Isi **ii**

I Pendahuluan **1**
1.1 Latar Belakang 1
1.2 Perumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 2
1.5 Rencana Kegiatan 2
1.6 Jadwal Kegiatan 3

II Kajian Pustaka **4**
2.1 Sudoku 4
2.2 *Satisfiability Problem*(SAT) 5
2.2.1 Sudoku Sebagai Masalah SAT 5
2.3 SAT Solver 6

III Metodologi **8**
3.1 Analisa spesifikasi dan Kebutuhan sistem 8
3.2 Translasi spesifikasi sistem 8
3.3 Pembuatan aplikasi 10
3.3.1 Spesifikasi Aplikasi 11
3.3.2 *Analysis* 11
3.3.3 *Design* 11
3.3.4 Implementasi 11
3.4 Pengujian aplikasi 12
3.5 Analisis sistem 12

Daftar Pustaka **13**

Lampiran **13**

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) [**?**]. Yang merupakan nilai kemampuan siswa pada matematika, sains, dan membaca. Pada kategori matematika Indosnesia menempati peringkat 65 dari 71 negara dengan nilai 386. Sedangkan nilai rata-rata OECD untuk matematika adalah 470 [**?**]. 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 [**?**] 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 keterpenuhan dari sebuah sudoku. Sehingga pemain dapat mengetahui sebuah sudoku memiliki solusi atau tidak.

Hingga saat ini, sudah banyak penelitian yang membahas penyelesaian sudoku secara matematis maupun komputasional. Salah satu metode yang cu-

kup dikenal adalah penyelesaian sudoku dengan memanfaatkan masalah keterpenuhan formula proposisional (*propositional satisfiability problem*). Dengan pendekataan ini, syarat-syarat yang harus dipenuhi oleh suatu sudoku dimodelkan dengan satu atau lebih formula logika proposisi [**?**, **?**]. 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++ [**?**]. Aplikasi akan menggunakan pengujian kotak hitam untuk menilai kualitas aplikasi [**?**]. Interaktif adalah sebuah hubungan dua arah [**?**]. Yang penulis maksud dengan hubungan dua arah adalah sebuah hubungan dimana pengguna dapat berinteraksi dengan aplikasi dengan mudah, berkesan dan memberikan kepuasan serta pengalaman yang menyenangkan.

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 (4×4), (9×9), (16×16).
2. Untuk setiap ukuran sudoku memiliki tiga tingkat kesulitan. Yaitu mudah, menengah, sulit.
3. Program mengeluarkan keterpenuhan dari sudoku.
4. Program mengeluarkan solusi dari sudoku jika pengguna mengiginkan.
5. Aplikasi akan diuji dengan metode pengujian kotak hitam (*black box testing*).

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. Analisa spesifikasi dan Kebutuhan sistem.
2. Translasi spesifikasi sistem.
3. Pembuatan aplikasi.
4. Pengujian aplikasi.
5. Analisis sistem.
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	Analisa spesifikasi dan kebutuhan sistem							
2	Translasi spesifikasi sistem							
3	Pembuatan aplikasi							
4	Pengujian aplikasi							
5	Analisis sistem							
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 [**?**]. 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.

Sebelum sudoku modern berkembang. *Puzzle latin square* terlebih dahulu berkembang yang di repretasikan oleh sebuah matriks berukuran $n \times n$ yang memiliki angka 1*hinggan* 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 [**?**]. Pada 19 November 1892 *Le Siècle* sebuah surat kabar di perancis mempublikasikan *magic square* yaitu sebuah teka-teki yang di repretasikan oleh sebuah matriks berukuran 9×9 yang memiliki dua buah submatriks berukuran 3×3. Tujuan dari *magic square* adalah pemain mengisi baris dan kolom sehingga nilai penjumlahan setiap baris dan kolom itu sama. Sudoku modern sendiri lahir dari Howard Garns yang dipublikasikan di *Dell Magazines* pada 1979 [**?**].

