PERANCANGAN PERANGKAT LUNAK KAMUS BAHASA ARAB INDONESIA DENGAN METODE PARSING

TUGAS AKHIR

Disusun oleh

IMAM KUSWARDAYAN
NRP 2695100036

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2001

PERANCANGAN PERANGKAT LUNAK KAMUS BAHASA ARAB INDONESIA DENGAN METODE PARSING

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing :

Pembimbing I	Pembimbing II
Ir. Aris Tjahjanto, M.Kom	Drs. Wahyuddin
NIP. 131 933 299	NIP. 131 577 240

SURABAYA,
Februari 2001

ABSTRAK

Pada aturan kata dalam Bahasa Arab, sebuah kata bisa berubah menjadi puluhan hingga ratusan bentuk, yang masing-masing memiliki arti sendiri-sendiri. Pola perubahan bentuk kata tersebut memiliki aturan-aturan yang jelas dan pasti. Untuk menghemat penyimpanan database kamus penerjemah, perlu dirancang bagaimana agar kamus tersebut cukup hanya menyimpan dalam bentuk dasar setiap kata yang bisa mengalami perubahan dalam bahasa Arab.

Metode parsing, bisa digunakan untuk mengenal pola kata Bahasa Arab dengan mengambil informasi *action* reduksi *production* mana saja yang telah sukses dilakukan untuk suatu masukan kata berbahasa Arab yang telah dikodekan menjadi token-token. Penggunaan database baik untuk *production rules* maupun kamus, memungkinkan aplikasi kamus Arab Indonesia untuk dikembangkan tanpa harus mengubah *source program*.

Ujicoba dilakukan dengan data sampling production rules yang mewakili perubahan kata berpola *wazn fa'ala* dengan *bina' shohih* serta beberapa kata bentuk dasar yang dijadikan data kamus. Dengan memasukkan suatu kata yang telah berubah bentuk dari bentuk asalnya –yang mana bentuk asalnya terdapat dalam kamus-, perangkat lunak ini mampu mendefinisikan kata tersebut serta memberikan keterangan-keterangan yang berhubungan dengan pola kata tersebut.

KATA PENGANTAR

Bismillahirrohmaanirrohiim...

Segala puji bagi Allah, karena berkat karuniaNya-lah kami dapat menyelesaikan tugas akhir ini. Setelah meniti waktu-waktu yang berlalu dengan sibuk berkutat dengan buku-buku diktat dan berhadapan dengan komputer, akhirnya kami bisa menyelesaikannya.

Bermula dari tujuan penulis yang ingin menyumbangkan ilmu pengetahuan komputer yang didapat dari bangku kuliah selama ini untuk kepentingan orang banyak -sebagai amalan fardhu kifayah-, maka topik Tugas Akhir ini dipilih. Hal ini juga mengingat keberadaan sebagian besar kaum muslimin di Indonesia yang kesulitan untuk memahami Bahasa Arab, padahal Bahasa Arab digunakan dalam ibadah ritual dan referensi mereka.

Sebagaimana harapan penulis, semoga pengembangan Tugas Akhir ini bisa bermanfaat bagi ummat Islam Indonesia pada khususnya dan orang Indonesia pada umumnya. Tentu saja, usaha yang telah dilakukan oleh penulis ini, tak luput dari kekurangan ataupun kesalahan. Sebab itu, kritik untuk pengembangan senantiasa diharapkan oleh penulis untuk kesempurnaan Aplikasi Tugas Akhir ini.

Tak lupa untuk mengucapkan terima kasih kepada pihak-pihak yang telah membantu untuk menyelesaikan Tugas Akhir ini:

1. **Bapak Arif Djunaedy**, selaku Ketua Jurusan Tehnik Informatika ITS, sekaligus sebagai dosen penguji yang telah memberikan masukan berharga dalam pengembangan karya tulis yang ilmiah.
2. **Bapak Aris Tjahyanto**, selaku dosen Pembimbing I, yang telah memberikan bimbingan dan masukan dalam hal teknik kompilasi serta memberikan motivasi kepada penulis untuk segera lulus.

