BAB III
ANALISIS DAN PERANCANGAN APLIKASI

3.1 Metode Penelitian

Metode penelitian yang digunakan dalam penelitian diuraikan sebagai berikut

1. Studi Literatur

Studi literatur dilakukan dengan mempelajari teori dan konsep yang berkaitan dengan pokok bahasan penelitian. Teori yang dipelajari diantaranya : *Nondeterministic Finite Automata*, konsep penerjemahan bahasa Inggris ke bahasa Indonesia, pengenalan struktur kalimat (*grammar*) dan jenis kalimat (*tenses*) yang ada, serta berbagai konsep pendukung lainnya. Referensi yang digunakan untuk membangun aplikasi ini berupa buku, jurnal ilmiah, artikel, dan lain – lain.

2. Perancangan Aplikasi

Perancangan awal aplikasi dilakukan sebelum aplikasi dibangun, perancangan aplikasi yang dilakukan berupa rancangan urutan pembacaan struktur kalimat (*grammar*) menggunakan konsep *Nondeterministic Finite Automata* dan rancangan *interface*.

3. Pembangunan Aplikasi

Pembangunan Aplikasi dilakukan dengan mengimplementasikan rancangan dan hasil studi literatur seperti yang telah disebutkan sebelumnya dengan menggunakan Microsoft Visual Studio 2008 untuk pembangunan *interface* dan Microsoft SQL Server 2012 sebagai penampung basis datanya.

4. Uji Coba dan Evaluasi

Uji coba dan evaluasi akan dilakukan terhadap aplikasi setelah aplikasi selesai dibangun disertai dengan evaluasi dari hasil yang didapatkan dari uji coba. Uji coba aplikasi bertujuan untuk memperlihatkan seberapa tinggi akurasi metode *Nondeterministic Finite Automata* dalam mengidentifikasi dan menerjemahkan kalimat bahasa Inggris ke bahasa Indonesia.

3.2 Analisis Aplikasi

Pada penelitian ini diperlukan dua buah komponen utama, yakni komponen yang berguna sebagai *interface* dan komponen yang menampung seluruh kata-kata yang menjadi basis data penerjemahan kata. Aplikasi dibangun sebagai sebuah *desktop application* yang diharapkan dapat digunakan (*compatible*) di seluruh jenis komputer dan laptop yang dapat menjalankan aplikasi **.exe.*

Desktop application ini dibuat dengan menggunakan bahasa pemrograman C# dan Microsoft Visual Studio 2008 sebagai *interface* dengan basis data menggunakan SQL Server 2012. Kesatuan aplikasi yang dibangun tersebut memiliki kesatuan yang diberi nama TranslatorId yang merupakan singkatan dari *Translator Identificaton,* yakni aplikasi yang dapat mengartikan dan mengidentifikasi jenis kalimat (*tenses*) dari bahasa Inggris ke bahasa Indonesia.

3.2.1 Analisis Nondeterministic Finite Automata

Berikut adalah analisis *state* dari 12 jenis kalimat bahasa Inggris (*tenses*) yang telah dilakukan oleh penulis dalam pembuatan NFA untuk penerjemah dan identifikasi kalimat.

Perlu diketahui sebelumnya, untuk memudahkan pembuatan NFA, dilakukan penyederhanaan bentuk kata dalam bahasa Inggris yang penulisannya dipaparkan sebagai berikut

1. ADJECTIVE = (Adj)
2. ADVERB = (Adv)
3. AUX = (Aux)
4. AUX_PAST = (AuxP)
5. ARTICLE= (Art)
6. CONJUNCTION = (Conj)
7. DETERMINER = (Det)
8. MODAL = (Modal)
9. NOUN = (Noun)
10. NUMBER = (Num)
11. PREPOSITION = (Prep)
12. PRONOUN = (Pron)
13. TOBE = (To)
14. TOBE_PAST = (ToP)
15. VERB_1 = (V1)
16. VERB_2 = (V2)

17. VERB_3 = (V3)

18. VERB_S = (VS)

19. VERB_ING = (Ving)

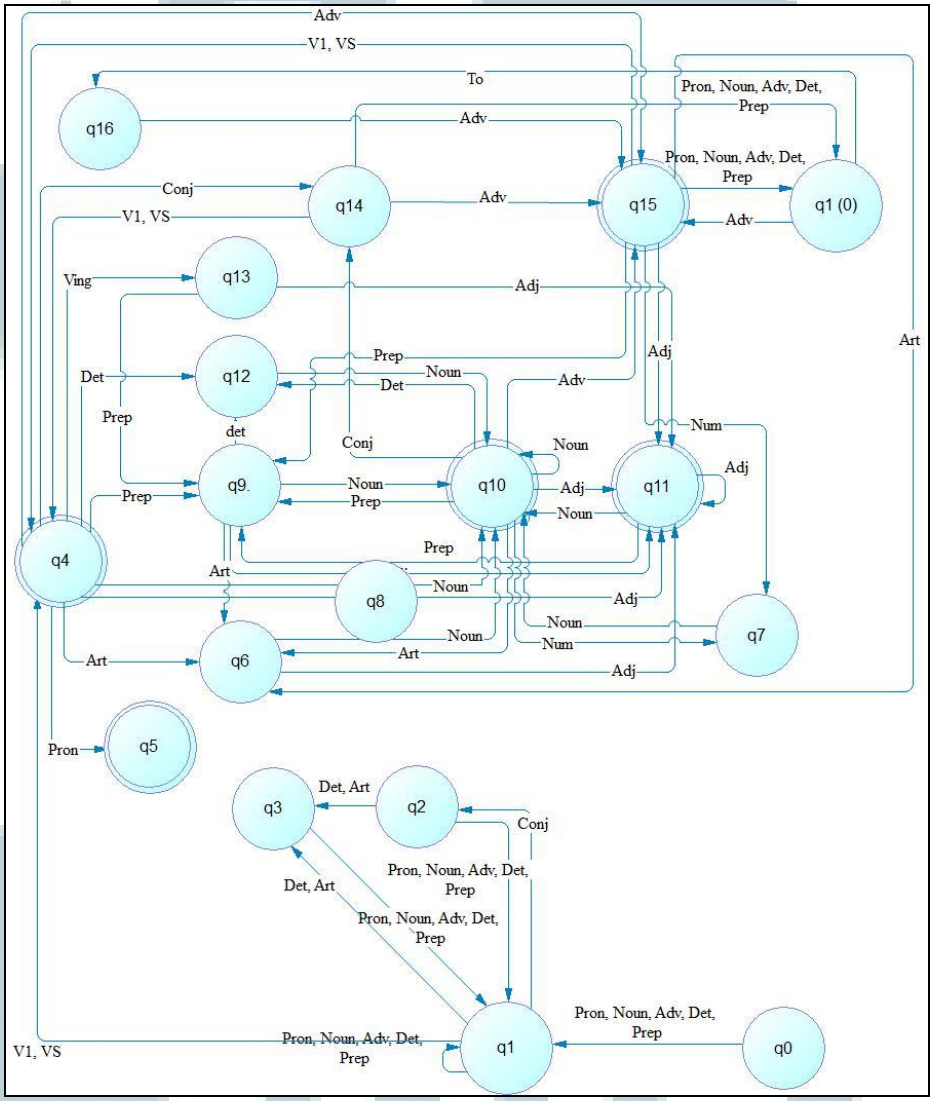
Sedangkan untuk jumlah jenis kalimat bahasa Inggris (*tenses*) yang diteliti berjumlah dua belas dengan bentuk *tenses* sebagai berikut

1. *Simple Present Tense*
2. *Present Continous Tense*
3. *Present Perfect Tense*
4. *Perfect Continous Tense*
5. *Simple Past Tense*
6. *Past Continous Tense*
7. *Past Perfect Tense*
8. *Past Perfect Continous Tense*
9. *Future Simple Tense*
10. *Future Continous Tense*
11. *Future Perfect Tense*
12. *Future Perfect Continous Tense*

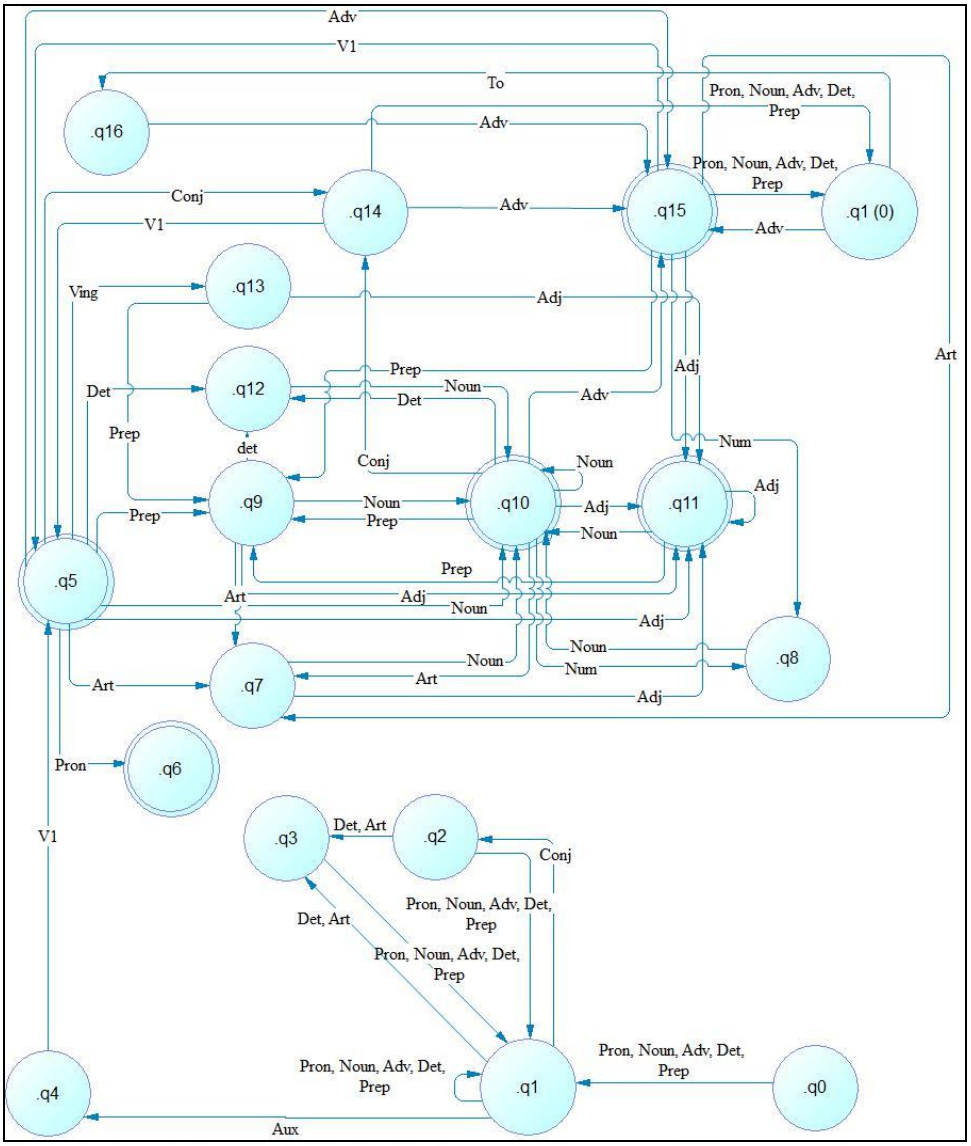
Berikut adalah analisis NFA yang penulis lakukan pada masing-masing tenses yang telah disebutkan di atas. Sebelumnya perlu diketahui, meskipun dalam penulisan diagram *automata* dikelompokan berdasarkan jenis kalimat (*tenses*), namun pada penerapannya seluruh diagram automata adalah saling terkait.

1. Simple Present Tense

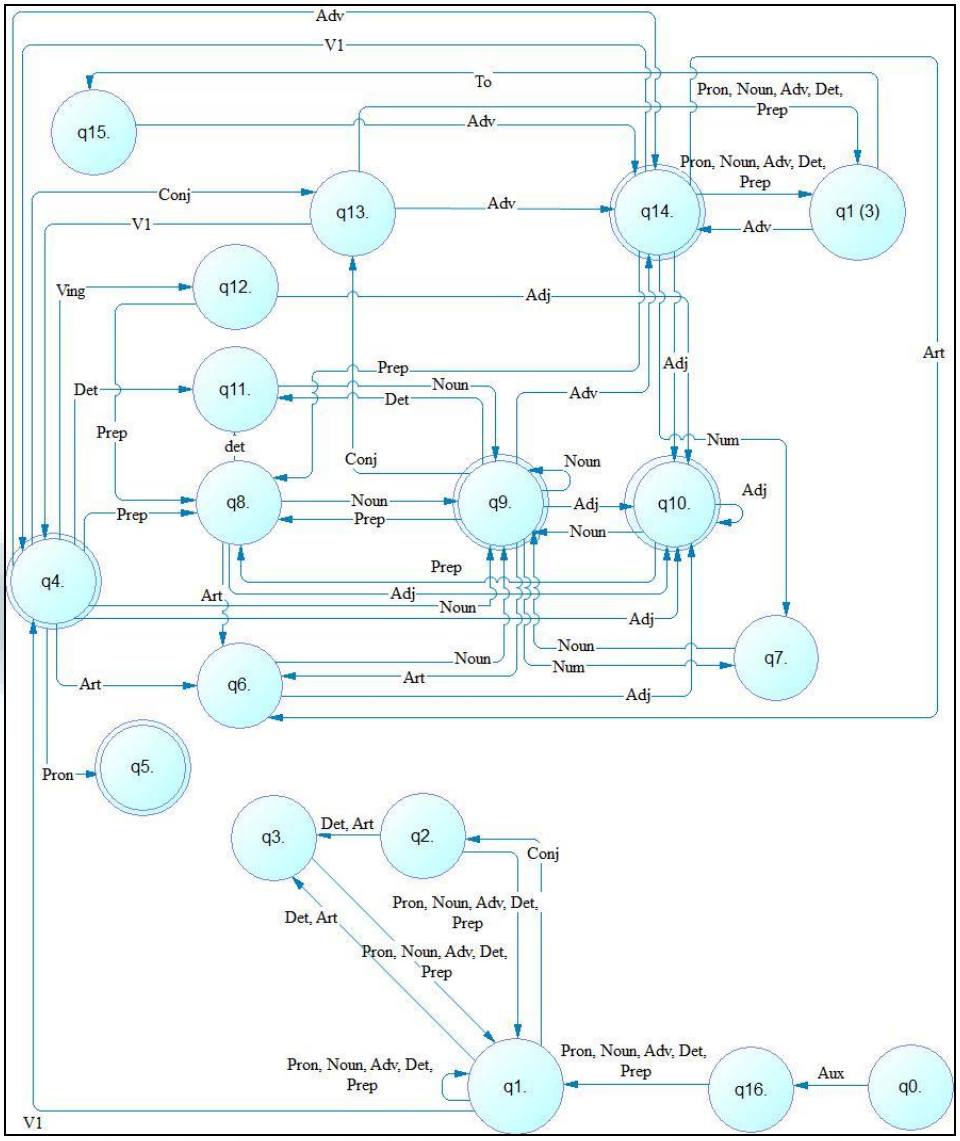
Kalimat *simple present tense* terbagi atas 2 jenis, yakni jenis 1 adalah kalimat *simple present tense* yang memiliki *verb* 1 atau *verb* s (kata kerja bentuk pertama) dan jenis 2 adalah kalimat *simple present tense* tanpa *verb* 1 atau *verb* s (kata kerja bentuk pertama).



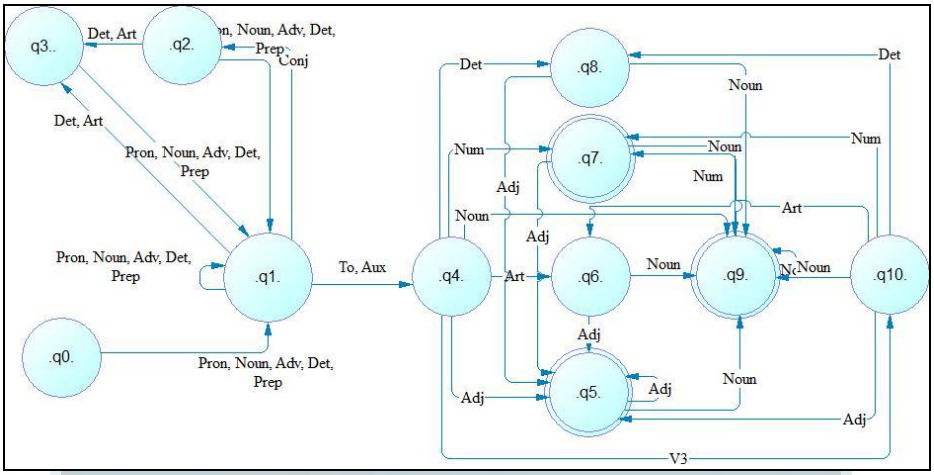
Gambar 3.1 Gambar NFA *Simple Present Tense* jenis 1 (kalimat positif)



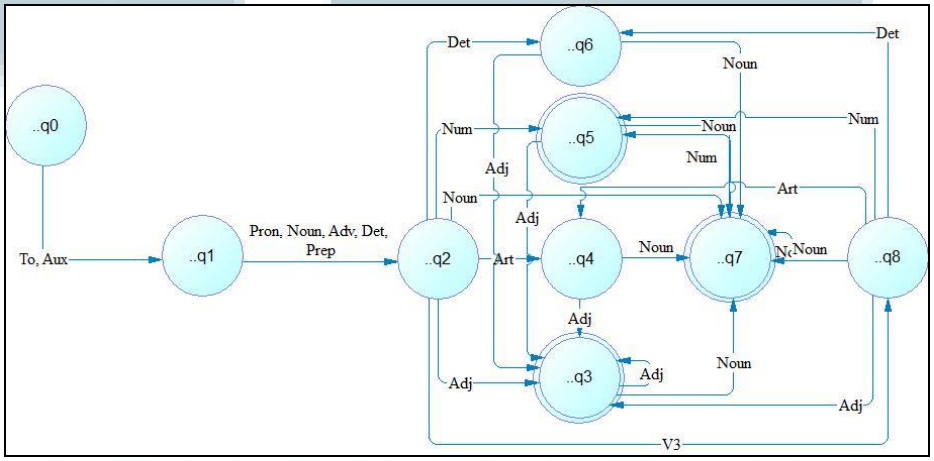
Gambar 3.2 Gambar NFA *Simple Present Tense* jenis 1 (kalimat negatif)



Gambar 3.3 Gambar NFA *Simple Present Tense* jenis 1 (kalimat tanya)

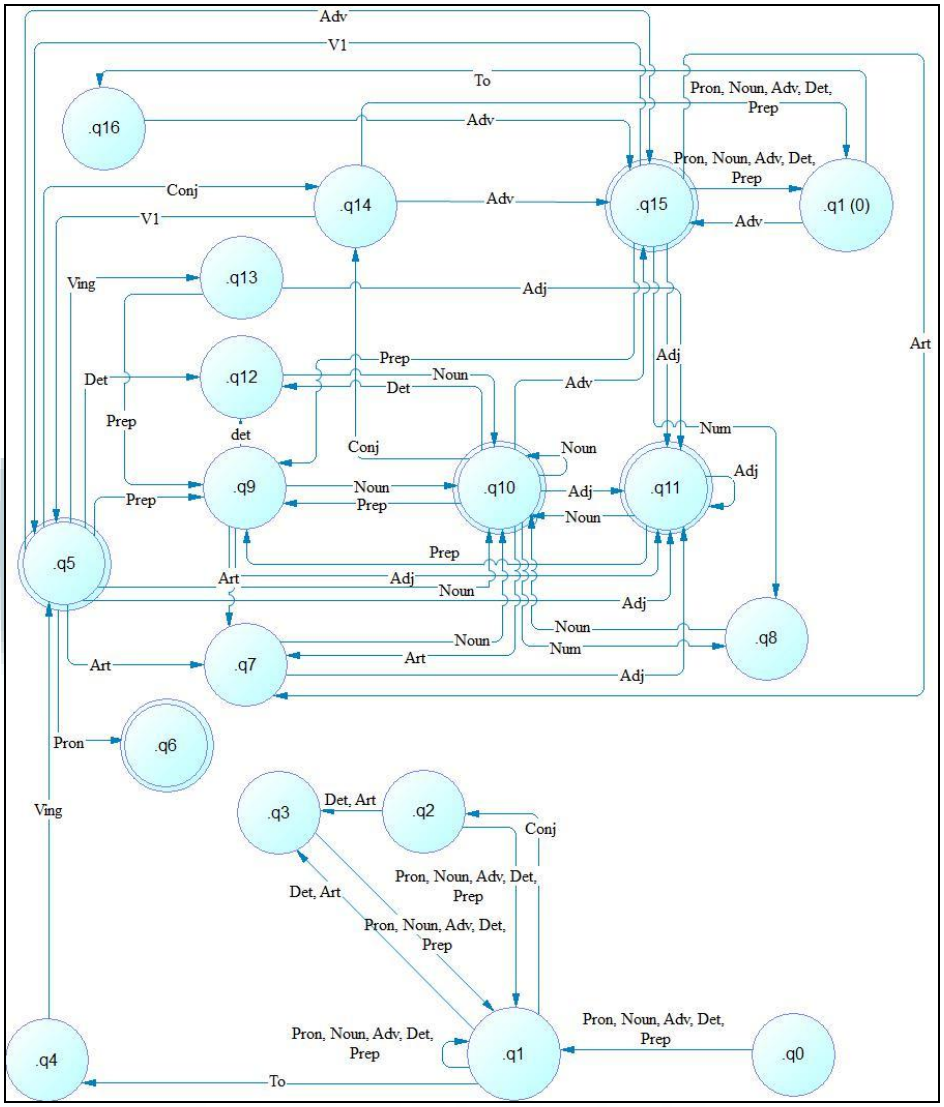


Gambar 3.4 Gambar NFA *Simple Present Tense* jenis 2 (kalimat positif dan negatif)