2.2 *Satisfiability Problem* (SAT)

Masalah SAT (SAT *problem*) adalah salah satu masalah penting dalam logika komputasional [**?**]. 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*. Masalah SAT merupakan masalah NP-complete, hingga saat ini tidak terdapat algoritma yang efisien untuk memecahkan masalah tersebut.

2.2.1 Sudoku Sebagai Masalah SAT

Sudoku memiliki beberapa aturan contohnya untuk sudoku berukuran 9×9 yaitu :

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

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

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 9×9 memiliki sel dengan ukuran 3×3 maka terdapat 9 sel pada sudoku sehingga :

- blok pertama akan dimulai dengan sel $(1,1)$ dan di akhiri pada sel $(3,3)$
- blok kedua akan dimulai dengan sel $(4,1)$dan di akhiri pada sel $(6,3)$
- blok ketiga akan dimulai dengan sel $(7,1)$dan di akhiri pada sel $(9,3)$
- blok keempat akan dimulai dengan sel $(1,4)$ dan di akhiri pada sel $(3,6)$
- blok kelima akan dimulai dengan sel $(4,4)$dan di akhiri pada sel $(6,6)$
- blok keenam akan dimulai dengan sel $(7,4)$dan di akhiri pada sel $(9,6)$
- blok ketujuh akan dimulai dengan sel $(1,7)$ dan di akhiri pada sel $(3,9)$
- blok kedelapan akan dimulai dengan sel $(4,7)$dan di akhiri pada sel $(6,9)$
- blok kesembilan akan dimulai dengan 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 $\bigwedge_{r=1}^{9}\bigwedge_{c=1}^{9}\bigwedge_{n=1}^{8}\bigwedge_{i=n+1}^{9} (\neg p(r,c,n) \vee \neg p(r,c,i))$.

2.3 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. Pertama baris komentar dapat ditulis dengan : `c <komentar>`
3. Lalu banyak variabel dan klausa`p cnf <banyak-varian> <banyak-klausa>`
4. Lalu klausa:
    - Setiap variabel di representasikan dengan sebuah integer ≥ 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 : $((x_1 \vee x_2) \wedge \neg x_3)$
Ditulis:

```
c Contoh formula satisfiable
p cnf 3 2
1 2 0
-3 0
```

Contoh keluaran SAT *solver*:
Formula: $((x_1 \vee x_2) \wedge \neg x_3)$

Keluaran:

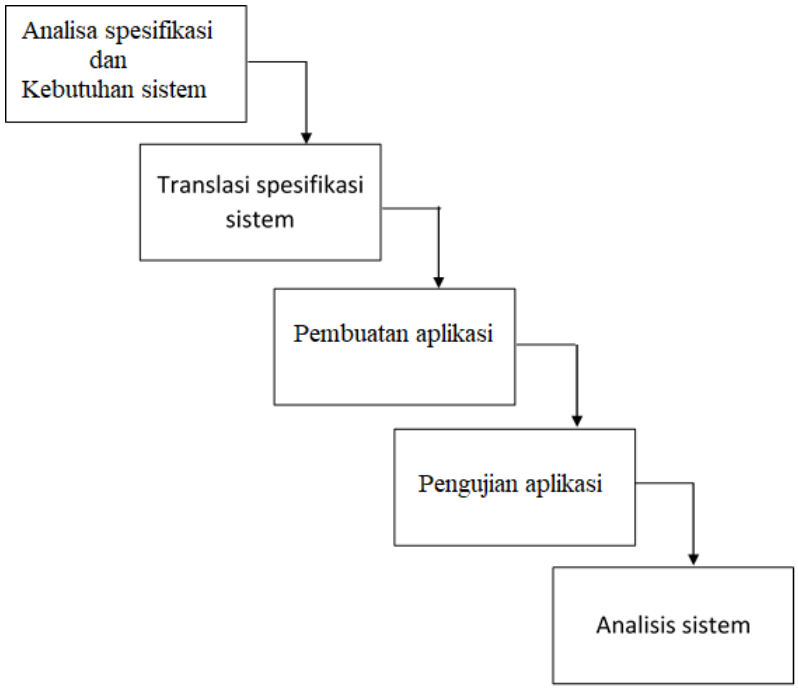
```
SAT
1 -2 -3 0
```

Untuk penggunaan minisat dapat ditulis dengan: `./minisat namaFileMasukan namaFileKeluaran`

Bab III

Metodologi

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



Gambar 3.1: Metodologi.