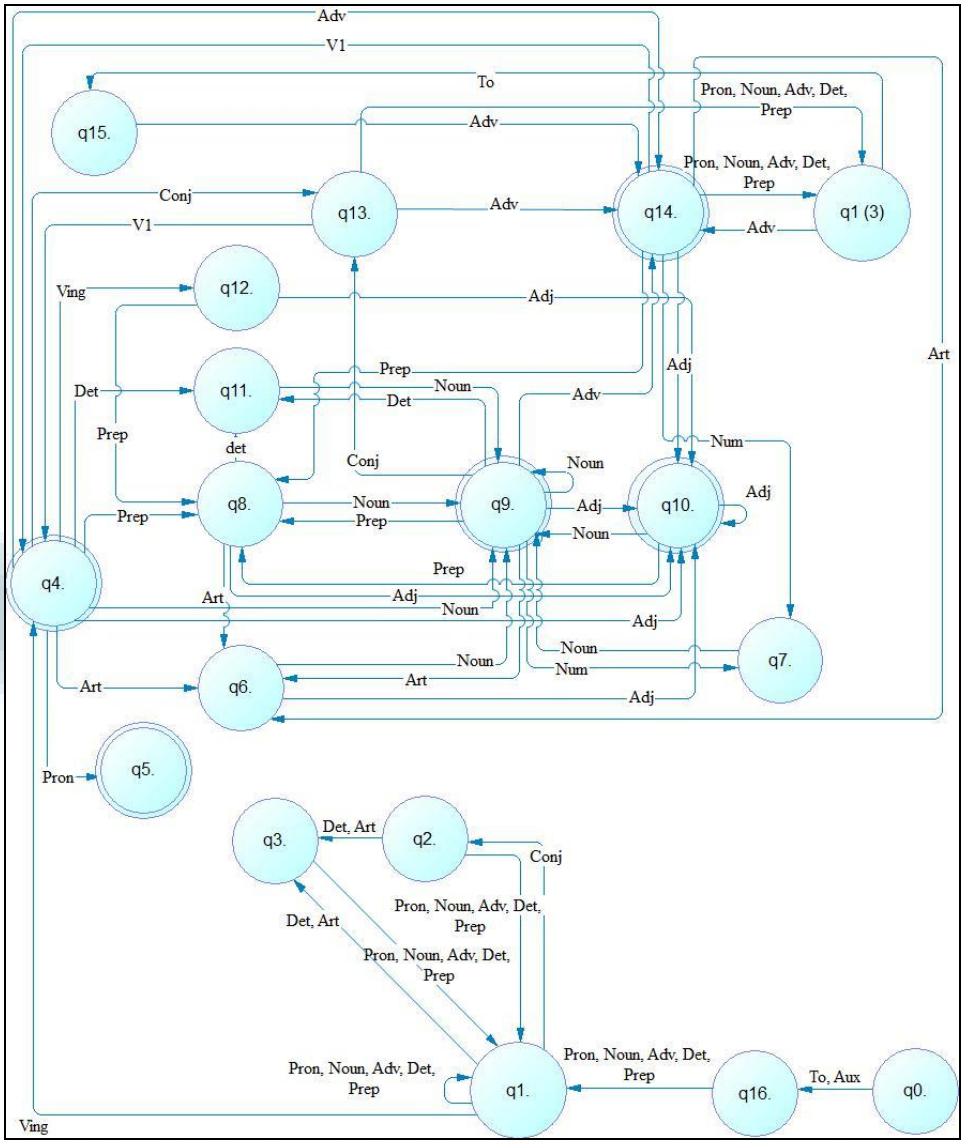


Gambar 3.5 Gambar NFA *Simple Present Tense* jenis 2 (kalimat tanya)

2. Present Continous Tense

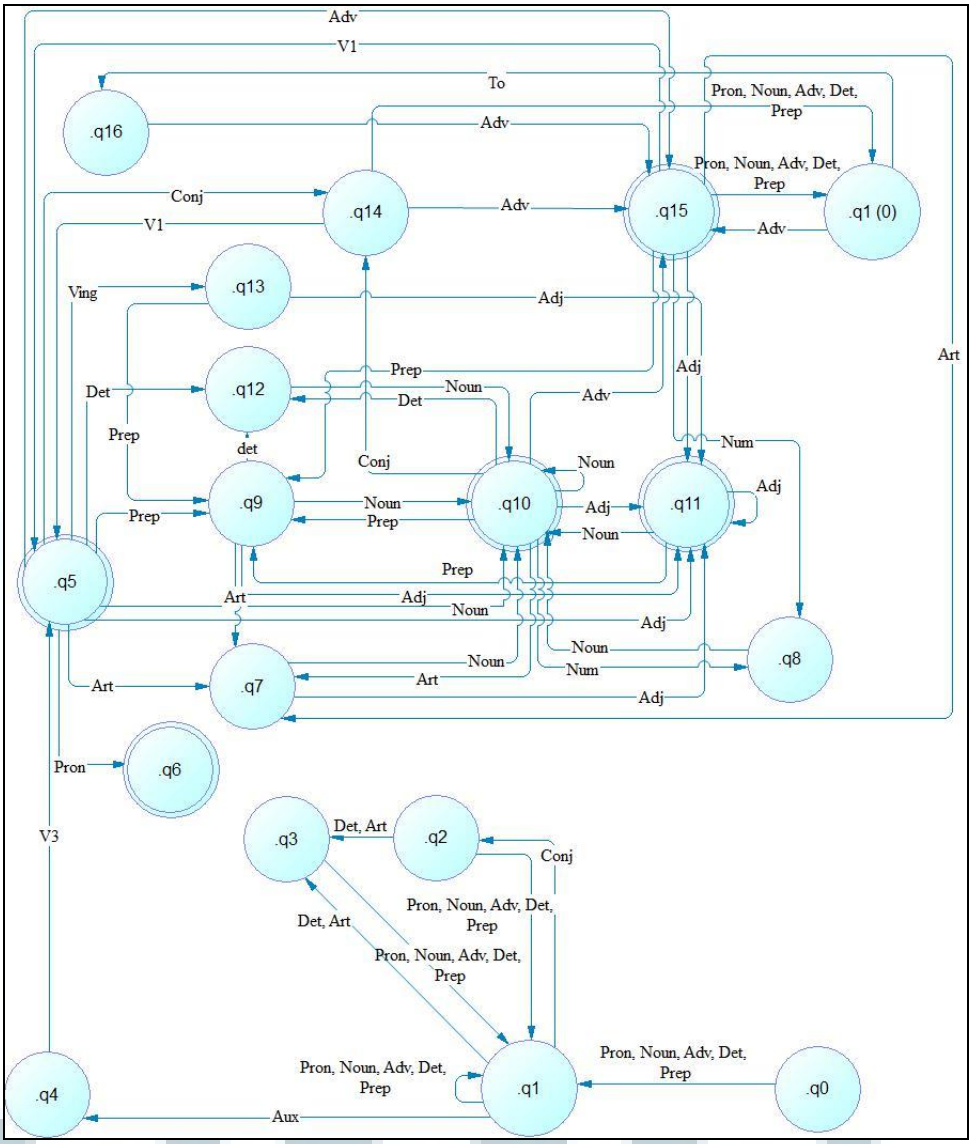


Gambar 3.6 Gambar NFA *Present Continous Tense* (kalimat positif dan negatif)

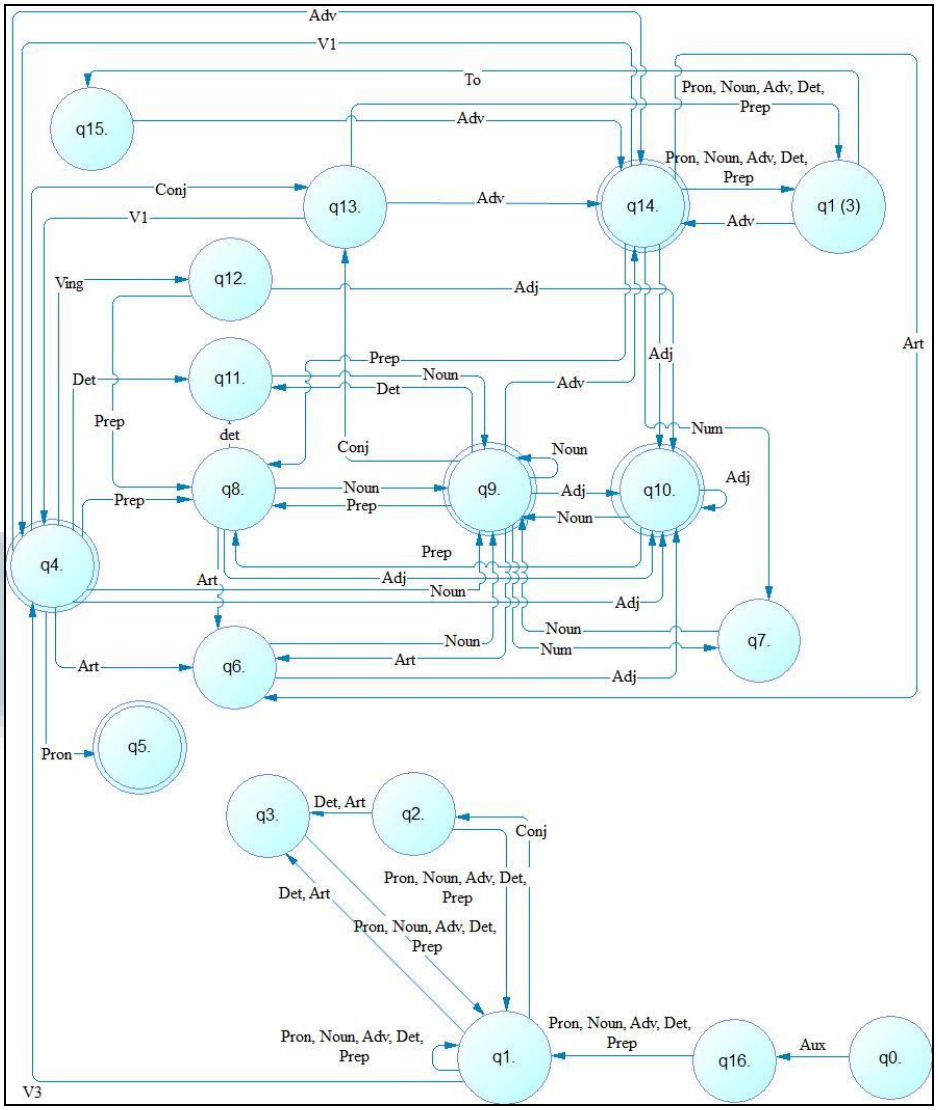


Gambar 3.7 Gambar NFA *Present Continous Tense* (kalimat tanya)

3. Present Perfect Tense

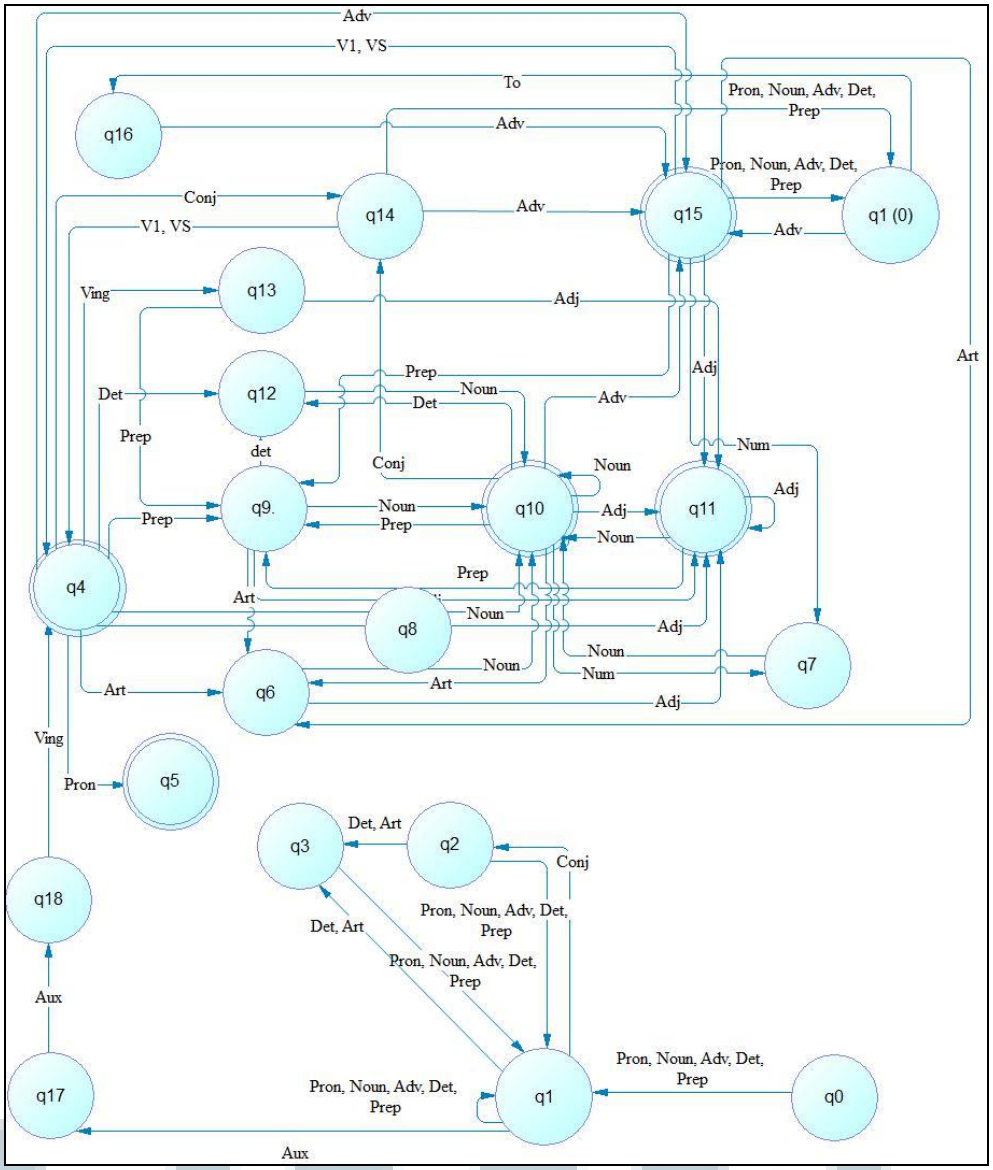


Gambar 3.8 Gambar NFA *Present Perfect Tense* (kalimat positif dan negatif)

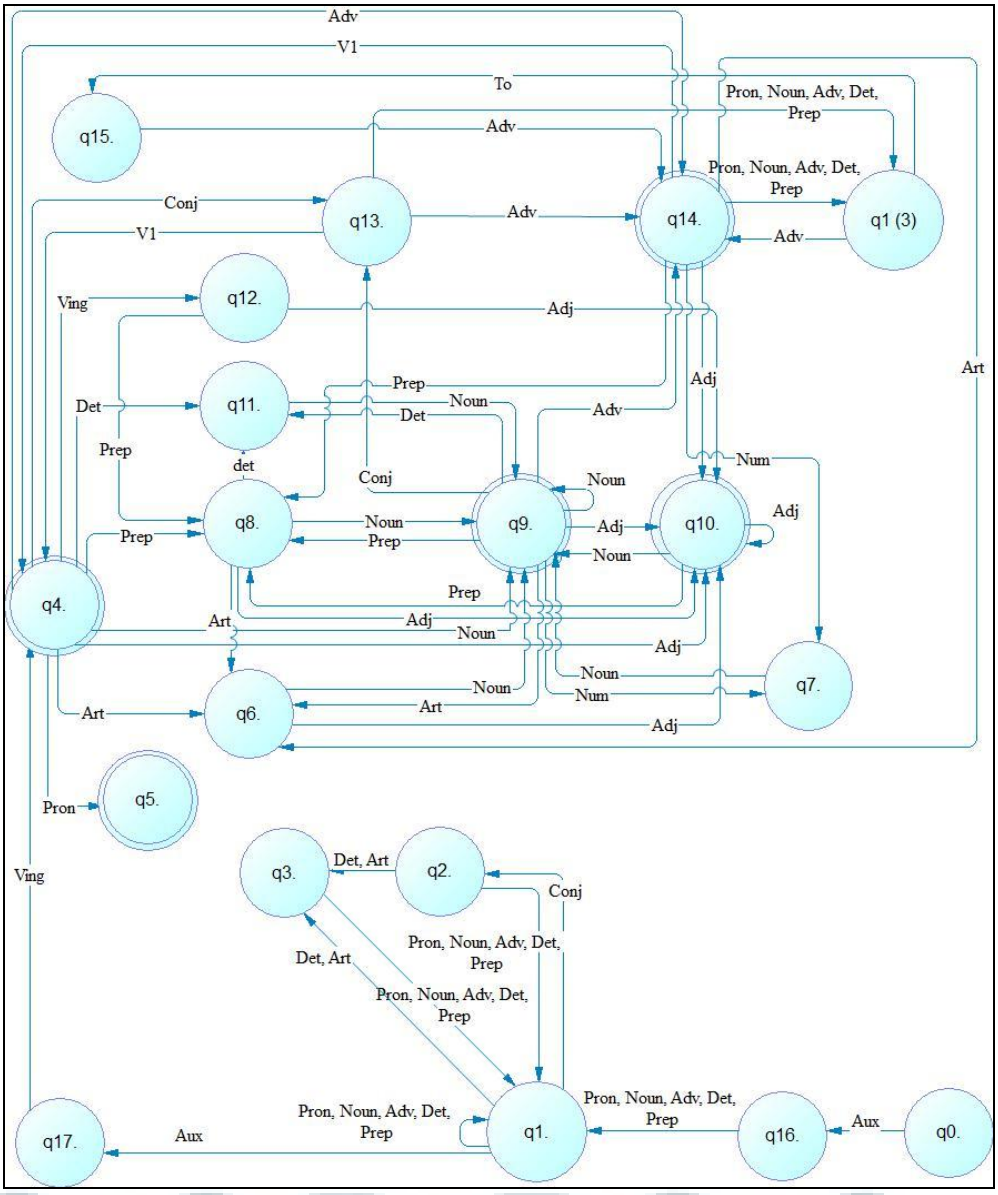


Gambar 3.9 Gambar NFA *Present Perfect Tense* (kalimat tanya)

4. Perfect Continous Tense



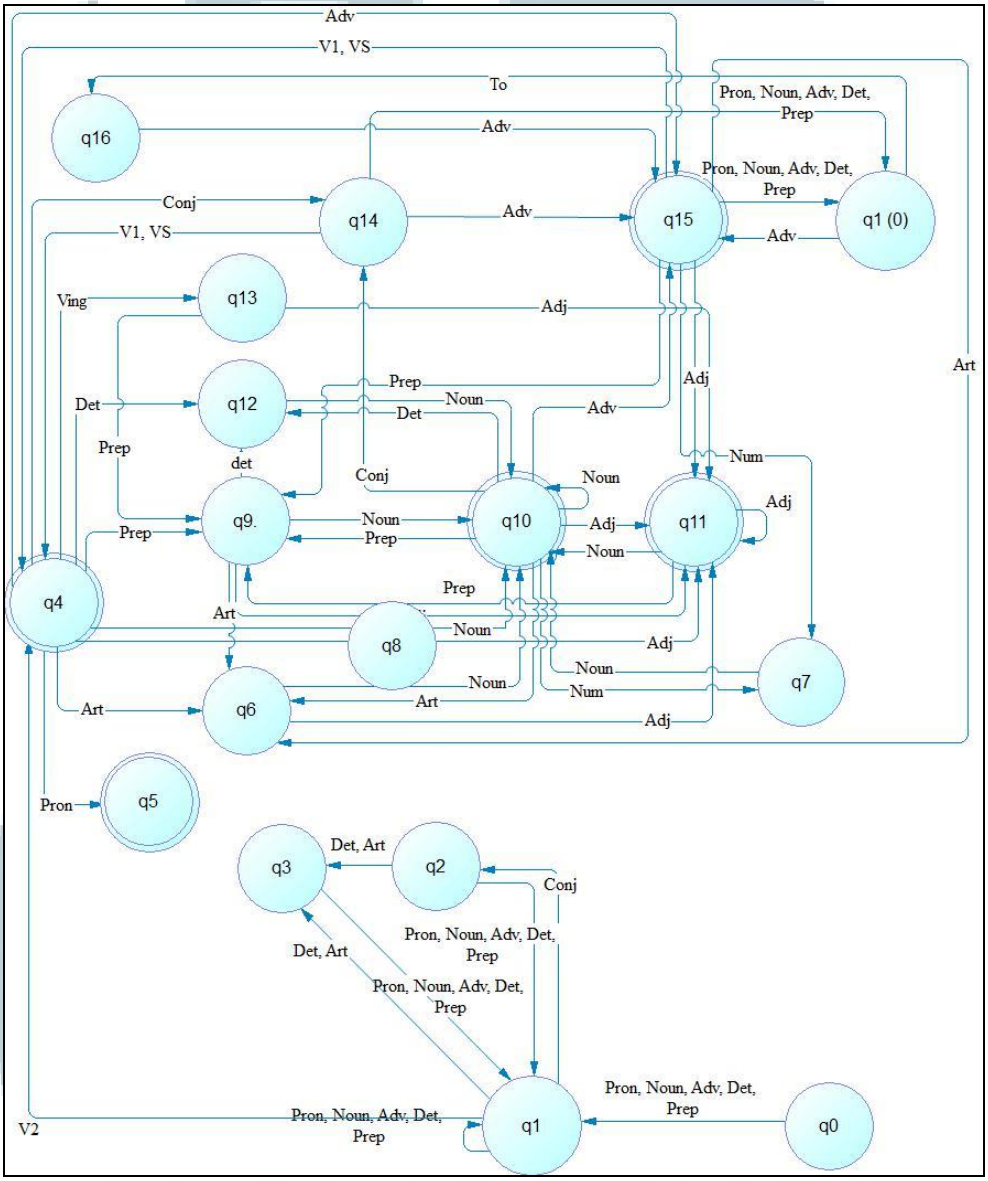
Gambar 3.10 Gambar NFA *Present Perfect Continous Tense* (kalimat positif dan negatif)



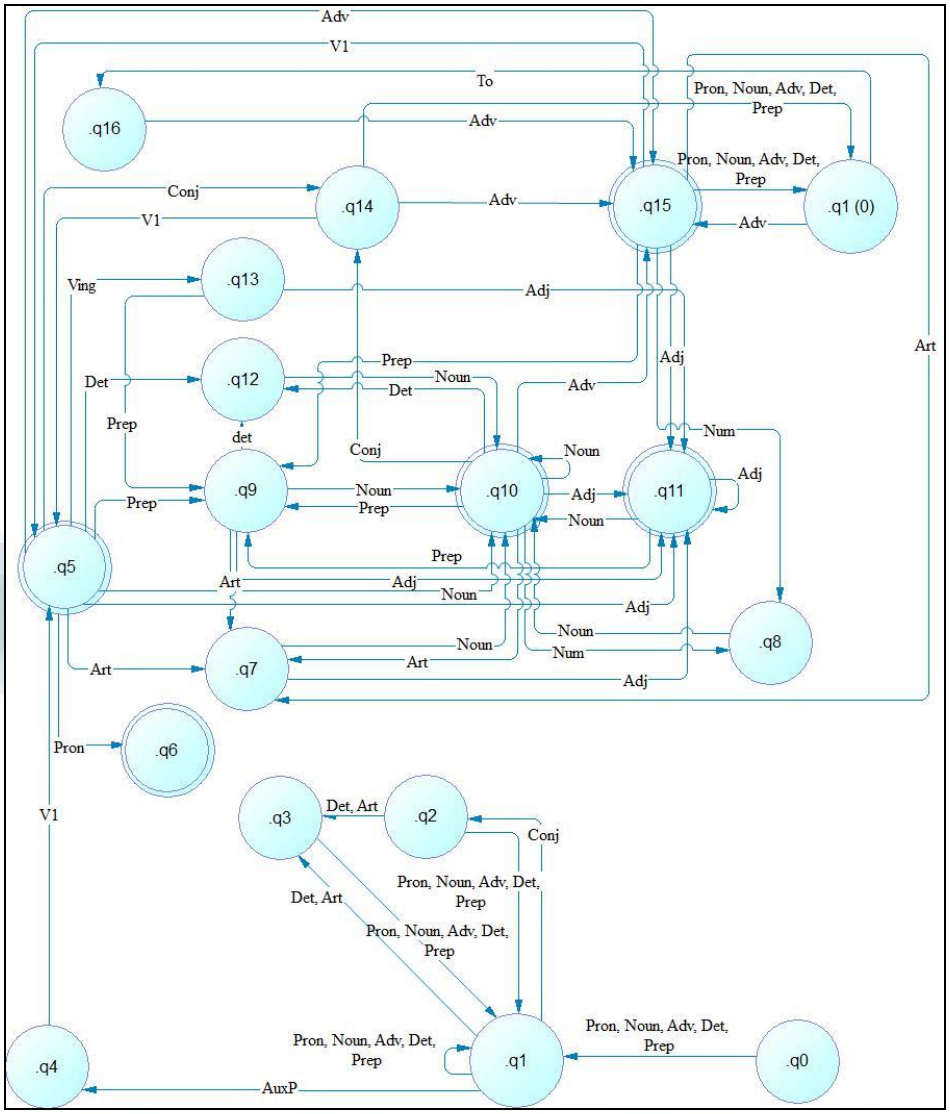
Gambar 3.11 Gambar NFA *Present Perfect Continous Tense* (kalimat tanya)

5. Simple Past Tense

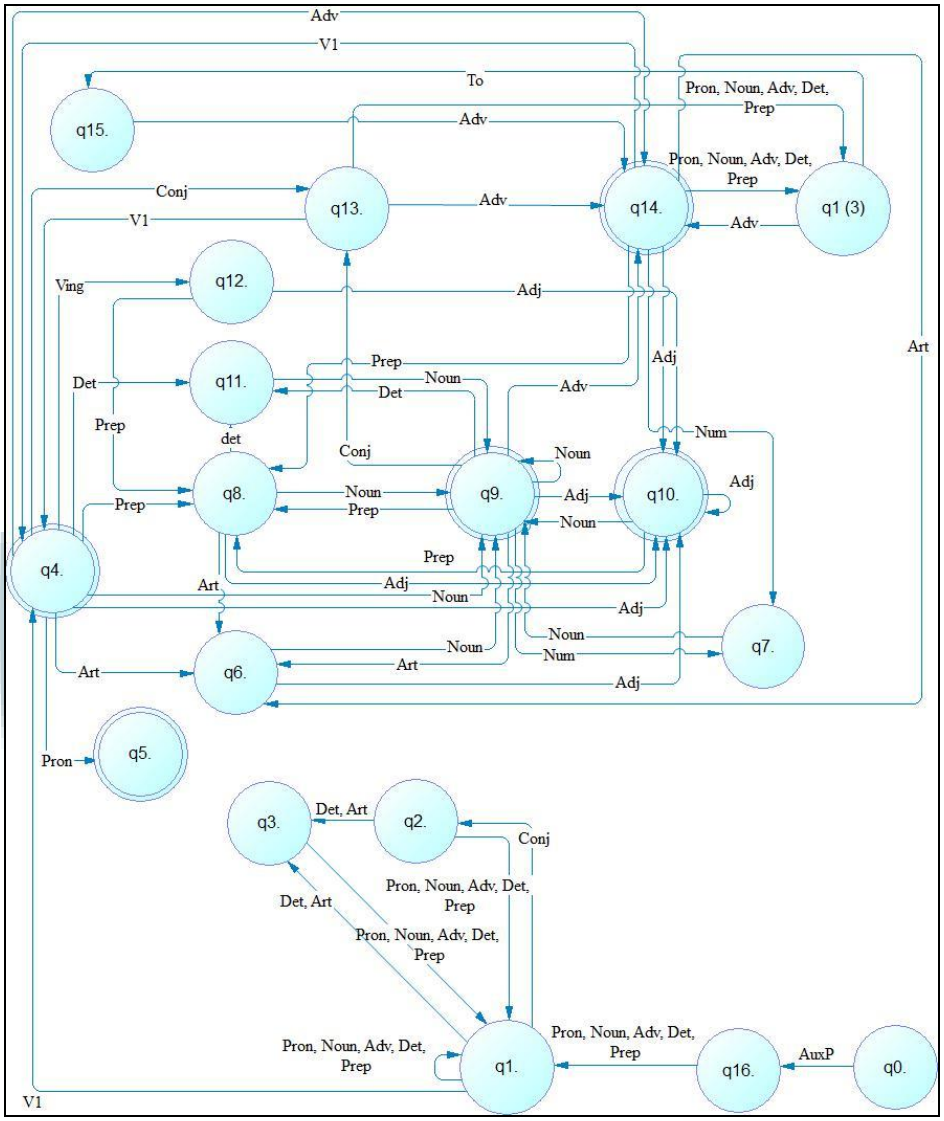
Kalimat *simple past tense* terbagi atas 2 jenis, sama seperti kalimat *simple present tense*, yakni jenis 1 adalah kalimat *simple past tense* yang memiliki *verb* 2 (kata kerja bentuk kedua) dan jenis 2 adalah kalimat *simple past tense* tanpa *verb* 2 (kata kerja bentuk kedua).



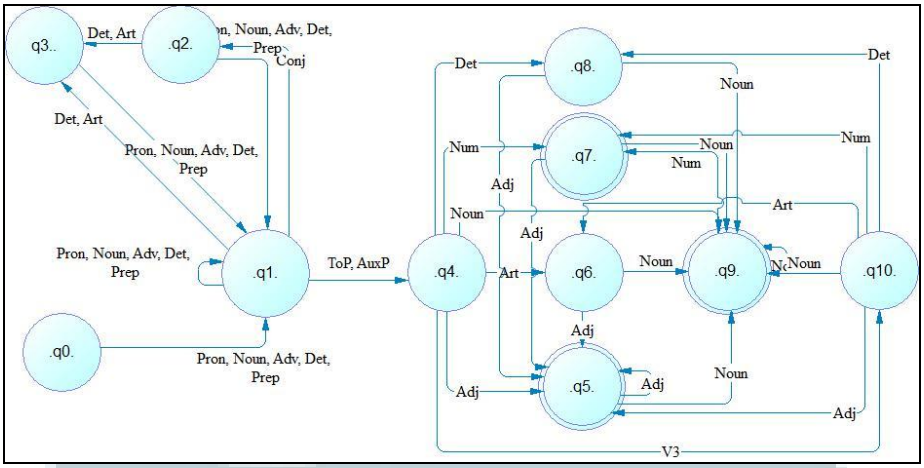
Gambar 3.12 Gambar NFA *Simple Past Tense* jenis 1 (kalimat positif)



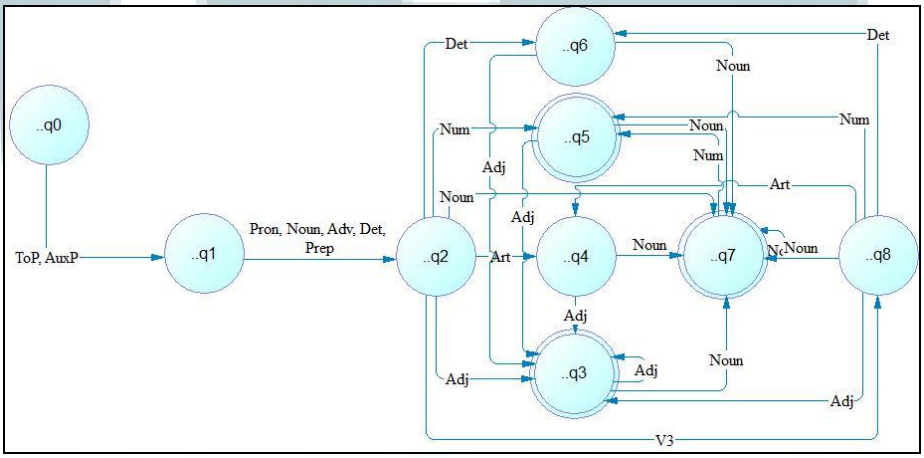
Gambar 3.13 Gambar NFA *Simple Past Tense* jenis 1 (kalimat negatif)



Gambar 3.14 Gambar NFA *Simple Past Tense* jenis 1 (kalimat tanya)

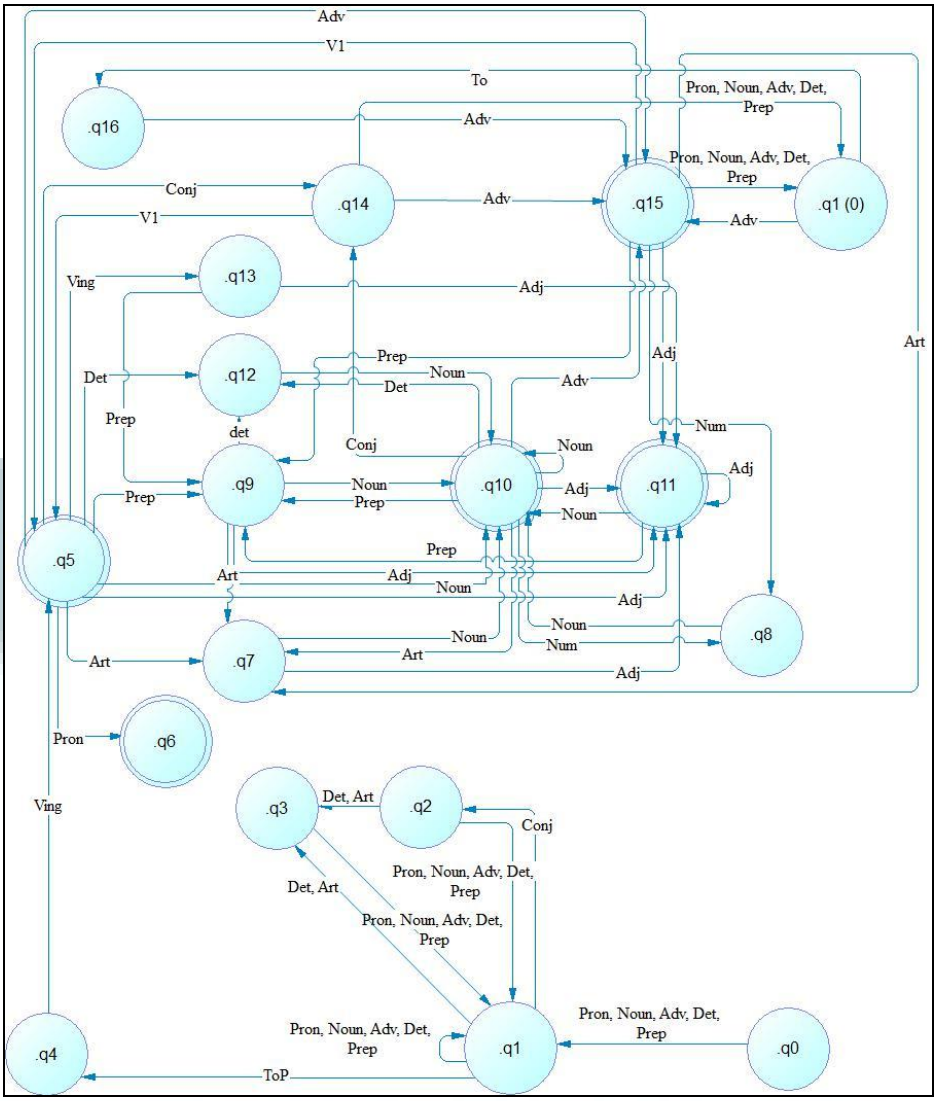


Gambar 3.15 Gambar NFA *Simple Past Tense* jenis 2 (kalimat positif dan negatif)

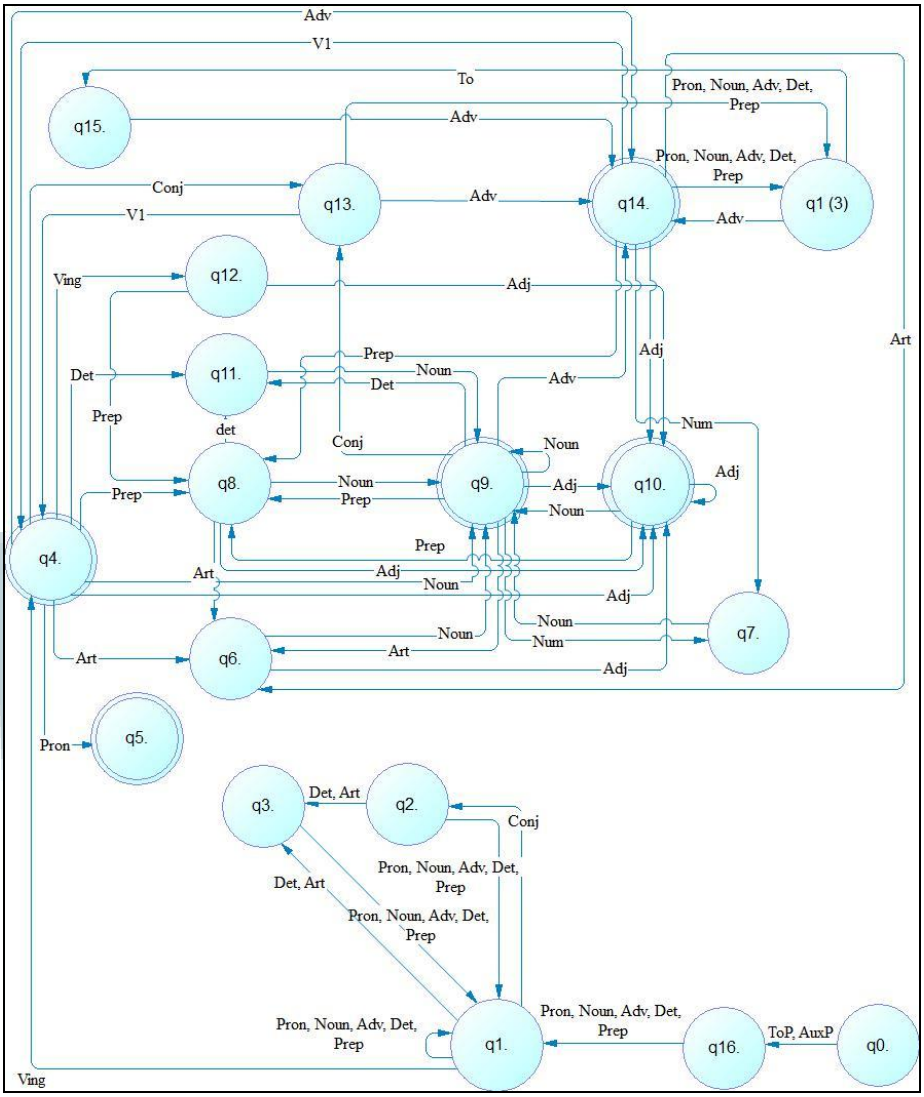


Gambar 3.16 Gambar NFA *Simple Past Tense* jenis 2 (kalimat tanya)

6. Past Continous Tense

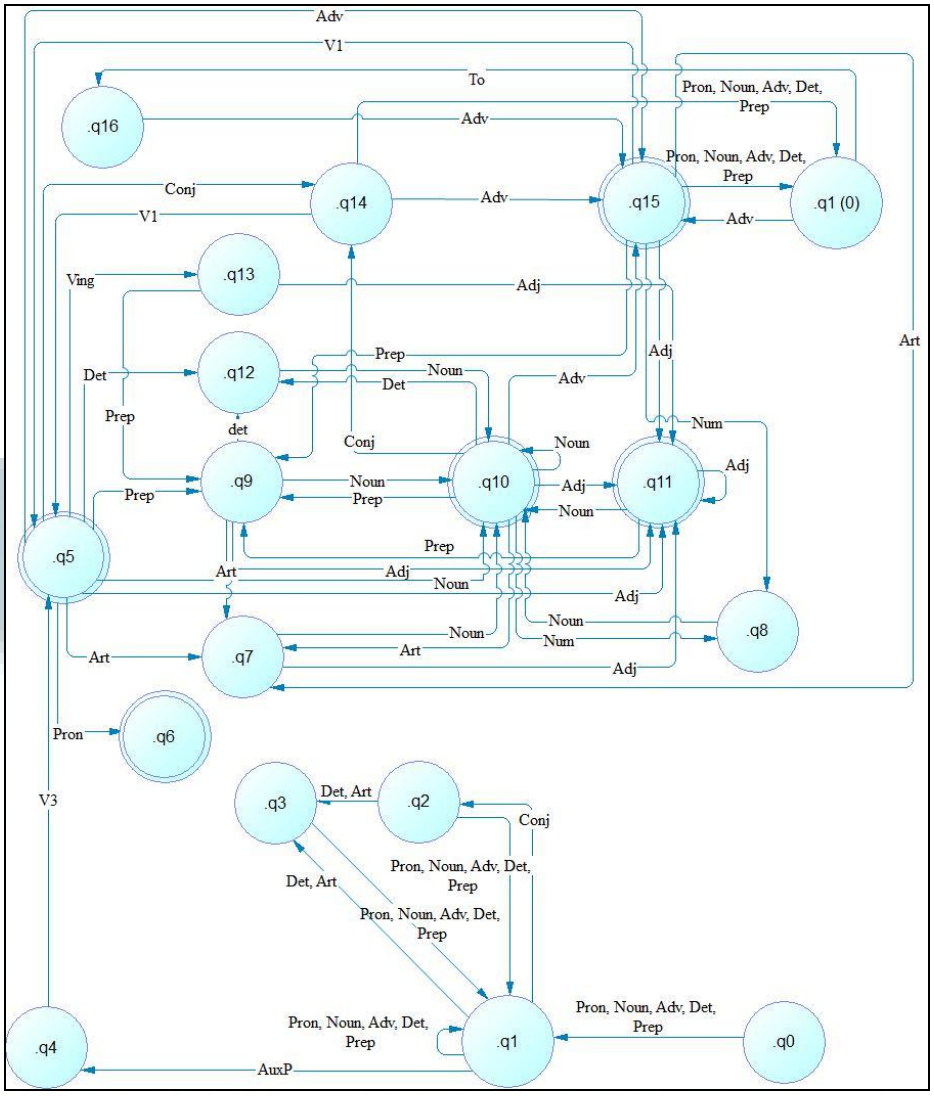


Gambar 3.17 Gambar NFA *Past Continous Tense* (kalimat positif dan negatif)