3.1 Analisa spesifikasi dan Kebutuhan sistem

Menganalisa spesifikasi-spesifikasi pada teka-teki sudoku, lalu membuat kebutuhan dari aplikasi sistem yang akan mengeluarkan *use case* aplikasi.

3.2 Translasi spesifikasi sistem

Aturan-aturan yang telah ditranslasikan dalam CNF di translasikan pada bahasa python dan dibuatkan aturan DIMACSnya seberti berikut:

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) + ' 0\n'
                global klausa
                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 aturan1(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) + ' 0\n'
                global klausa
                klausa += 1
        return dimacs
```

3. Setiap submatriks atau blok 3×3 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 aturan1(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)
+ ' 0\n'
                            global klausa
                            klausa += 1
        return dimacs
```

Setiap sel memuat paling banyak satu bilangan antara 1 hingga 9 :

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 aturan1(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'
                    global klausa
                    klausa += 1
        return dimacs
```

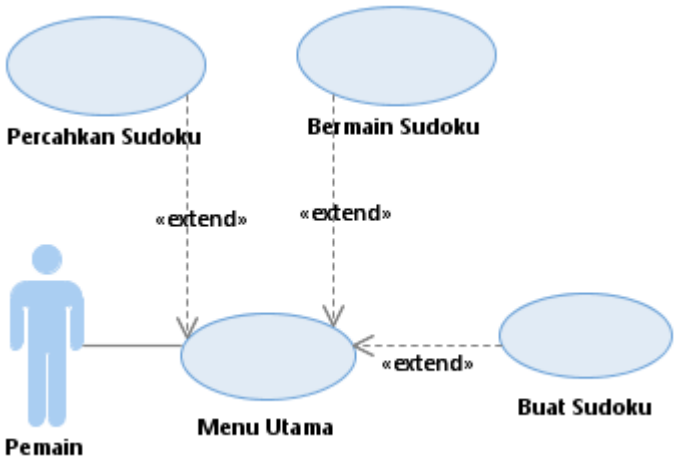
3.3 Pembuatan aplikasi

Spesifikasi yang sudah dalam bentuk CNF akan digunakan pada *SAT Solver*. Lalu *use case* yang didapat akan dibuat fungsionalitasnya. Pembuatan aplikasi menggunakan *waterfall design* tahapan-nya berupa:

3.3.1 Spesifikasi Aplikasi

Spesifikai aplikasi adalah sebagai berikut:

1. Bermain sudoku.
2. Membuat sudoku.
3. Memecahkan sudoku.



Gambar 3.2: *Use Case Diagram.*

3.3.2 *Analysis*

Dalam *analysis*, penulis menjabarkan mulai dari hal mendasar yaitu kendala global dalam pengembangan perangkat lunak yang sudah ada. Adapun masalahnya adalah minimnya aplikasi pemecah sudoku yang interaktif.

3.3.3 *Design*

Fitur yang akan ada pada aplikasi tersebut adalah sebagai berikut

1. Bermain sudoku.
2. Membuat sudoku.
3. Pecahkan sudoku.

3.3.4 Implementasi

Lalu dalam tahap *coding* penulis akan menggunakan bahasa pemograman python dengan SAT *solver* [**?**] sebagai dasarnya. Setelah itu akan ditambahkan GUI dan *use case* yang telah di buat.

3.4 Pengujian aplikasi

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

3.5 Analisis sistem

Menganalisa kelayakan kualitas dari aplikasi berdasarkan hasil dari pengujian.

Lampiran

Pada bab ini akan berisi lampiran dari pengerjaan tugas akhir, contohnya skrip NuSMV.