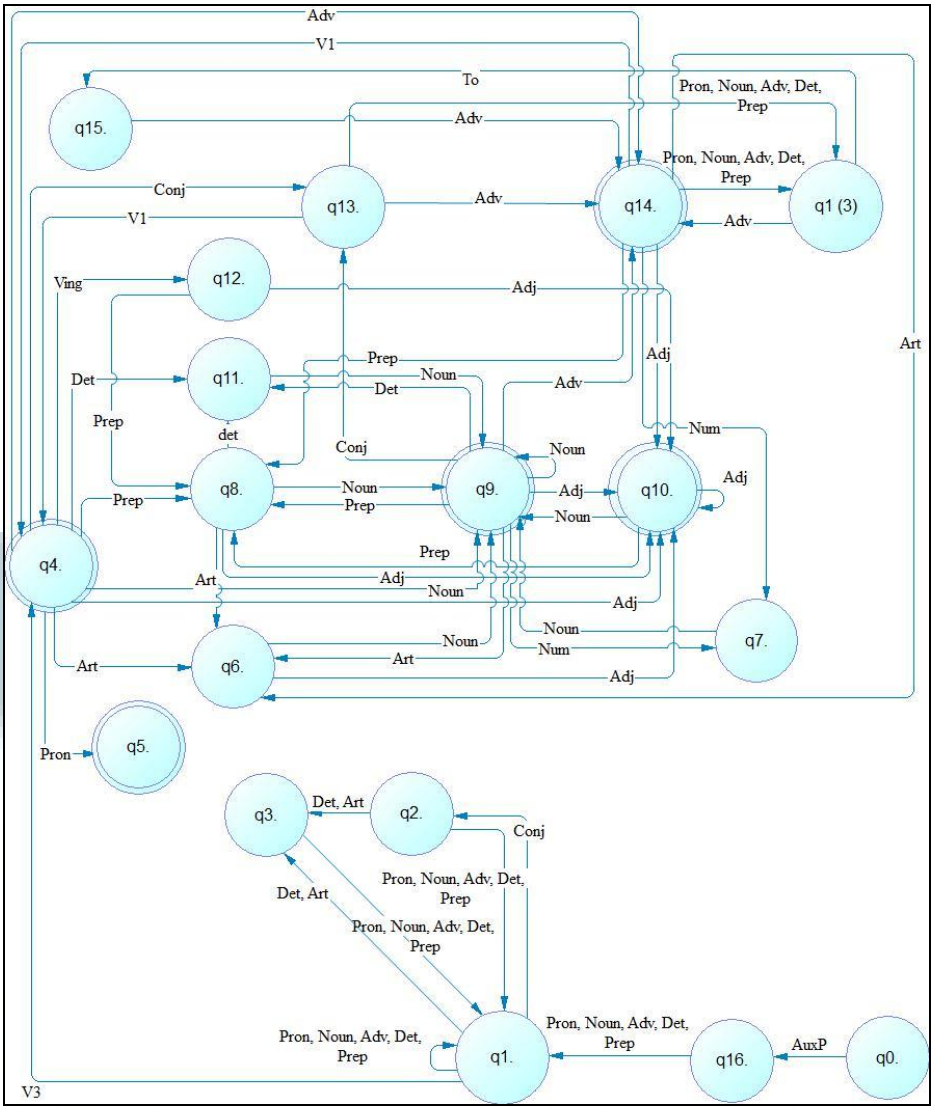


Gambar 3.18 Gambar NFA *Past Continous Tense* (kalimat tanya)

7. Past Perfect Tense

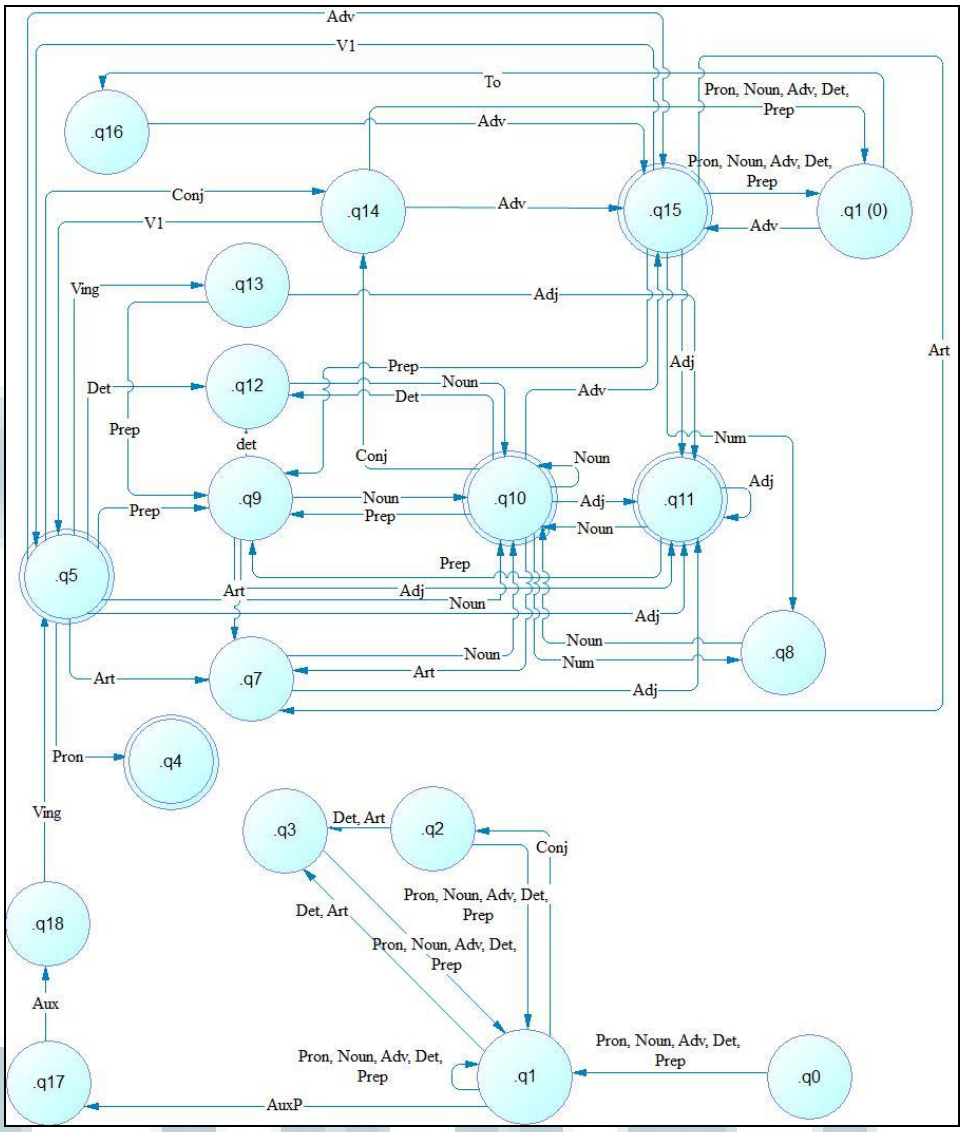


Gambar 3.19 Gambar NFA *Past Perfect Tense* (kalimat positif dan negatif)

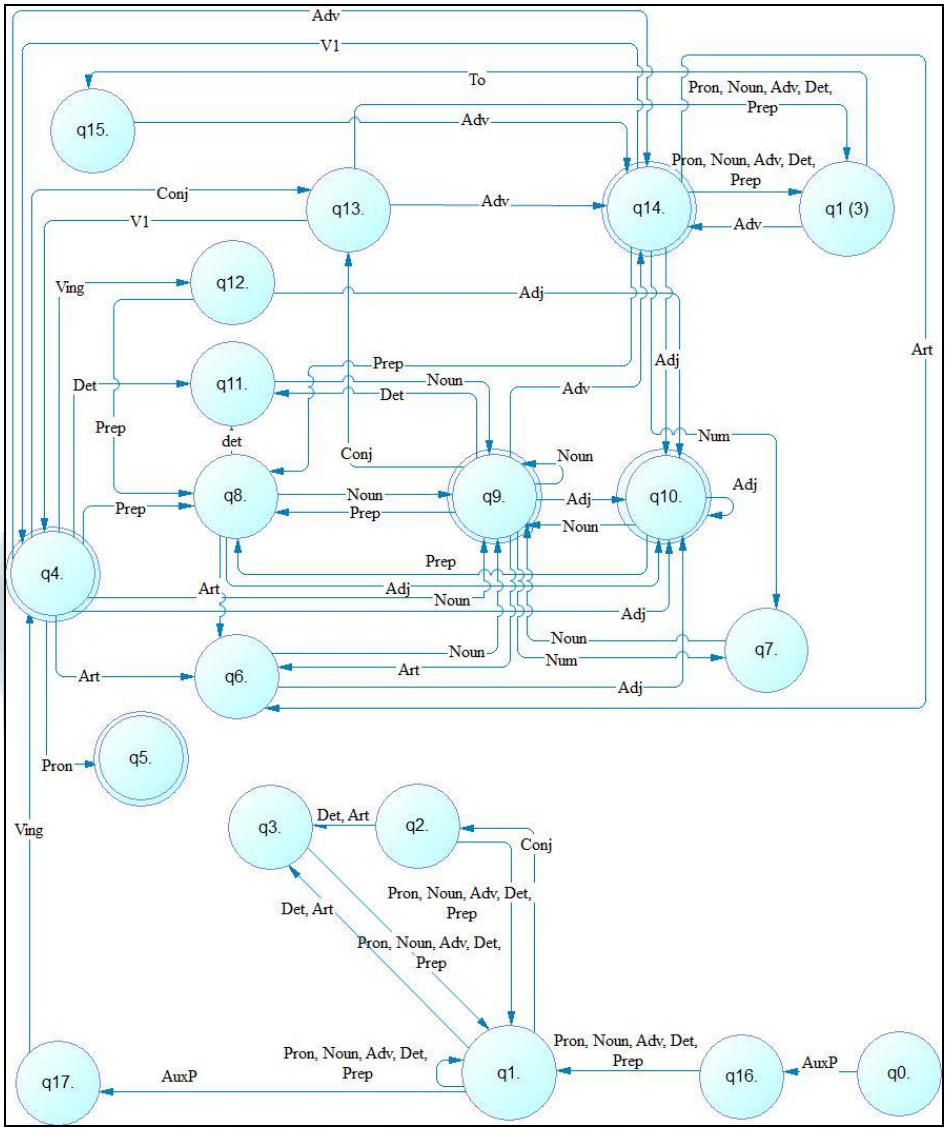


Gambar 3.20 Gambar NFA *Past Perfect Tense* (kalimat tanya)

8. Past Perfect Continous Tense

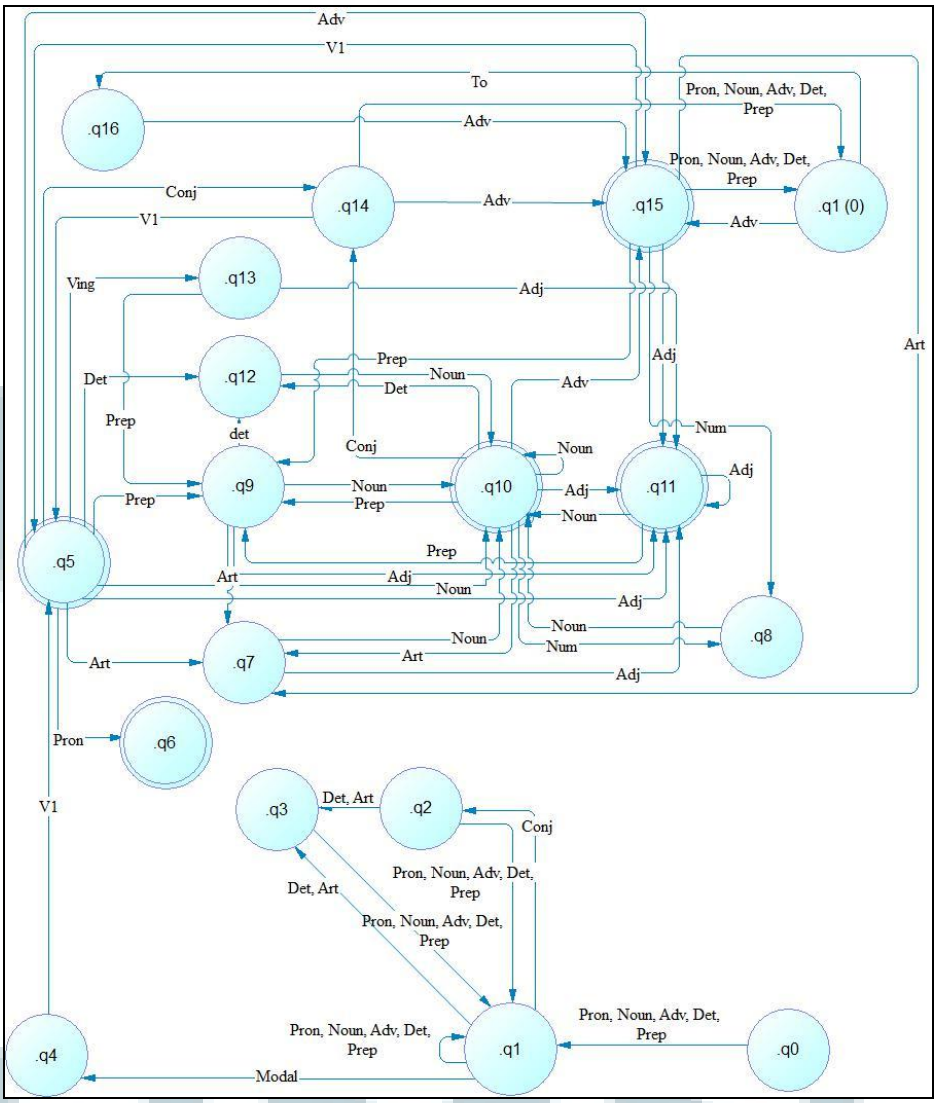


Gambar 3.21 Gambar NFA *Past Perfect Continous Tense* (kalimat positif dan negatif)

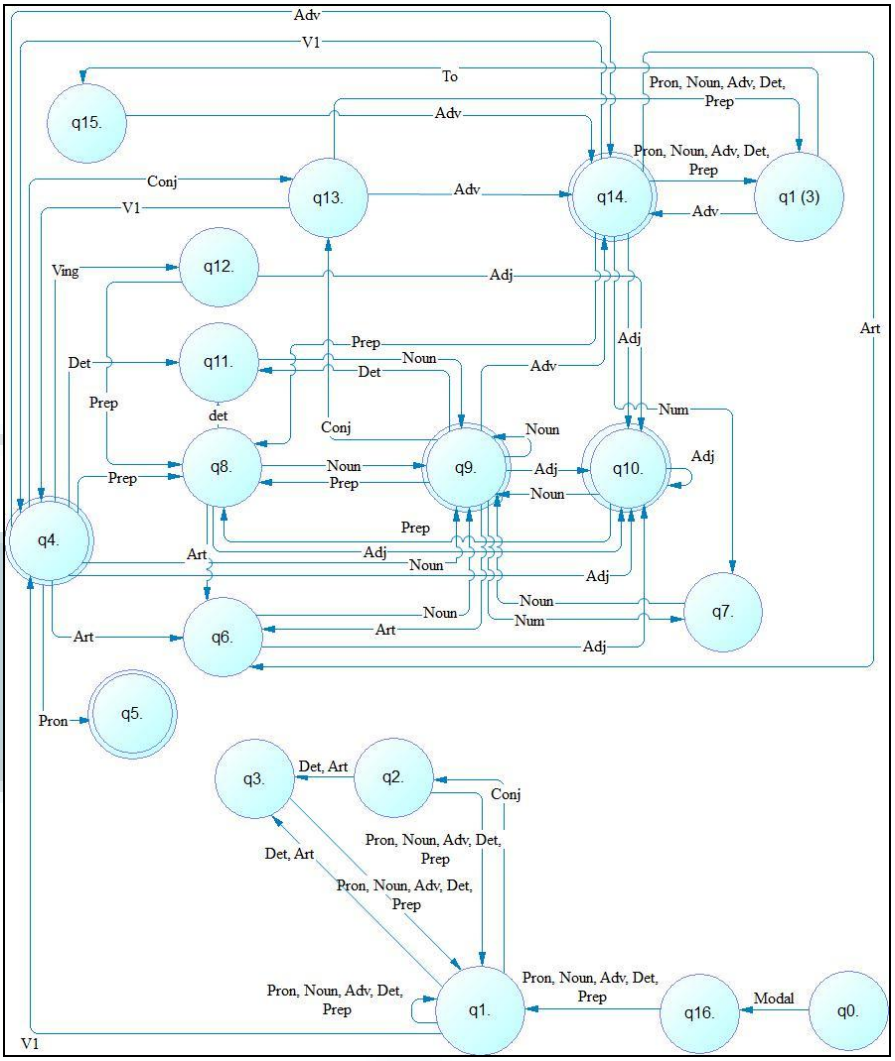


Gambar 3.22 Gambar NFA *Past Perfect Continous Tense* (kalimat tanya)

9. Future Simple Tense

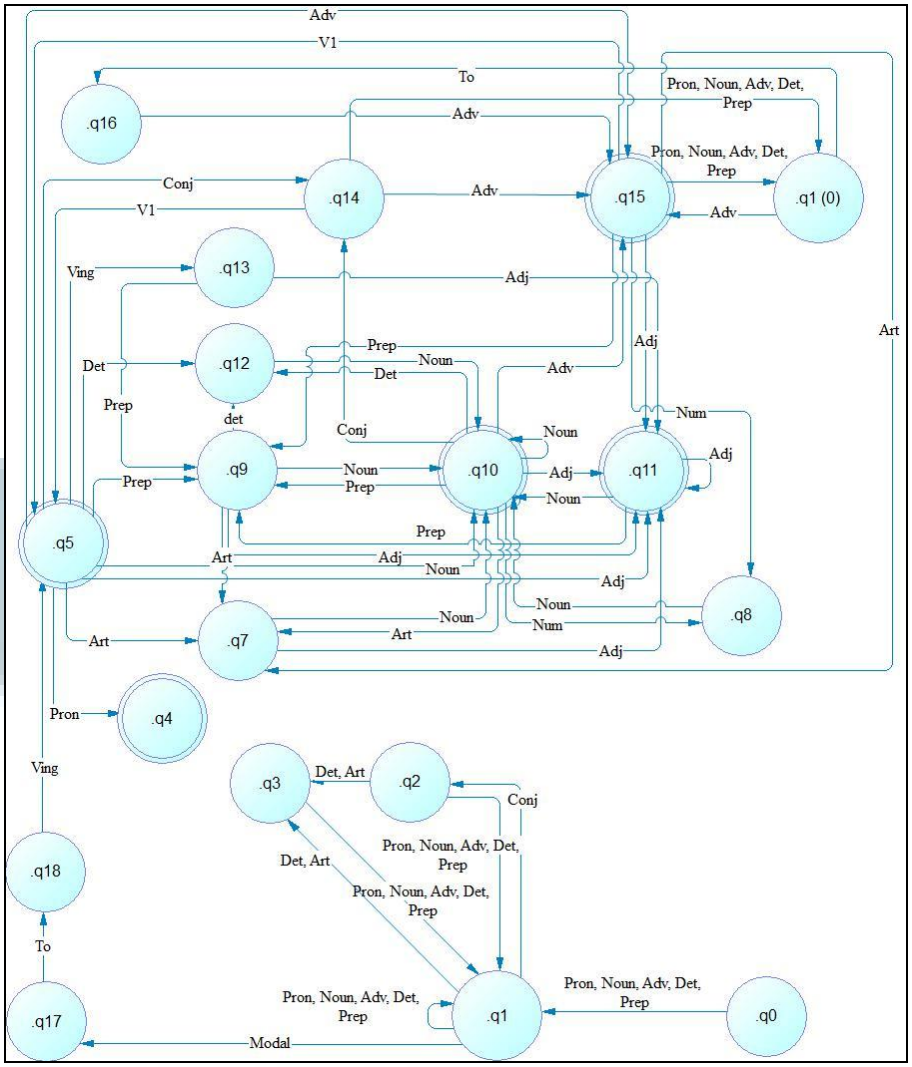


Gambar 3.23 Gambar NFA *Future Simple Tense* (kalimat positif dan negatif)

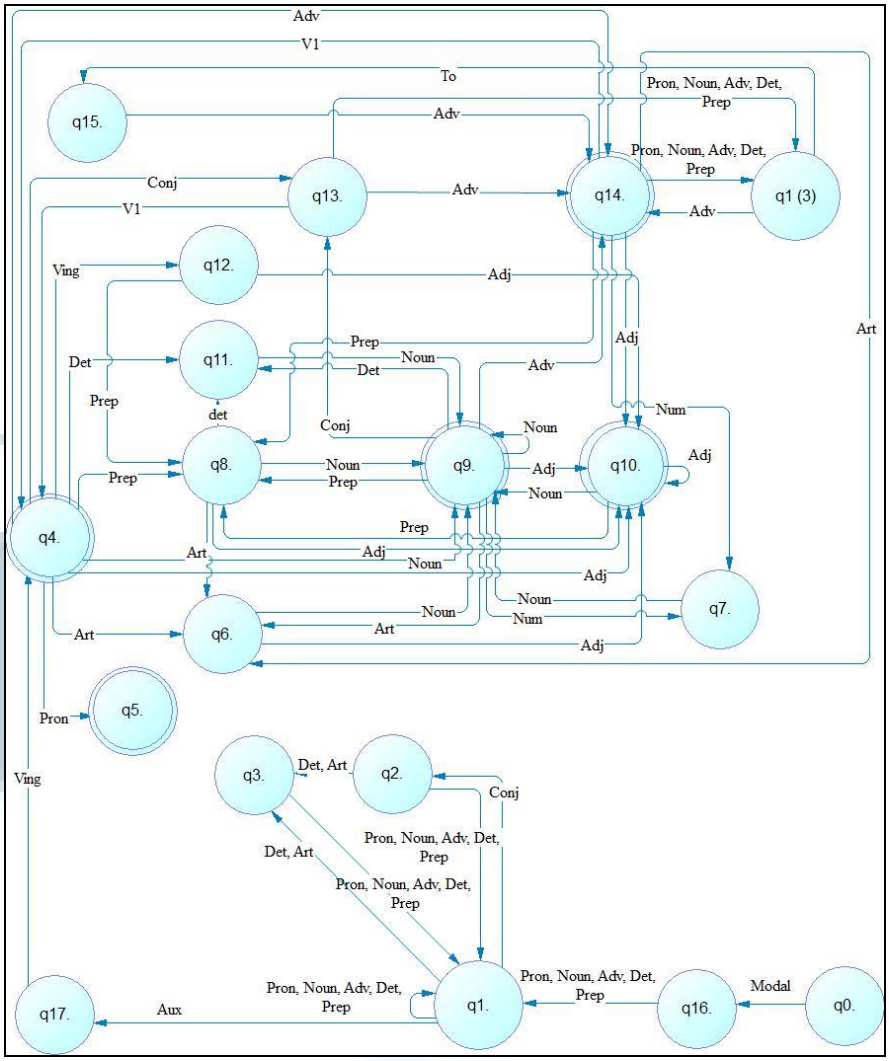


Gambar 3.24 Gambar NFA *Future Simple Tense* (kalimat tanya)

10. Future Continous Tense

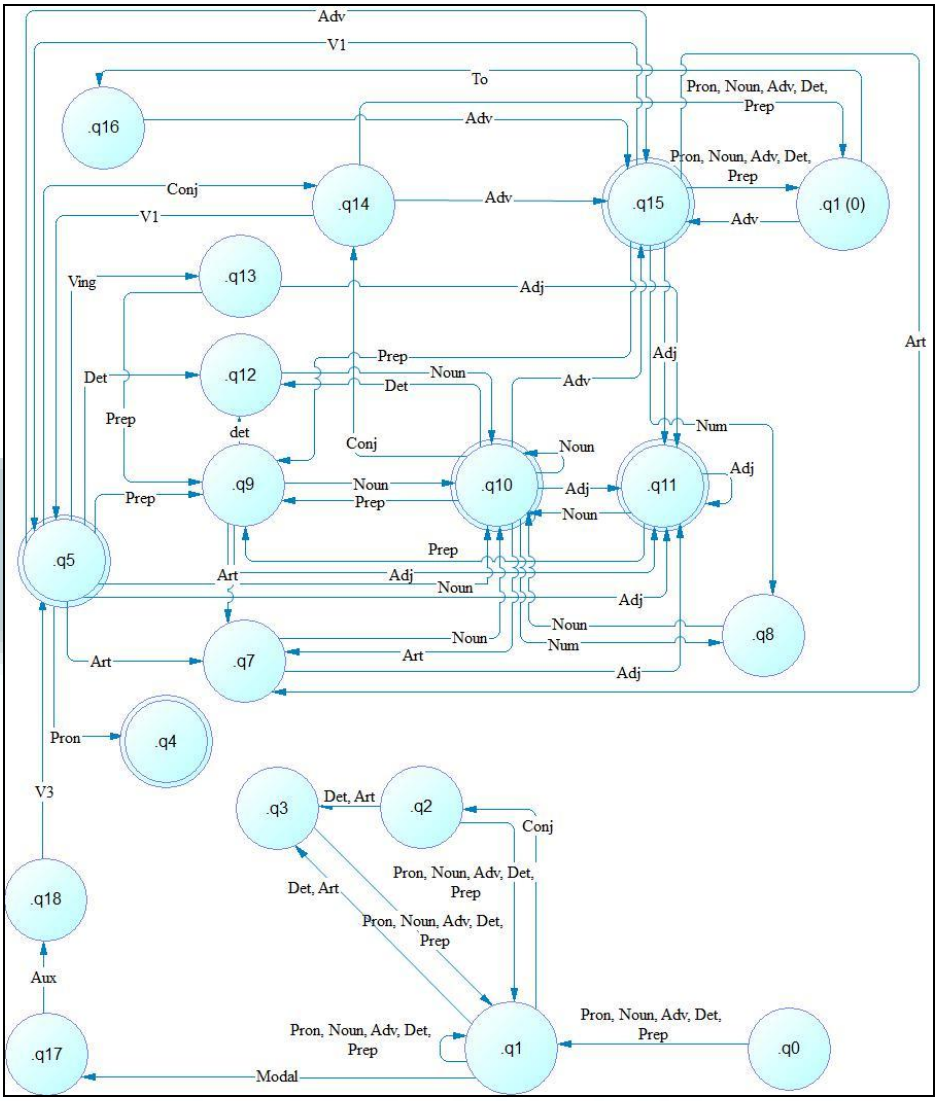


Gambar 3.25 Gambar NFA *Future Continous Tense* (kalimat positif dan negatif)

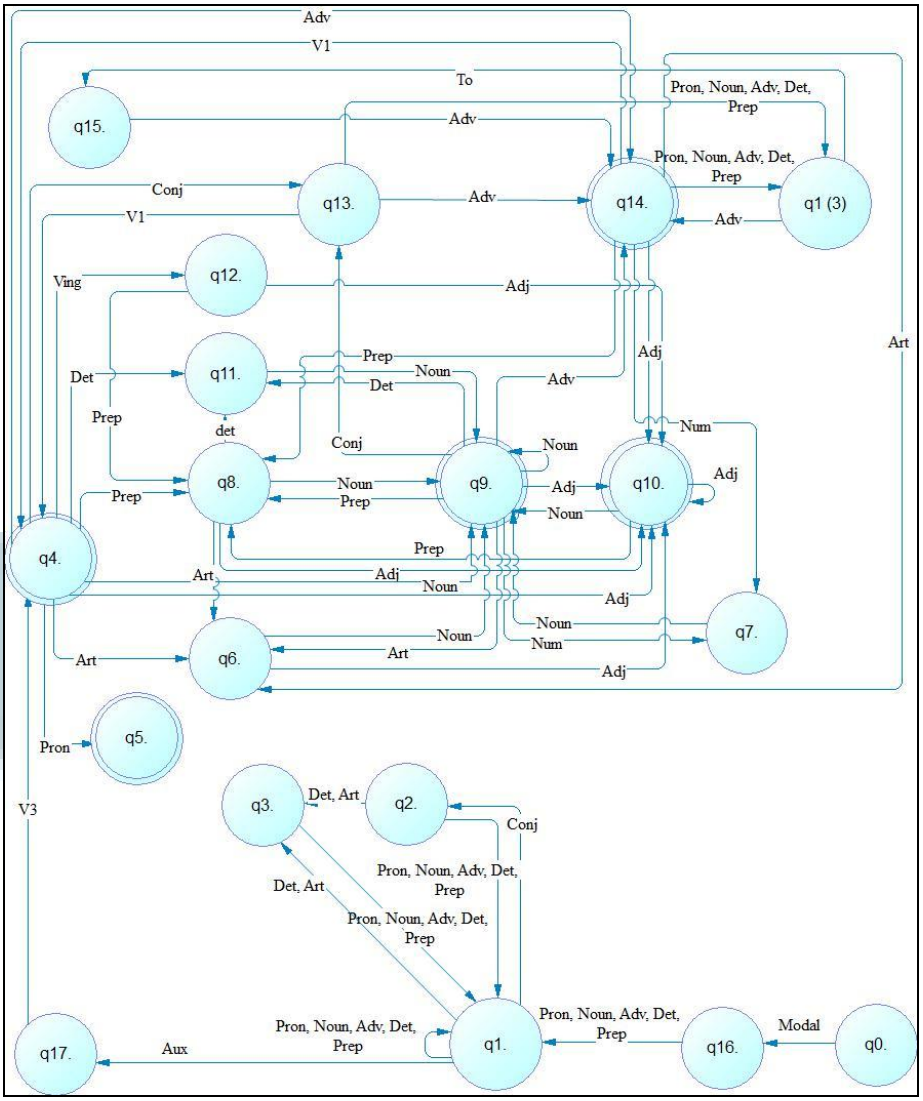


Gambar 3.26 Gambar NFA *Future Continous Tense* (kalimat tanya)

11. Future Perfect Tense

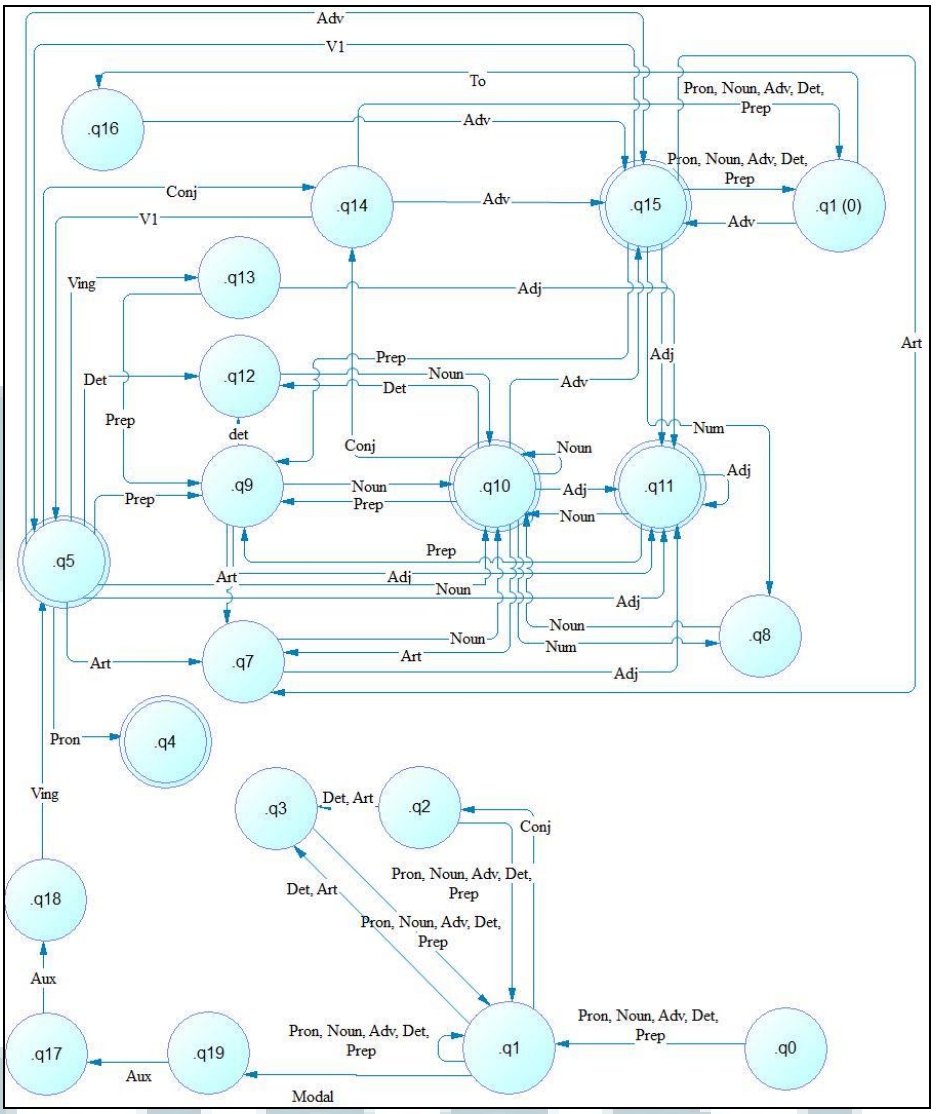


Gambar 3.27 Gambar NFA *Future Perfect Tense* (kalimat positif dan negatif)

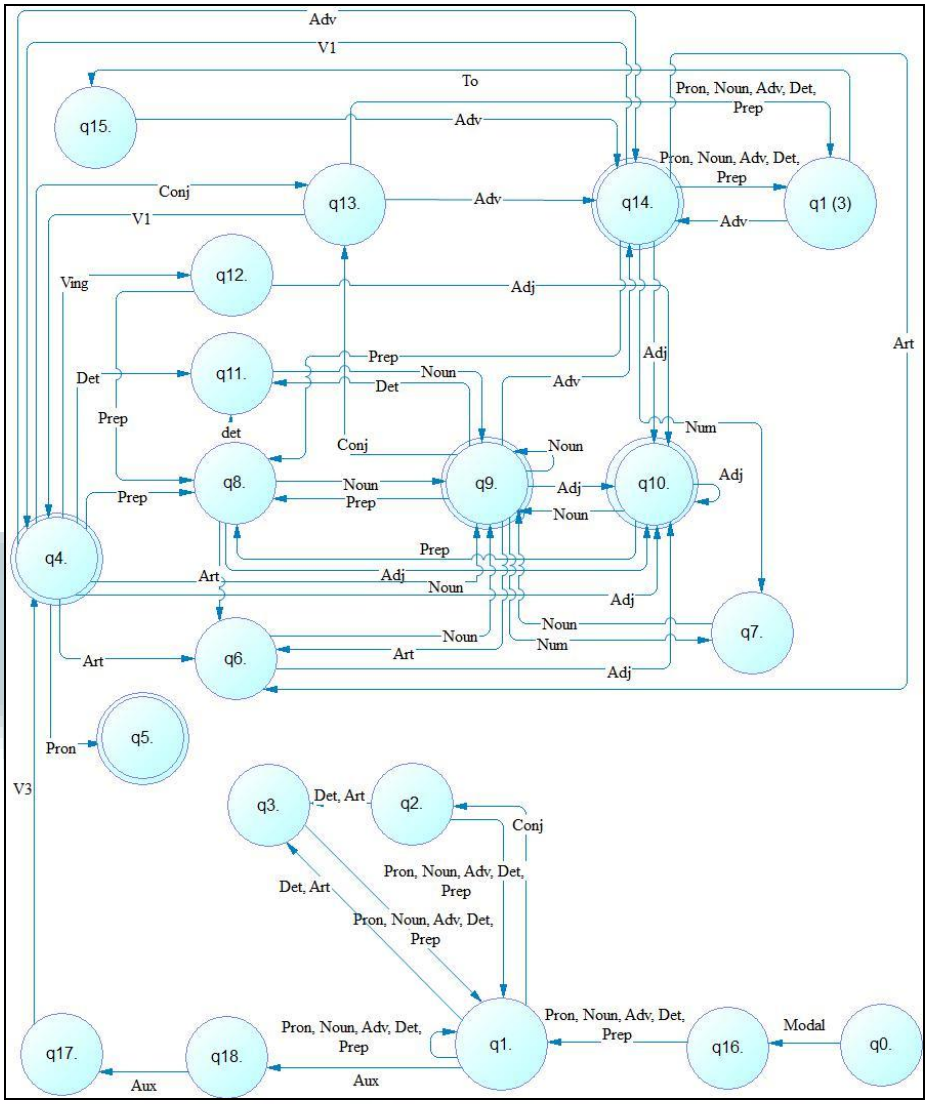


Gambar 3.28 Gambar NFA *Future Perfect Tense* (kalimat tanya)

12. Future Perfect Continous Tense

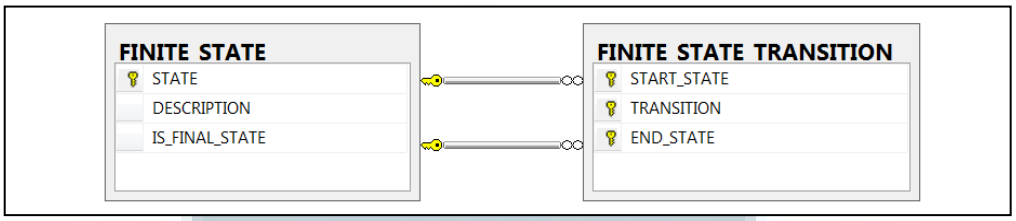


Gambar 3.29 Gambar NFA *Future Perfect Continous Tens*e (kalimat positif dan negatif)



Gambar 3.30 Gambar NFA *Future Perfect Continous Tense* (tanya)

3.2.2 Entity Relationship Diagram



Gambar 3.31 *Entity Relationship Diagram*

Relasi antar tabel pada aplikasi TranslatorId digambarkan dengan gambar di atas. Sebenarnya aplikasi ini memiliki banyak tabel yang berfungsi untuk menampung per jenis kata, namun dalam pengimplementasiannya, seluruh kata-kata tersebut dijadikan "view". Hal ini dilakukan dengan pertimbangan agar proses pembacaan data menjadi lebih cepat.

Relasi antar tabel di atas adalah hubungan antara perpindahan state dari pembacaan kata per kata dan relasi untuk menentukan apakah jenis kalimat (*tense*) pada kalimat berhasil ditemukan.

3.2.3 Struktur Tabel

Basis data pada aplikasi ini dibuat menggunakan SQL Server 2012. Basis data ini berisikan seluruh data yang digunakan dalam aplikasi TranslatorId yang sebagian besar dari data yang disimpan adalah berupa kata-kata dalam bahasa Indonesia dan bahasa Inggris. Terdapat tujuh belas tabel yang digunakan yang dijabarkan sebagai berikut.

1. Nama Tabel : ADJECTIVE

   Fungsi : Menampung data berupa kata-kata dengan jenis *adjective.*

Tabel 3.1 Tabel ADJECTIVE

Nama Kolom	**Tipe Data**	**Keterangan**
ID	int	*id, primary key*
ADJECTIVE	nvarchar(150)	daftar kata-kata dengan jenis *adjective* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *adjective* dalam bahasa Indonesia

2. Nama Tabel : ADVERB

   Fungsi : Menampung data berupa kata-kata dengan jenis *adverb.*

Tabel 3.2 Tabel ADVERB

Nama Kolom	**Tipe Data**	**Keterangan**
ID	int	*id, primary key*
ADVERB	nvarchar(150)	daftar kata-kata dengan jenis *adverb* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *adverb* dalam bahasa Indonesia

3. Nama Tabel : ARTICLE

   Fungsi : Menampung data berupa kata-kata dengan jenis *article.*

Tabel 3.3 Tabel ARTICLE

Nama Kolom	**Tipe Data**	**Keterangan**
ID	int	*id, primary key*
ARTICLE	nvarchar(150)	daftar kata-kata dengan jenis *article* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *article* dalam bahasa Indonesia

4. Nama Tabel : AUXILIARY

Fungsi : Menampung data berupa kata-kata dengan jenis *auxiliary verb.*

Tabel 3.4 Tabel AUXILIARY

Nama Kolom	Tipe Data	Keterangan
ID	int	*id, primary key*
AUXILIARY	nvarchar(150)	daftar kata-kata dengan jenis *auxiliary* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *auxiliary* dalam bahasa Indonesia

5. Nama Tabel : AUXILIARY_PAST

Fungsi : Menampung data berupa kata-kata dengan jenis *auxiliary* (bentuk lampau).

Tabel 3.5 Tabel AUXILIARY_PAST

Nama Kolom	Tipe Data	Keterangan
ID	int	*id, primary key*
AUXILIARY	nvarchar(150)	daftar kata-kata dengan jenis *auxiliary* (bentuk lampau) dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *auxiliary* (bentuk lampau) dalam bahasa Indonesia

6. Nama Tabel : CONJUNCTION

Fungsi : Menampung data berupa kata-kata dengan jenis *conjuction.*

Tabel 3.6 Tabel CONJUNCTION

Nama Kolom	Tipe Data	Keterangan
ID	int	*id, primary key*
CONJUNCTION	nvarchar(150)	daftar kata-kata dengan jenis *conjunction* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *conjunction* dalam bahasa Indonesia

7. Nama Tabel : DETERMINER

   Fungsi : Menampung data berupa kata-kata dengan jenis *determiner*.

Tabel 3.7 Tabel DETERMINER

Nama Kolom	Tipe Data	Keterangan
ID	int	*id, primary key*
DETERMINER	nvarchar(150)	daftar kata-kata dengan jenis *determiner* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *determiner* dalam bahasa Indonesia

8. Nama Tabel : MODAL

   Fungsi : Menampung data berupa kata-kata dengan jenis *modal verb.*

Tabel 3.8 Tabel MODAL

Nama Kolom	Tipe Data	Keterangan
ID	int	*id, primary key*
MODAL	nvarchar(150)	daftar kata-kata dengan jenis *modal* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *modal* dalam bahasa Indonesia

9. Nama Tabel : NOUN

   Fungsi : Menampung data berupa kata-kata dengan jenis *noun.*

Tabel 3.9 Tabel NOUN

Nama Kolom	Tipe Data	Keterangan
ID	int	*id, primary key*
NOUN	nvarchar(150)	daftar kata-kata dengan jenis *noun* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *noun* dalam bahasa Indonesia

10. Nama Tabel : PREPOSITION

Fungsi : Menampung data berupa kata-kata dengan jenis *preposition.*

Tabel 3.10 Tabel PREPOSITION

Nama Kolom	**Tipe Data**	**Keterangan**
ID	int	*id, primary key*
PREPOSITION	nvarchar(150)	daftar kata-kata dengan jenis *preposition* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *preposition* dalam bahasa Indonesia

11. Nama Tabel : PRONOUN

Fungsi : Menampung data berupa kata-kata dengan jenis *pronoun.*

Tabel 3.11 Tabel PRONOUN

Nama Kolom	**Tipe Data**	**Keterangan**
ID	int	*id, primary key*
PRONOUN	nvarchar(150)	daftar kata-kata dengan jenis *pronoun* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *pronoun* dalam bahasa Indonesia

12. Nama Tabel : TOBE

Fungsi : Menampung data berupa kata-kata dengan jenis *tobe.*

Tabel 3.12 Tabel TOBE

Nama Kolom	**Tipe Data**	**Keterangan**
ID	int	*id, primary key*
TO_BE	nvarchar(150)	daftar kata-kata dengan jenis *to be* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *to be* dalam bahasa Indonesia

13. Nama Tabel : TOBE_PAST

Fungsi : Menampung data berupa kata-kata dengan jenis *tobe* (bentuk lampau).

Tabel 3.13 Tabel TOBE_PAST

Nama Kolom	Tipe Data	Keterangan
ID	int	*id, primary key*
TO_BE	nvarchar(150)	daftar kata-kata dengan jenis *to be* (bentuk lampau) dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *to be* (bentuk lampau) dalam bahasa Indonesia

14. Nama Tabel : VERB

Fungsi : Menampung data berupa kata-kata dengan jenis *verb.*

Tabel 3.14 Tabel VERB

Nama Kolom	Tipe Data	Keterangan
ID	int	*id, primary key*
VERB	nvarchar(150)	daftar kata-kata dengan jenis *verb* dalam bahasa Inggris
MEAN	nvarchar(50)	daftar kata-kata dengan jenis *verb* dalam bahasa Indonesia

15. Nama Tabel : WORDS

Fungsi : Menampung data berupa referensi kata-kata yang diterjemahkan.

Tabel 3.15 Tabel WORDS

Nama Kolom	Tipe Data	Keterangan
ID	int	*id, primary key*
WORD	nvarchar(100)	kata-kata yang disimpan
TYPE	nvarchar(20)	tipe dari kata-kata
ORIGIN_ TABLE	nvarchar(50)	tabel asal kata-kata
REF_ID	int	id yang direferensikan

16. Nama Tabel : FINITE_STATE

Fungsi : Menampung data berupa jenis kalimat bahasa Inggris (*tenses)* yang diperoleh dari state terakhir.

Tabel 3.16 Tabel FINITE_STATE

Nama Kolom	**Tipe Data**	**Keterangan**
STATE	nvarchar(10)	*state automata, primary key*
DESCRIPTION	nvarchar(100)	jenis *tenses* yang diperoleh
IS_FINAL_ STATE	nvarchar(1)	pengecekan final state

17. Nama Tabel : FINITE_STATE_TRANSITION

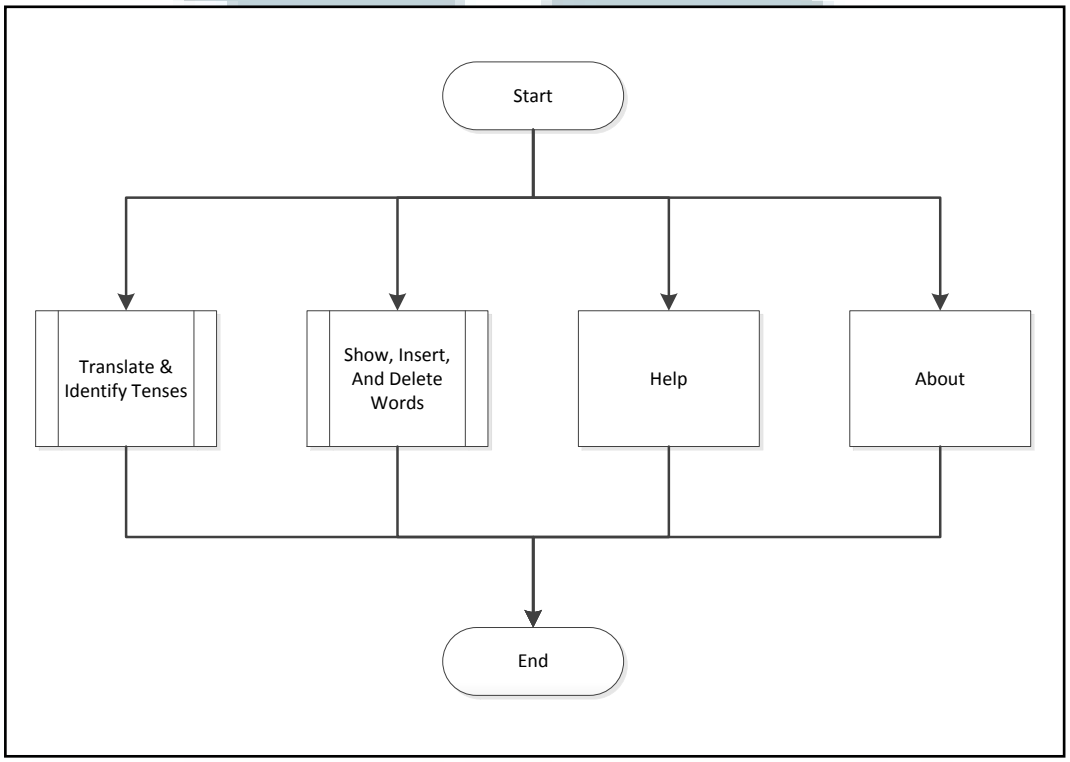
Fungsi : Menampung data berupa daftar transisi *nondeterministic finite automata*.

Tabel 3.17 Tabel FINITE_STATE_TRANSITION

Nama Kolom	**Tipe Data**	**Keterangan**
START_STATE	nvarchar(10)	menyimpan *state* awal, *primary key*
TRANSITION	nvarchar(20)	jenis transisi yang dilalui
END_STATE	nvarchar(10)	menyimpan *state* yang dituju, primary key

3.2.4 Flowchart Sistem

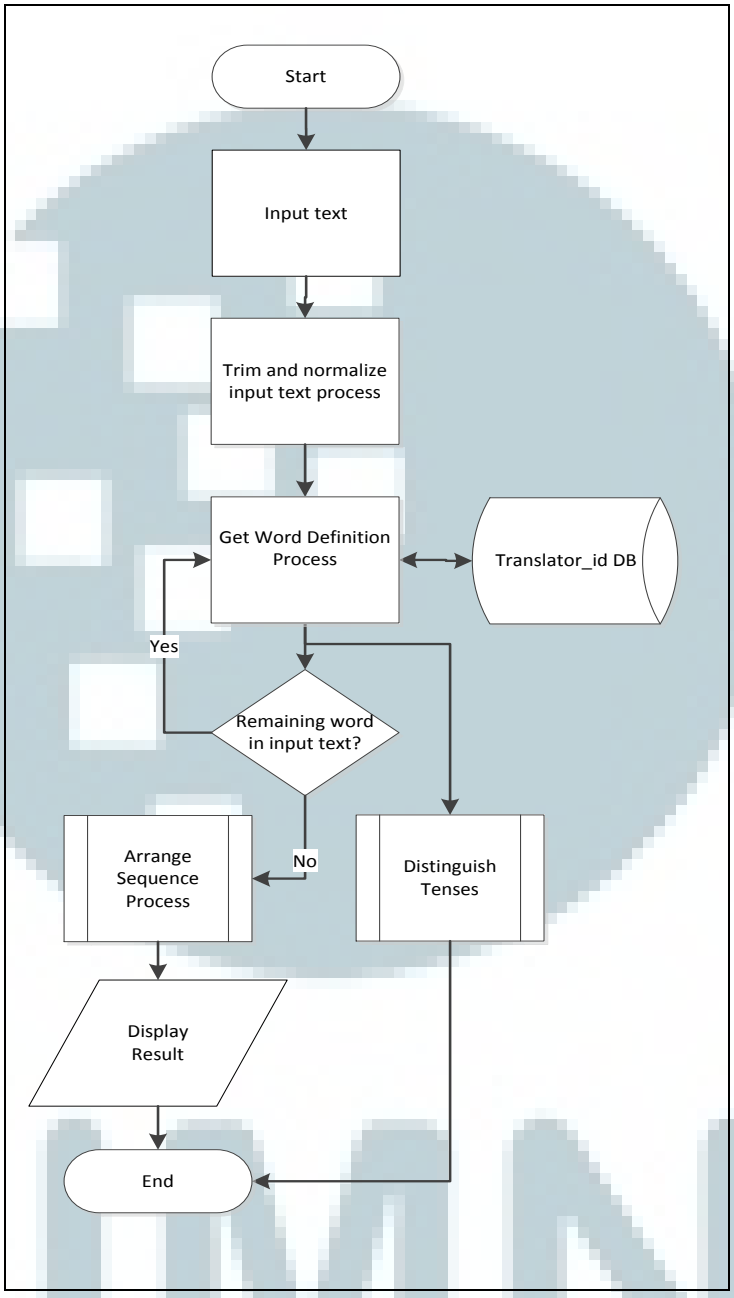
Bagian ini berisi bagan dengan jalur proses yang digunakan untuk menggambarkan langkah-langkah yang dilalui oleh sistem dalam menyelesaikan suatu masalah.



Gambar 3.32 *Flowchart* Sistem TranslatorId

Aplikasi TranslatorId ini memiliki 4 proses utama seperti yang ditunjukkan oleh gambar di atas. Proses tersebut mencakup seluruh aktifitas yang dilakukan oleh aplikasi ini.

1. *Flowchart* Sistem (Subproses *Translate & Identify Tenses*)



Gambar 3.33 Gambar Subproses *Translate & Identify Tenses*

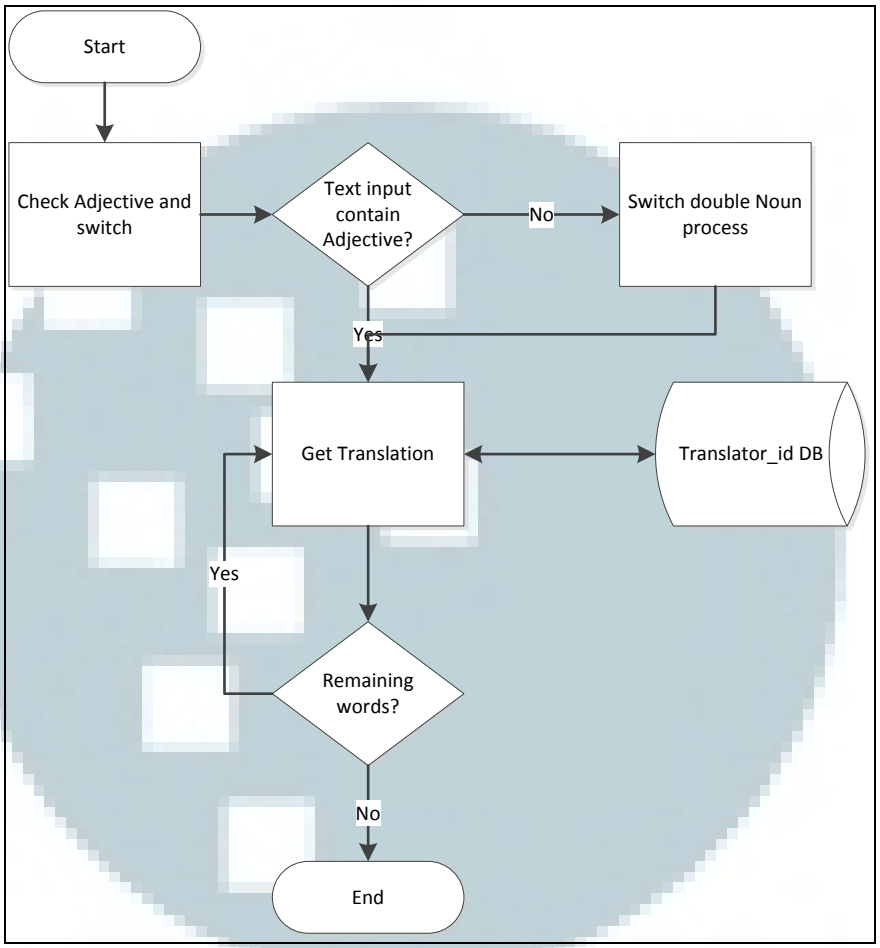
Aplikasi TranslatorId digunakan sebagai aplikasi yang mampu menerjemahkan sebuah kalimat dan mengidentifikasi jenis kalimat tersebut. Oleh karenanya, langkah awal penerjemahan dilakukan dengan pemotongan kalimat

yang dimasukkan dan penormalisasian kalimat tersebut agar menjadi potongan-potongan kata yang akan diperiksa artinya satu persatu. Potongan kata-kata ini akan diperiksa menggunakan data yang tersimpan di *database* TranslatorId.

Setelah semua kata-kata hasil pemotongan tersebut sudah habis diterjemahkan, maka langkah selanjutnya terbagi dalam dua subproses yakni subproses penyusunan kata-kata tersebut menjadi sebuah kalimat dalam bahasa Indonesia dan subproses untuk mengidentifikasi *state* guna memperoleh jenis kalimat (*tenses*) dari kalimat yang diterjemahkan tersebut.

Pada aplikasi TranslatorId, hasil terjemahan kalimat akan ditampilkan sebagai hasil terjemahan dan hasil dari identifikasi akan ditampilkan sebagai keterangan.

2. *Flowchart* Sistem (Subproses *Arrange Sequence Process*)

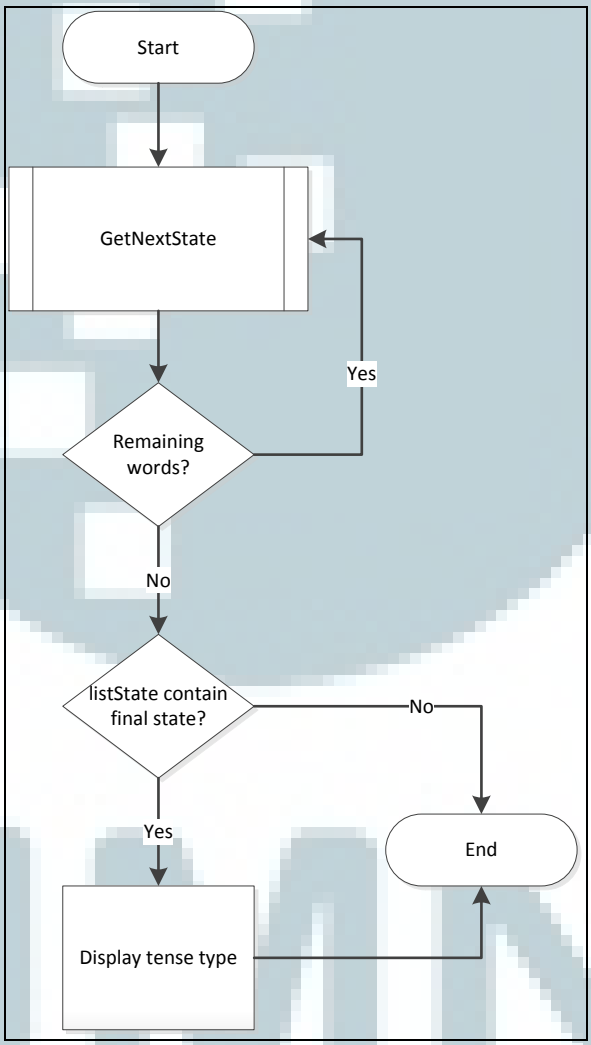


Gambar 3.34 Gambar Subproses *Arrange Sequence Process*

Pada Subproses *Arrange Sequence Process*, kata-kata yang memiliki jenis kata *adjective* dan *noun* dipisahkan untuk dilakukan pertukaran posisi karena pola bahasa Indonesia dan bahasa Inggris memiliki perbedaan, terutama pada jenis kata benda dan kata sifat yang dibolak-balik. Langkah awal pada subproses ini diawali dengan memeriksa adanya kata sifat (*adjective*) dan kata benda (noun). Apabila ada keduanya, posisi kata sifat dan kata benda akan ditukar di mana kata benda didahulukan di depan kata sifat. Apabila ada dua buah atau lebih kata benda, proses akan dilanjutkan dengan pembalikan kata-kata benda tersebut, namun

apabila tidak ada kata benda, proses selanjutnya adalah penerjemahan kata-kata tersebut menggunakan data yang ada di *database* TranslatorId. Subproses *Arrange Sequence Process* akan diakhiri apabila tidak ada *input* lebih lanjut.

3. *Flowchart* Sistem (Subproses *Distinguish Tenses*)



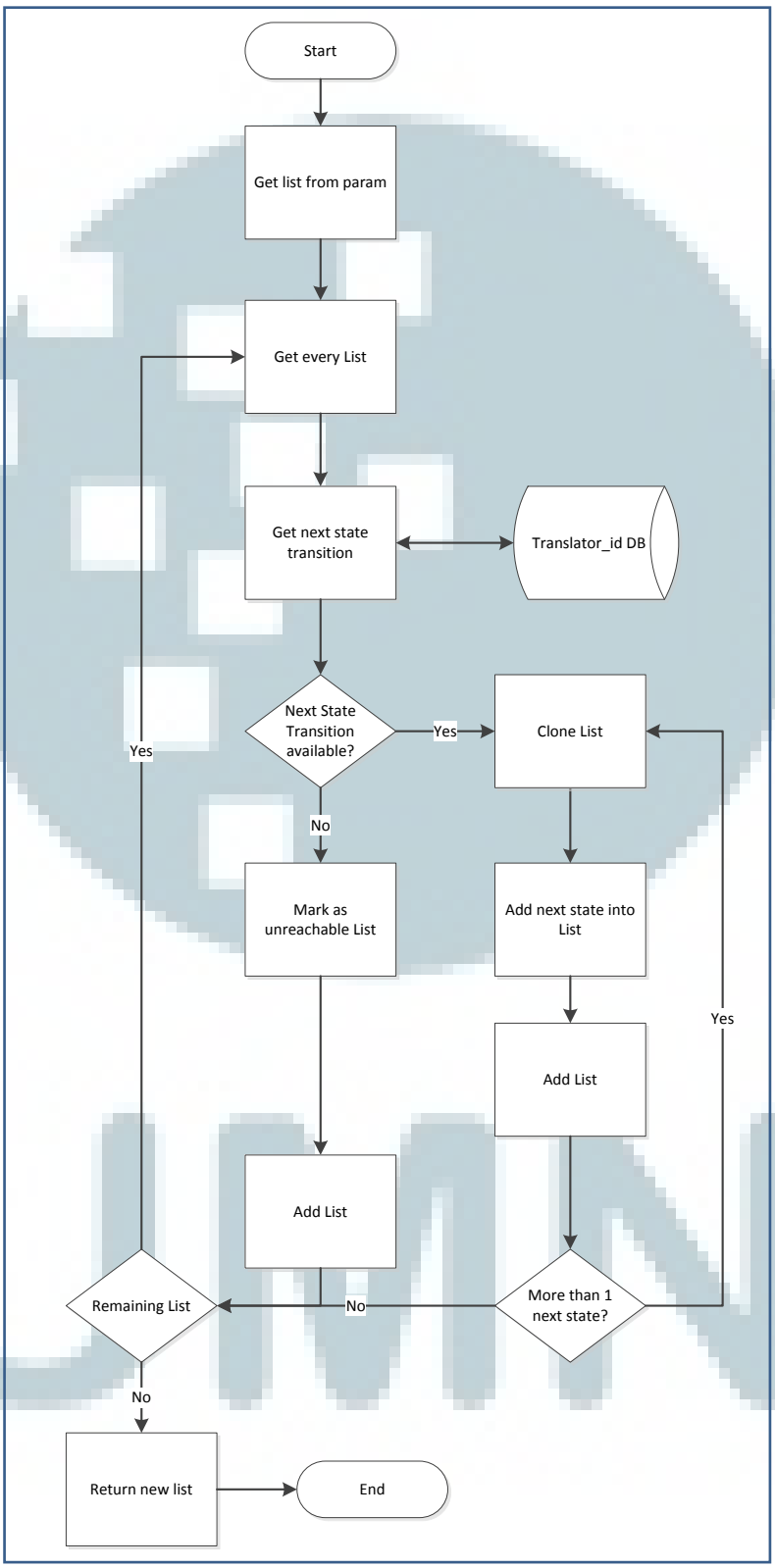
Gambar 3.35 Gambar *Subproses Distinguish Tenses*

Pada Subproses *Distinguish Tenses* dilakukan penelusuran *state* dari kalimat yang diterjemahkan. Pada proses awal penerjemahan, aplikasi telah

menyimpan urutan sifat kata dari hasil pemotongan kata. Urutan sifat kata tersebut akan digunakan untuk mengetahui jenis dari kalimat bahasa Inggris (*tenses*). Pada subproses ini, konsep *Nondeterministic Finite Automata* diterapkan.

Langkah awal dari subproses ini adalah membaca urutan kata yang diterjemahkan. Berdasarkan urutan dari kata-kata yang dipotong, akan dilakukan penelusuran *state* hingga kata terakhir pada kalimat tersebut. Apabila sudah tidak ada lagi kata-kata yang bisa ditelusuri, maka proses penelusuran kata diakhiri dan akan diperiksa apakah *state* yang terkandung dalam kata-kata tersebut adalah *final state*. Apabila *state* terakhir adalah *final state*, maka akan ditampilkan jenis kalimat (t*enses*) yang berhasil diidentifikasi pada kolom keterangan menu penerjemah (*translator*) aplikasi TranslatorId. Untuk proses lengkap mengenai cara pembacaan *state*, akan dijelaskan pada subproses *GetNextState.*

4. *Flowchart* Sistem (Subproses *GetNextState)*



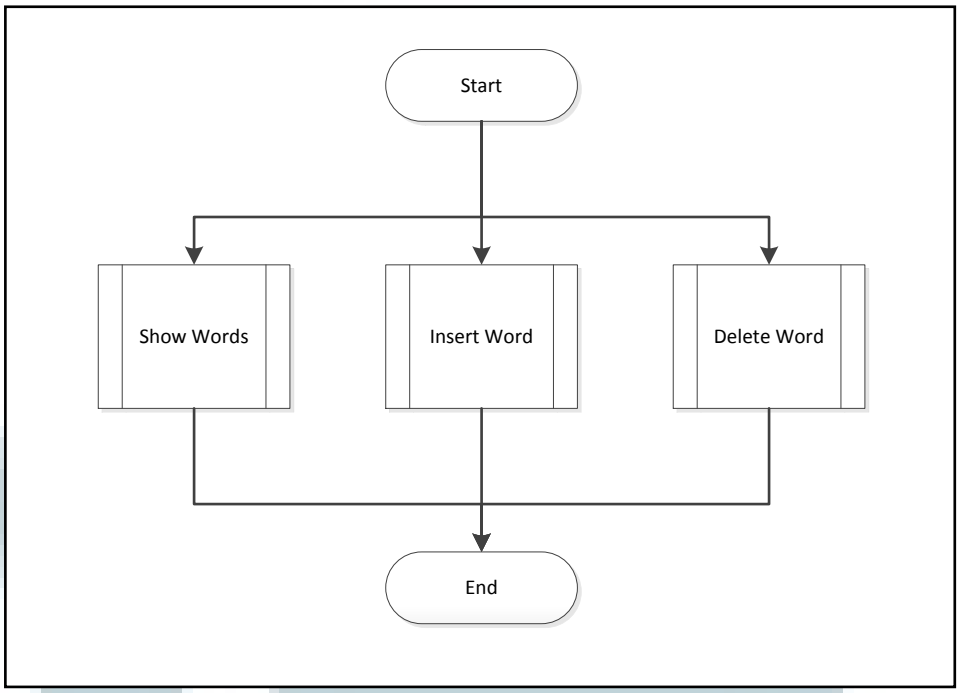
Gambar 3.36 *Flowchart* Sistem Penelusuran *State*

Pada subproses *GetNextState* dilakukan penelusuran untuk membaca urutan *state* yang diawali dengan mengambil urutan dari jenis kata yang sebelumnya sudah dipotong sebagai parameter. Potongan kata tersebut akan dicatat urutan *state* (berdasarkan jenis kata) yang akan dilaluinya. Transisi dari *state* diperiksa ke dalam *database* TranslatorId.

Apabila ada *state* selanjutnya maka dilakukan penggandaan terhadap daftar transisi. Penduplikasian ini akan digunakan untuk mencatat jalur transisi apabila ada lebih dari satu jalan untuk mencapai *state* berikutnya. Pada tahapan ini dilakukan pencatatan *state* kata yang dibaca terakhir dan *state* dari kata berikutnya, selain dilakukan pencatatan kata dan *state* dari kata, dilakukan pula pencatatan terhadap transisi *state* yang dilaluinya.

Apabila tidak ada *state* selanjutnya (tidak terdaftar), maka akan dilakukan pencatatan terhadap *state* dari pembacaan terakhir. Pada akhir proses, akan dikembalikan posisi *state* terakhir yang diperoleh dari hasil pembacaan potongan-potongan kata.

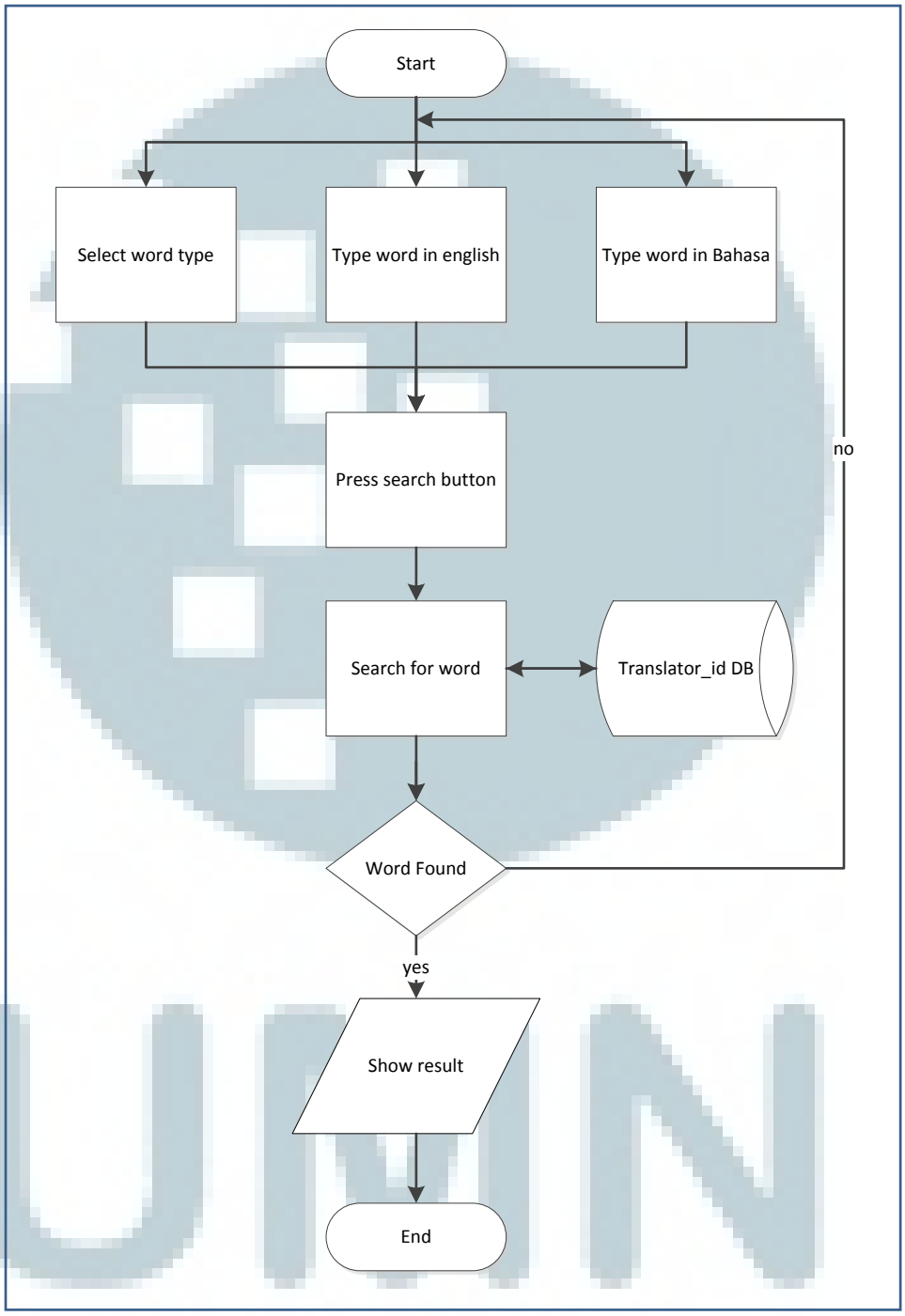
5. *Flowchart* Sistem (Subproses *Show, Insert, And Delete Words*)



Gambar 3.37 Gambar Subproses *Show, Insert, And Delete Words*

Pada Subproses *Show, Insert, And Delete Words* terdapat tiga buah subproses lain, yakni *Subproses Show Words* yang menggambarkan proses pencarian kata oleh pengguna, *Subproses Insert Words* yang menggambarkan proses penambahan kata oleh pengguna, dan *Subproses Delete Word* yang menggambarkan proses penghapusan kata oleh pengguna aplikasi.

6. *Flowchart* Sistem (Subproses *Show Words*)

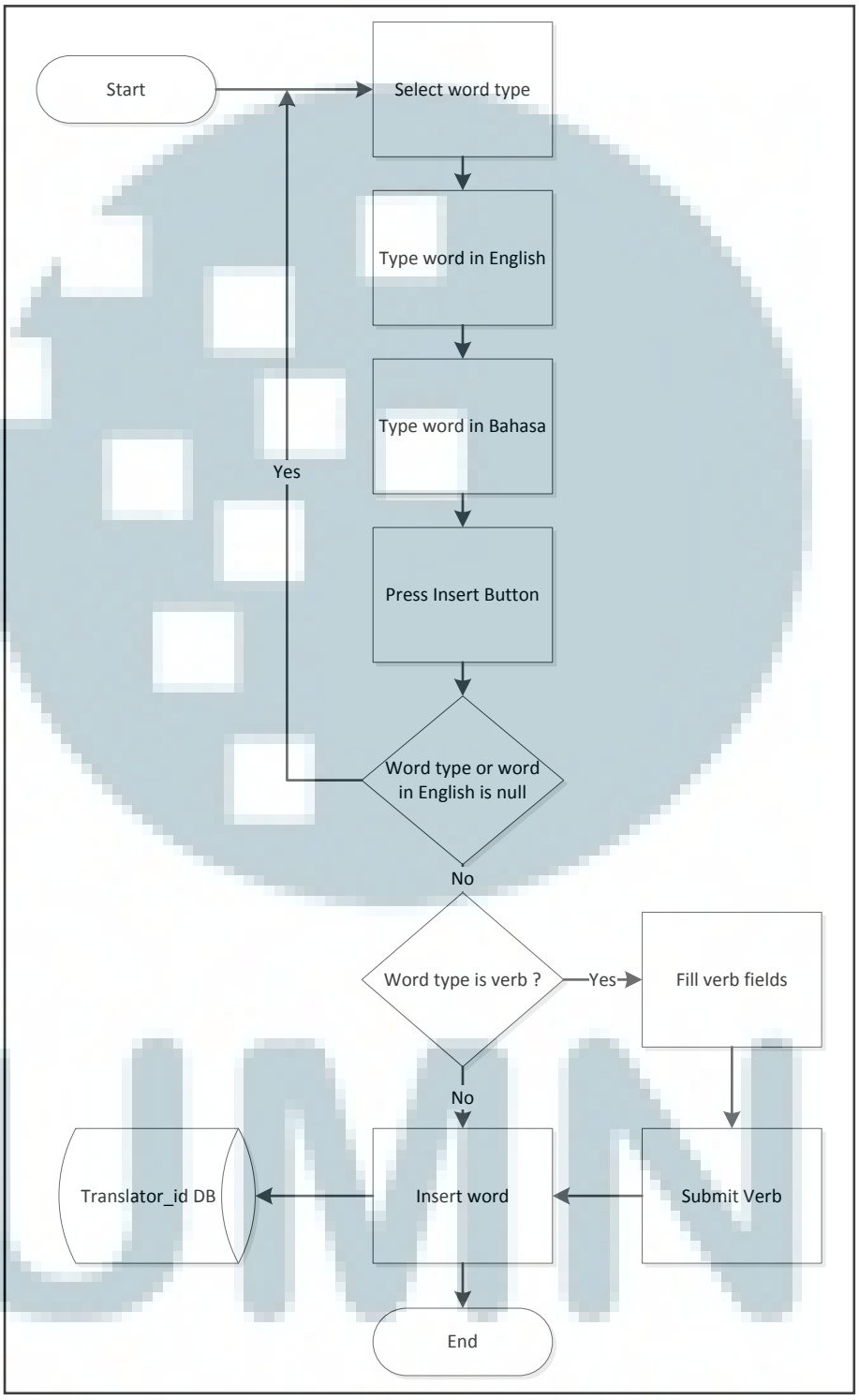


Gambar 3.38 Gambar Subproses *Show Words*

Subproses *Show Words* yang menggambarkan proses pencarian kata diawali dengan memasukan salah satu dari tiga parameter pencarian. Pengguna dapat mencari berdasarkan jenis kata, kata dalam bahasa Inggris, atau kata dalam bahasa Indonesia. Untuk pencarian yang lebih akurat, pengguna juga dapat memasukkan tiga parameter tersebut sekaligus.

Setelah dilakukan klik pada tombol pencarian, maka kata tersebut akan dicari ke dalam basis data sesuai dengan parameter yang telah dimasukkan sebelumnya. Apabila ditemukan, maka kata-kata yang memiliki kemiripan dengan kata yang dimasukkan ke dalam parameter akan ditampilkan, namun apabila tidak ditemukan kata yang dicari, maka tidak akan ditampilkan apapun di layar dan pengguna aplikasi dapat melakukan pencarian ulang.

7. *Flowchart* Sistem (Subproses *Insert Word*)

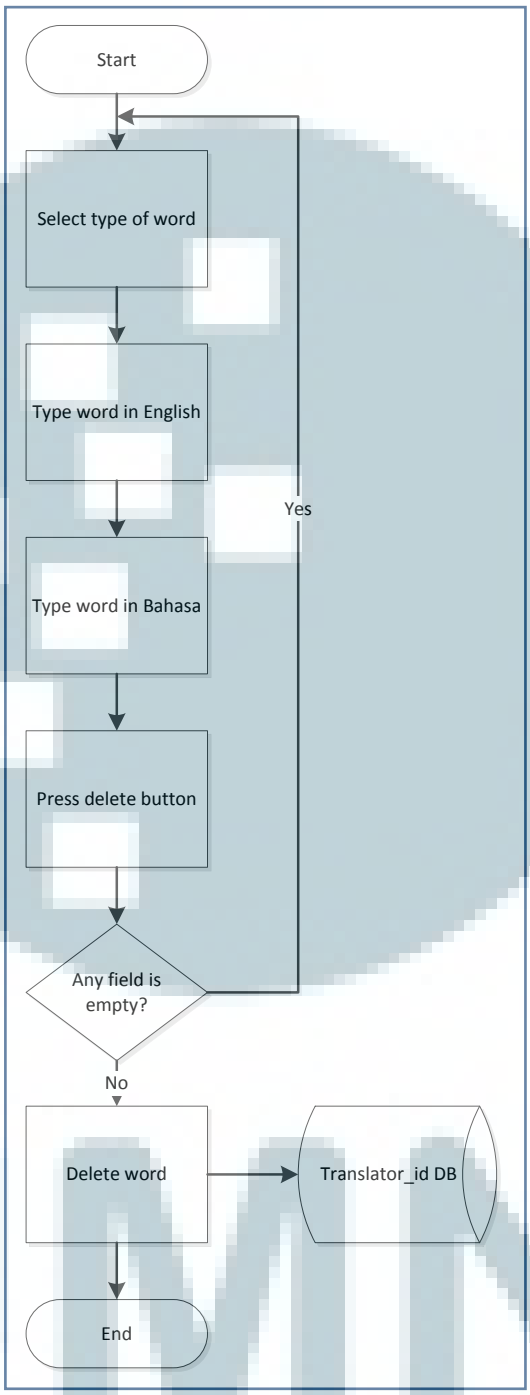


Gambar 3.39 Gambar Subproses *Insert Word*

Subproses *Insert Words* yang menggambarkan proses menambahkan kata ke dalam basis data. Proses ini diawali dengan memasukkan parameter jenis kata, kata dalam bahasa Inggris dan kata dalam bahasa Indonesia. Parameter jenis kata dan kata dalam bahasa Inggris harus dimasukkan sebagai *primary key* di dalam tabel, namun kata dalam bahasa Indonesia dapat dikosongkan karena ada kata tertentu dalam bahasa Inggris yang tidak memiliki arti dalam bahasa Indonesia.

Setelah parameter jenis kata dan kata dalam bahasa Inggris sudah dimasukkan, maka akan dilihat apakah jenis kata adalah *verb* atau bukan. Kata dengan jenis *verb* dibedakan karena jenis kata *verb* memiliki tiga bentuk (kini, lampau, dan bentuk sudah), serta memiliki dua arti yakni aktif dan pasif. Jenis kata selain *verb* hanya memiliki bentuk kata dalam bahasa Inggris dan artinya dalam bahasa Indonesia. Setelah seluruh parameter telah dimasukkan, maka kata akan disimpan ke dalam basis data.

8. *Flowchart* Sistem (Subproses *Delete Word*)



Gambar 3.40 Gambar Subproses *Delete Word*

Subproses *Delete Word* yang menggambarkan proses penghapusan kata dari basis data. Proses ini diawali dengan memasukkan parameter jenis kata, kata dalam bahasa Inggris dan kata dalam bahasa Indonesia. Ketiga parameter harus dimasukkan dengan pertimbangan agar aplikasi dalam melakukan penghapusan kata sesuai dengan apa yang ingin dihapus oleh pengguna. Apabila seluruh parameter telah terisi dengan benar, maka kata akan dihapuskan dari basis data.

3.2.5 Potongan Source Code Program

Pembahasan mengenai alur proses (*flowchart*) sistem merupakan sebuah komponen penting dalam memberikan gambaran mengenai sebuah aplikasi. Selain *flowchart*, adapula komponen lain yang dapat memberikan gambaran tentang bagaimana aplikasi ini berproses, yakni dengan membaca *potongan source code program*. Berikut adalah potongan *source code* sesuai dengan urutan yang pembahasan proses kerjanyanya telah dijelaskan pada *flowchart* aplikasi TranslatorId.

1. *Translate Sentences*

```
private void btn_terjemahkan_Click(object sender, EventArgs e)
{
    try
    {
        listStateTransition.Clear();
        acceptedList.Clear();
        dictionary.Clear();
        dictionaryArranged.Clear();
        isContainAdj = false;

        this.rawText = tbox_inggris.Text.Replace("\r\n", " ").Replace("\n", " ").Replace("\r", " ");
        String[] rawWords = rawText.Trim().Split(' ');

        if (rawWords.Length == 1 && tbox_inggris.Text.Length > 0)
        {
            if (Regex.IsMatch(tbox_inggris.Text, @"^\d+$"))
            {
                DataTable dt = new DataTable();
                dt.Columns.Add("id", typeof(int));
                dt.Columns.Add("word", typeof(string));
                dt.Columns.Add("type", typeof(string));
                dt.Columns.Add("origin_table", typeof(string));
                dt.Columns.Add("ref_id", typeof(int));


            int a = 0;
            Int32.TryParse(tbox_inggris.Text, out a);

            dt.Rows.Add(0, tbox_inggris.Text, "NUMBER", "NUMBER", a);
            dictionary.Add(tbox_inggris.Text, dt);
        }
        else
        {
            dictionary = words.GetWord(dictionary, Regex.Replace(rawWords[0], "[?!.]", String.Empty));
        }
    }
    else
    {
        dictionary = new Dictionary<string, DataTable>();
        foreach (String raw in rawWords)
        {
            if (Regex.IsMatch(raw, @"^\d+$"))
            {
                DataTable dt = new DataTable();
                dt.Columns.Add("id", typeof(int));
                dt.Columns.Add("word", typeof(string));
                dt.Columns.Add("type", typeof(string));
                dt.Columns.Add("origin_table", typeof(string));
                dt.Columns.Add("ref_id", typeof(int));

                int a = 0;
                Int32.TryParse(raw, out a);


                   dt.Rows.Add(0, raw, "NUMBER", "NUMBER", a);
                   dictionary.Add(raw, dt);
               }
               else
               {
                   dictionary = words.GetWord(dictionary, Regex.Replace(raw, "[?!.]", String.Empty));
               }
           }
       }

       this.OrganinizeDictionary(dictionary);
       this.DisplayResult(ArrangeSequence(dictionary));
       this.DisplayDescription(dictionary);
   }
```

Gambar 3.41 Potongan *source code Translate Sentences*

```
public String GetTranslation(String table_name, int id, Boolean? isActive)
{
    DataTable dt = null;
    try
    {
        if (table_name.Equals("verb", StringComparison.InvariantCultureIgnoreCase))
        {
            if (isActive.HasValue)
                return this.GetVerbTranslation(id, isActive.Value);
            else
                return this.GetVerbTranslation(id, true);
        }

        SqlCommand cmd = new SqlCommand("sp_translate", Utility.GetConnection());
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.Add("@p_table", SqlDbType.NVarChar).Value = table_name;
        cmd.Parameters.Add("@p_id", SqlDbType.Int).Value = id;
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        dt = new DataTable();
        da.Fill(dt);

        return dt.Rows[0]["MEAN"].ToString();
    }

private String GetVerbTranslation(int id, Boolean isActive)
{
    DataTable dt = null;
    try
    {
        SqlCommand cmd = new SqlCommand("sp_translate_verb", Utility.GetConnection());
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.Add("@p_id", SqlDbType.Int).Value = id;
        if (isActive)
        {
            cmd.Parameters.Add("@p_is_active", SqlDbType.NVarChar).Value = "ACTIVE";
        }
        else
        {
            cmd.Parameters.Add("@p_is_active", SqlDbType.NVarChar).Value = "PASSIVE";
        }
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        dt = new DataTable();
        da.Fill(dt);

        return dt.Rows[0]["MEAN"].ToString();
    }
```

Gambar 3.42 Potongan *source code Translate Sentences* (*cont'd*)

2. Arrange Sequence Process

```
private Dictionary<string, DataTable> ArrangeSequence(Dictionary<string, DataTable> dict)
{
    try
    {
        List<KeyValuePair<string, DataTable>> tempList = new List<KeyValuePair<string, DataTable>>();
        List<KeyValuePair<string, DataTable>> resultList = new List<KeyValuePair<string, DataTable>>();
        List<KeyValuePair<string, DataTable>> resultList2 = new List<KeyValuePair<string, DataTable>>();

        foreach (KeyValuePair<string, DataTable> key in dict)
        {
            if (key.Value.Rows.Count > 0)
            {
                tempList.Add(key);
            }
        }

        resultList = CheckNoun(tempList);

        if (!isContainAdj)
        {
            resultList = SwitchDoubleNoun(tempList);
        }

        for (int i = 0; i < resultList2.Count(); i++)
        {
            dictionaryArranged.Add(resultList2[i].Key, resultList2[i].Value);
        }

        return dictionaryArranged;
    }
```

Gambar 3.43 Potongan *source code Arrange Sequence Process*

3. GetNextState & Distinguish Tenses

```
public List<List<StateTransition>> GetNextState(List<List<StateTransition>> list, String transition)
{
    List<List<StateTransition>> newList = null;
    DataTable dt = null;
    List<StateTransition> tmpList = null;
    FiniteState state = null;
    StateTransition stateTransition = null;

    try
    {
        newList = new List<List<StateTransition>>();

        if (list.Count <= 0)
        {
            list.Add(new List<StateTransition>());
        }

        foreach (List<StateTransition> ls in list)
        {
            SqlCommand cmd = new SqlCommand("sp_finite_state_transition_get_next", Utility.GetConnection());
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.Add("@p_transition", SqlDbType.NVarChar).Value = transition;


            if (ls.Count == 0)
            {
                FiniteState startState = FiniteState.GetFirstNode();
                cmd.Parameters.Add("@p_start_state", SqlDbType.NVarChar).Value = startState.state;

                SqlDataAdapter da = new SqlDataAdapter(cmd);
                dt = new DataTable();
                da.Fill(dt);

                if (dt.Rows.Count > 0)
                {
                    foreach (DataRow dr in dt.Rows)
                    {
                        tmpList = new List<StateTransition>(ls);
                        bool flag = false;
                        if (dr["IS_FINAL_STATE"].Equals("1"))
                        {
                            flag = true;
                        }
                        state = new FiniteState(dr["STATE"].ToString(), dr["DESCRIPTION"].ToString(), flag);
                        stateTransition = new StateTransition(startState, transition, state);
                        tmpList.Add(stateTransition);
                        newList.Add(tmpList);
                    }
                }

                else
                {
                    cmd.Parameters.Add("@p_start_state", SqlDbType.NVarChar).Value = ls.Last().endState.state;

                    SqlDataAdapter da = new SqlDataAdapter(cmd);
                    dt = new DataTable();
                    da.Fill(dt);

                    if (dt.Rows.Count > 0)
                    {
                        foreach (DataRow dr in dt.Rows)
                        {
                            tmpList = new List<StateTransition>(ls);
                            bool flag = false;
                            if (dr["IS_FINAL_STATE"].Equals("1"))
                            {
                                flag = true;
                            }

                            state = new FiniteState(dr["STATE"].ToString(), dr["DESCRIPTION"].ToString(), flag);
                            stateTransition = new StateTransition(ls.Last().endState, transition, state);
                            tmpList.Add(stateTransition);
                            newList.Add(tmpList);
                        }
                    }
                    else
                    {
                        stateTransition = new StateTransition(FiniteState.GetFirstNode(), transition, FiniteState.GetUnknownNode());
                        ls.Add(stateTransition);
                        newList.Add(ls);
                    }
                }
            }

        return newList;
    }
```

Gambar 3.44 Potongan *source code GetNextState & Distinguish Tenses*

4. *Show Words*

```
public static DataTable SelectWord (string jenis, string kata, string arti)
{
    DataTable dt = null;
    try
    {
        SqlCommand cmd = new SqlCommand("sp_select", Utility.GetConnection());
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.Add("@p_jenis_kata", SqlDbType.NVarChar).Value = jenis;
        cmd.Parameters.Add("@p_kata_kata", SqlDbType.NVarChar).Value = kata;
        cmd.Parameters.Add("@p_arti", SqlDbType.NVarChar).Value = arti;

        SqlDataAdapter da = new SqlDataAdapter(cmd);
        dt = new DataTable();
        da.Fill(dt);
    }
```

Gambar 3. 45 Potongan *source code Show Words*

5. *Insert Word*

```
public static int InsertWord(string tabel, string kata, string arti)
{
    try
    {
        SqlCommand cmd = new SqlCommand("sp_insert", Utility.GetConnection());
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.Add("@p_jenis_kata", SqlDbType.NVarChar).Value = tabel;
        cmd.Parameters.Add("@p_kata", SqlDbType.NVarChar).Value = kata;
        cmd.Parameters.Add("@p_arti", SqlDbType.NVarChar).Value = arti;
        return cmd.ExecuteNonQuery();
    }
```

Gambar 3.46 Potongan *source code insert word*

```
public static int InsertVerb(string v1, string v2, string v3, string vs, string ving, string artia, string artip)
{
    try
    {
        SqlCommand cmd = new SqlCommand("sp_insert_verb", Utility.GetConnection());
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.Add("@p_verb_1", SqlDbType.NVarChar).Value = v1;
        cmd.Parameters.Add("@p_verb_2", SqlDbType.NVarChar).Value = v2;
        cmd.Parameters.Add("@p_verb_3", SqlDbType.NVarChar).Value = v3;
        cmd.Parameters.Add("@p_verb_s", SqlDbType.NVarChar).Value = vs;
        cmd.Parameters.Add("@p_verb_ing", SqlDbType.NVarChar).Value = ving;
        cmd.Parameters.Add("@p_arti_aktif", SqlDbType.NVarChar).Value = artia;
        cmd.Parameters.Add("@p_arti_pasif", SqlDbType.NVarChar).Value = artip;

        return cmd.ExecuteNonQuery();
    }
```

Gambar 3.47 Potongan *source code insert verb*

6. Delete Word

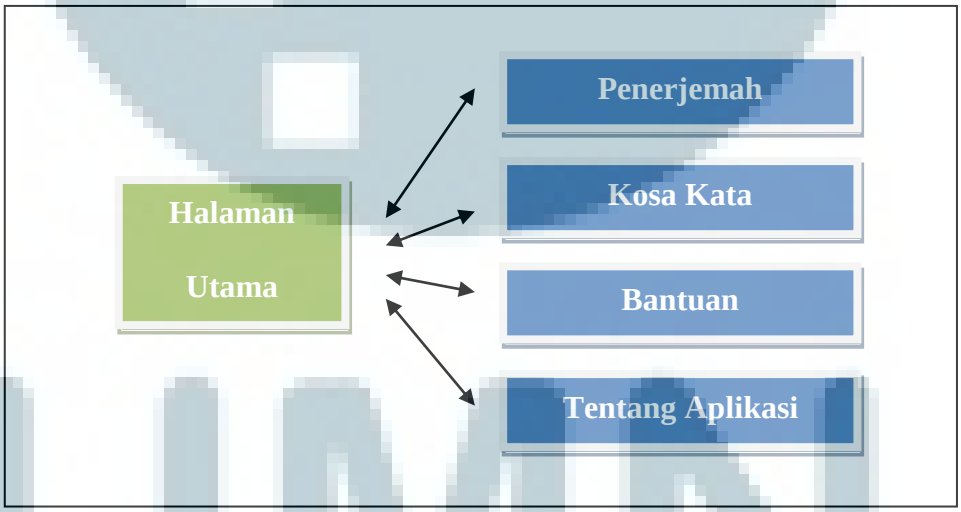
```
public static int DeleteWord(string tabel, int id)
{
    try
    {
        SqlCommand cmd = new SqlCommand("sp_delete", Utility.GetConnection());
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.Add("@origin_table", SqlDbType.NVarChar).Value = tabel;
        cmd.Parameters.Add("@ref_id", SqlDbType.Int).Value = id;
        return cmd.ExecuteNonQuery();
    }
```

Gambar 3.48 Potongan *source code delete word*

3.3 Perancangan Aplikasi

3.3.1 Struktur Navigasi Menu

Berikut adalah struktur navigasi menu yang dapat dipilih oleh pengguna saat menjalankan aplikasi.



Gambar 3.49 Navigasi Menu Aplikasi

Pada halaman utama aplikasi (halaman pertama yang tampil saat pengguna menjalankan aplikasi) terdapat tiga menu utama dengan rincian masing – masing menu adalah sebagai berikut.

1. Penerjemah

Pada menu ini akan ditampilkan halaman yang menjadi tempat pengguna untuk melakukan *input* kalimat dalam bahasa Inggris dan penerjemahan kalimat. Apabila kalimat sudah dimasukkan di kolom penerjemah dan pengguna sudah melakukan konfirmasi untuk penerjemahan, maka kalimat dalam bahasa Inggris tersebut akan diterjemahkan. Kalimat yang strukturnya dapat dikenali oleh aplikasi akan diinformasikan tentang jenis kalimat bahasa Inggrisnya (*tenses*) kepada pengguna beserta keterangan singkat tentang jenis kalimat tersebut.

2. Kosa Kata

Pada menu ini akan ditampilkan menu halaman di mana pengguna bisa melihat kata-kata apa saja yang dapat digunakan dalam aplikasi ini. Kata-kata tersebut diletakan per kategori sesuai jenis katanya. Pada menu ini pengguna juga dapat menambahkan kata baru atau melakukan perngubahan terhadap kata-kata yang sudah tersedia pada aplikasi ini.

3. Bantuan

Pada menu ini akan ditampilkan halaman yang berisikan bantuan mengenai tata cara penggunaan aplikasi dan cara memasukkan kalimat, serta syarat kalimat yang akan diterima oleh sistem dari aplikasi.

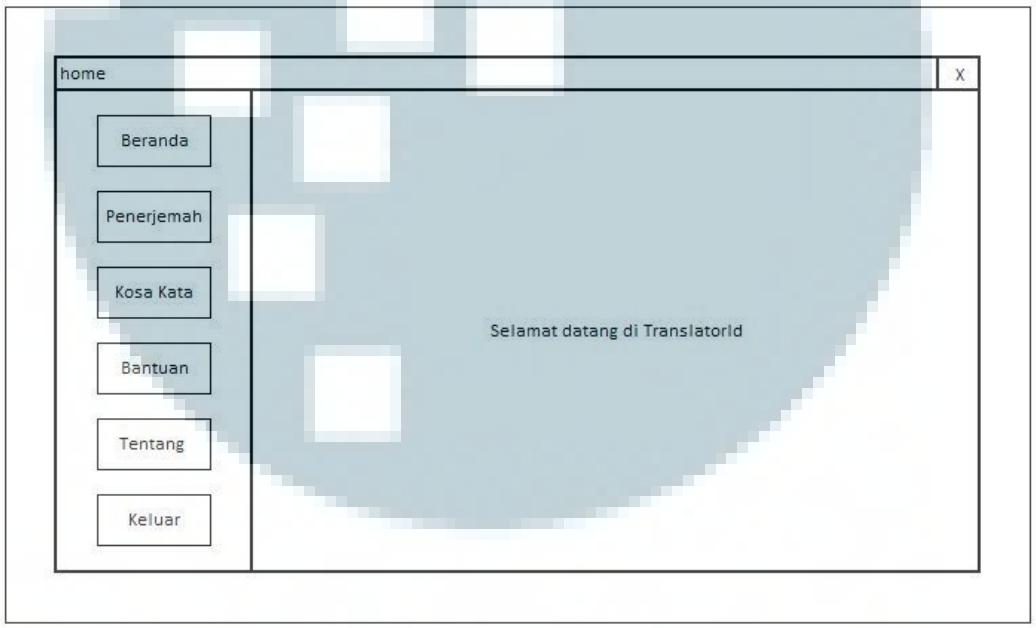
4. Tentang

Pada menu ini akan ditampilkan halaman yang berisikan informasi singkat mengenai aplikasi. Informasi yang ditampilkan berupa kegunaan aplikasi, serta informasi singkat mengenai pembuat aplikasi

3.3.2. Perancangan Antarmuka

Perancangan antar muka menjadi pedoman utama dalam membangun aplikasi ini secara keseluruhan dalam penelitian ini. Hasil implementasi dari perancangan ini akan menjadi *interface* aplikasi ini nantinya. Berikut adalah gambar-gambar perancangan sketsa *interface* aplikasi dalam penelitian ini.

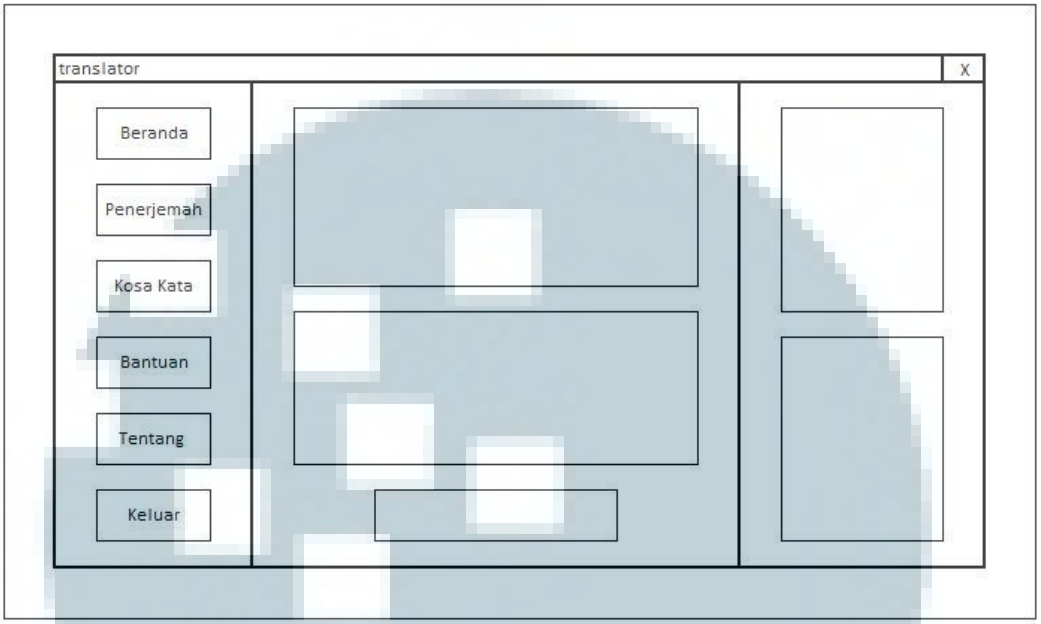
1. Halaman Utama



Gambar 3.50 Rancangan Tatap Muka Halaman Utama

Halaman beranda merupakan halaman yang pertama kali ditampilkan pada saat aplikasi ini dijalankan. Halaman ini menampilkan tulisan selamat datang kepada pengguna aplikasi.

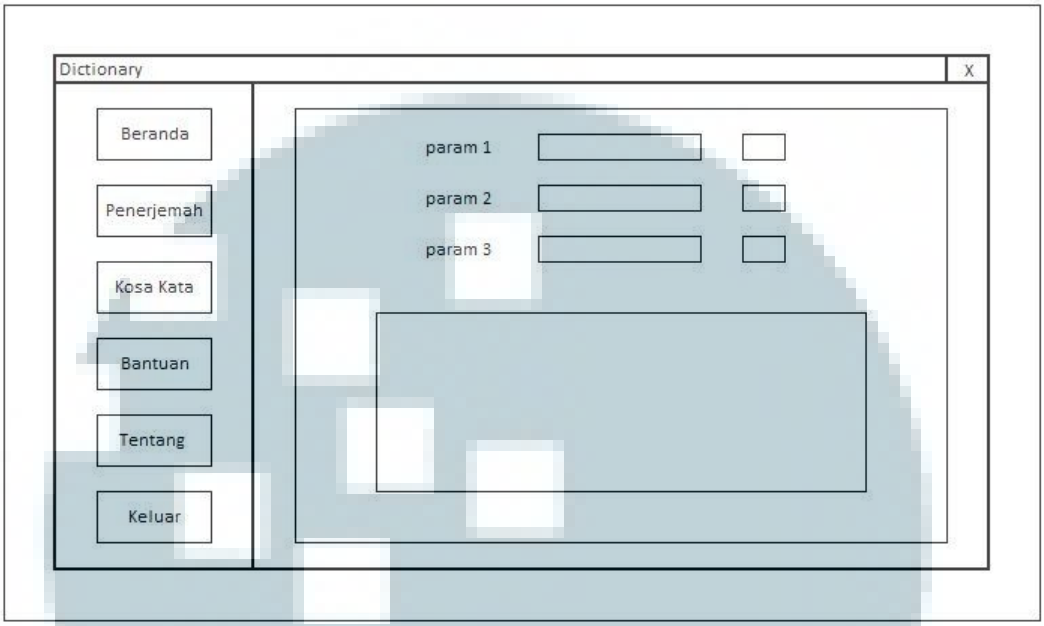
2. Halaman Penerjemah



Gambar 3.51 Rancangan Tatap Muka Penerjemah

Halaman penerjemah ini akan ditampilkan apabila pengguna melakukan klik pada tombol “Penerjemah” pada bagian kiri aplikasi. Halaman ini berisikan empat buah kotak teks, kotak teks pada posisi kanan atas merupakan tempat memasukkan kalimat bahasa Inggris pada aplikasi ini, kotak teks pada bagian kiri bawah menampilkan terjemahan, sedangkan kotak teks pada bagian kanan merupakan hasil dari identifikasi kalimat yang dimasukkan pengguna.

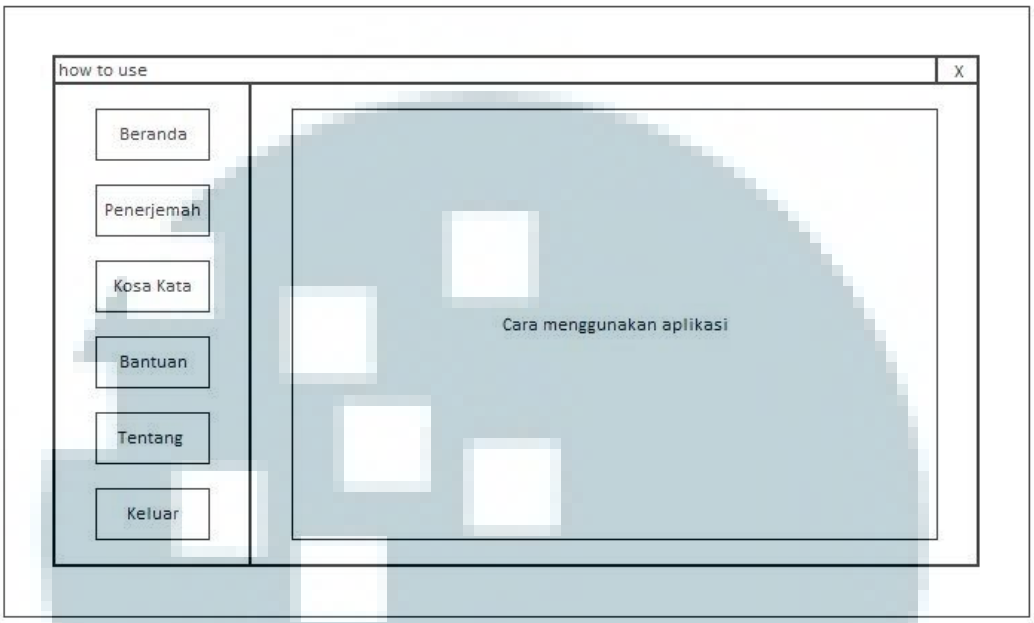
3. Halaman Kosa Kata



Gambar 3.52 Rancangan Tatap Muka Halaman Kosa Kata

Halaman kosa kata dapat diakses dengan melakukan klik pada tombol "Kosa Kata" di bagian kanan aplikasi. Menu ini dapat digunakan untuk melakukan penambahan dan penyuntingan kata. Pengguna juga dapat melihat kata-kata yang sudah terdaftar pada aplikasi TranslatorId di menu kosa kata ini.

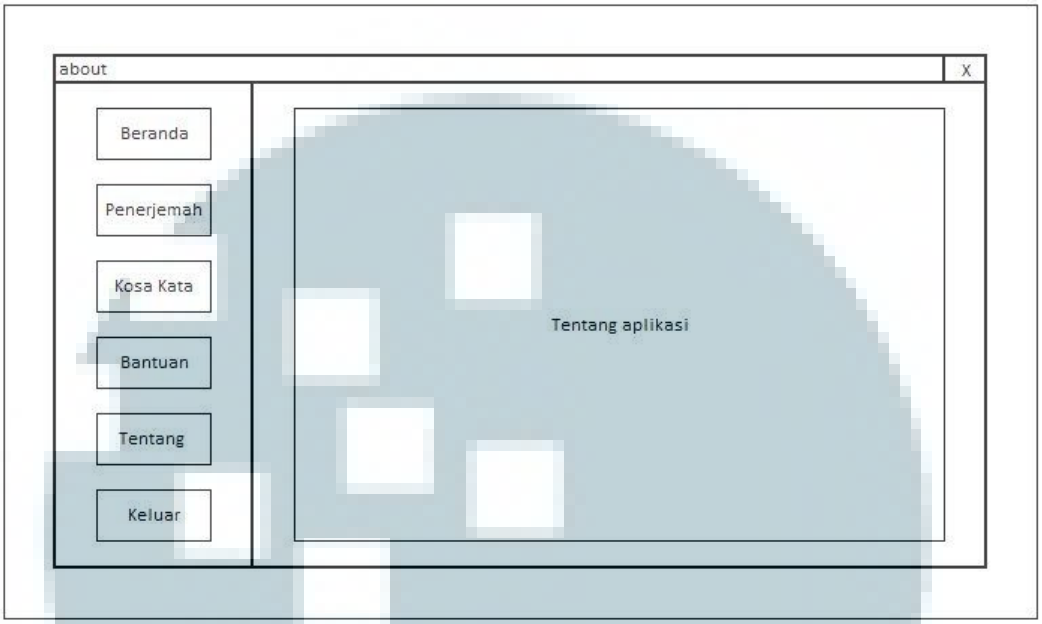
4. Halaman Bantuan



Gambar 3.53 Rancangan Tatap Muka Halaman Bantuan

Halaman bantuan akan ditampilkan ketika pengguna melakukan klik pada tombol "Bantuan" yang terdapat pada menu di bagian kiri aplikasi. Halaman menu ini menjabarkan tentang tata cara penggunaan aplikasi.

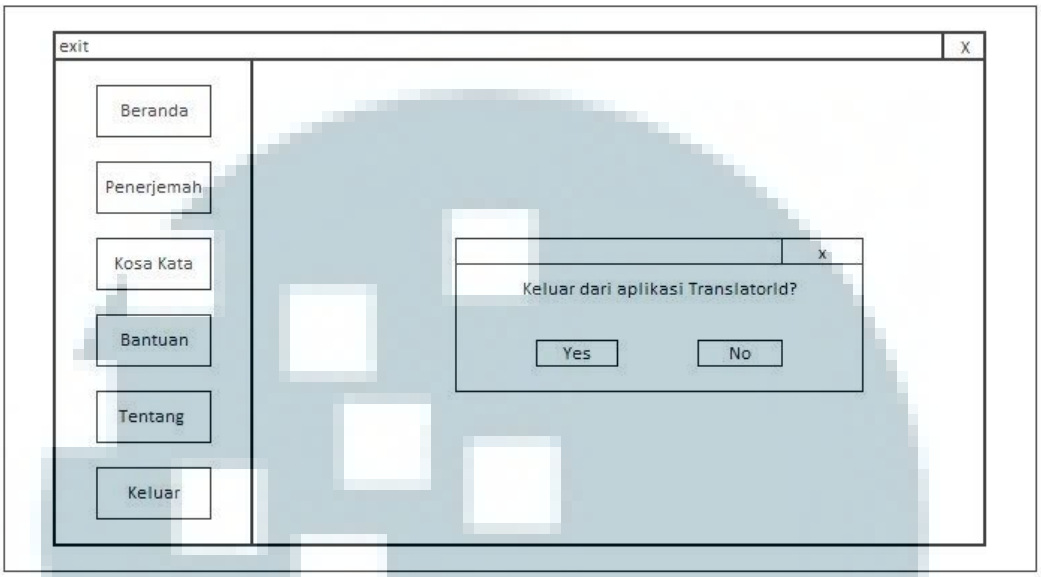
5. Halaman Tentang Aplikasi



Gambar 3.54 Rancangan Tatap Muka Tentang Aplikasi

Halaman tentang aplikasi akan ditampilkan ketika pengguna melakukan klik pada tombol "Tentang" yang terdapat pada menu di bagian kiri aplikasi. Pada halaman menu ini akan ditampilkan mengenai informasi kegunaan aplikasi, serta informasi singkat mengenai pembuat aplikasi.

6. Halaman Keluar dari Aplikasi



Gambar 3.55 Rancangan Tatap Muka Keluar dari Aplikasi

Halaman tentang aplikasi akan ditampilkan ketika pengguna melakukan klik pada tombol “Keluar” yang terdapat pada menu di bagian kiri aplikasi. Pada halaman dilakukan konfirmasi apabila pengguna yakin keluar dari aplikasi TranslatorId.