3. **Bapak Wahyuddin**, selaku dosen MKDU yang menjadi dosen Pembimbing 2, yang telah memberikan masukan dan bimbingan dalam hal Bahasa Arab.
4. **Bapak Rully Sulaiman**, selaku dosen wali yang selalu memberikan motivasi untuk segera lulus serta memberikan tawaran untuk membantu penyelesaian Tugas Akhir ini.
5. **Bapak Victor, Pak Dwi, Pak Fajar, Bu Esther, Pak Hari Ginardi, serta dosen lainnya** yang telah menguji penulis pada saat seminar dan demo program dan memberikan masukan-masukan yang bermanfaat untuk pengembangan Tugas Akhir ini.
6. **Bapak Royyana**, selaku dosen TC yang memberikan motivasi penulis agar segera lulus dan menjadi dosen di TC.
7. **Ayah dan ibu** yang telah mendorong penulis untuk bersegera menyelesaikan tugas akhir. Terima kasih atas dukungan moril, doa dan finansial selama kuliah. Maafkan kalau sempat *molor* 1 tahun. Kini tampaknya sudah waktunya bagi anakmu untuk membalas kebaikan selama ini.
8. **Akhi Abdulloh Hasan, Ukhti Suli Mardhiyyah, Adek, Nak Pipit** dan lainnya yang telah mendoakan agar penulis diberi kelancaran dan kemudahan oleh Allah Subhanahu Wata'ala dalam mengerjakan tugas akhir ini.
9. **Akhi Abdulloh Hasan, Gatot Ariwibowo dan Budi Santoso** yang telah membantu untuk memberikan referensi baik program pembanding ataupun narasumber yang berhubungan dengan kelancaran pembuatan Tugas Akhir ini. Terima kasih pula atas dukungan dan motivasinya agar tugas akhir ini bisa bermanfaat bagi kaum muslimin di Indonesia. Insya Allah penulis akan melanjutkan tugas akhir ini sebagaimana yang antum harapkan semua.
10. **Tathit, Sal-man, Uli, I'in, Silvia dan Dwi Puspitasari**, yang telah meminjamkan buku-buku referensi tentang kompilasi dan Bahasa

Arab, ataupun menawarkan diri untuk menjadi penguji program dan berdiskusi bersama.

11. **Sesama TA-ers**; Hermanu, Salim, Ahmed, Gandi, David, Yudo, Fitri, Mukib, Luluk yang saling bantu-membantu memberikan informasi dan motivasi bersama untuk pengerjaan tugas akhir. Sayang sekali kawan, sebagian anda belum beruntung...
12. **Kru LabProg dan Puskom**; Kenji, HoE, Cphee, Dimas, Ade, Mbah Nur, Ajun, Ma-man, Ciwok, Yusuf, Pak Arif, Budi yang menemani penulis untuk 'melekan' mengerjakan Tugas Akhir.
13. **Para 'Hantu' LabProg**; CahNdeso, ~!@#$%, Prof. Amien Rais, Anisah The Great, Jendral Anu, Sego Pecel / Kucing Abang, Darderdor, King Paimo dan lainnya yang menjadi 'provokator' untuk tidak mudah berputus asa menghadapi komputer. Ingat kawan, 'the legendary "Prince of Ghost"' pernah masuk sejarah LabProg Informatika ITS ☺.
14. **Para 'penagih traktiran'** yang secara tidak langsung memberi motivasi kepada penulis untuk segera lulus; Kenji, Mbah Nur, Heri Superman, Aris, Ajun, Yudi, Bu Vita, Ita, Alive, Nurita, Uli, I'in, kru Pikti, dan lainnya yang kelupaan. Doakan saja lagi punya uang sehingga bisa nraktir. Kalau ndak punya, bagi-bagi permen saja ya... ☺.
15. Pihak-pihak yang tidak bisa disebutkan satu-persatu di sini yang telah membantu dan memotivasi. Penulis ucapkan terima kasih yang sebesar-besarnya kepada anda semuanya. Semoga Allah membalas kebaikan anda semua.

Surabaya Februari 2001

Imam Kuswardayan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Bahasa Arab merupakan Bahasa ummat Islam, sehingga untuk memahami agama Islam penguasaan bahasa Arab merupakan hal yang sangat penting, karena sebagian besar pedoman ajaran agama Islam ditulis dengan bahasa Arab.

Mengingat pentingnya bahasa Arab, diperlukan bagi seseorang muslim untuk mengerti akan artinya. Berbeda dengan bahasa yang lain, dalam bahasa Arab bentuk dasar dari suatu kata bisa diturunkan menjadi lebih dari seratus kata sehingga yang terdapat dalam kamus bahasa Arab adalah bentuk dasarnya dan beberapa bentuk kata turunannya.

Kesulitan yang biasanya dihadapi dalam mencari suatu kata dalam kamus bahasa Arab adalah apabila kata yang dicari bukan merupakan bentuk dasar, karena kata tersebut dicantumkan di bawah bentuk dasarnya, atau malah tidak dicantumkan sama sekali. Hal ini disebabkan banyaknya turunan kata dari satu bentuk kata dasar.

1.2 TUJUAN DAN MANFAAT TUGAS AKHIR

1. Menyajikan suatu konsep cara menganalisa pola pada suatu kata bahasa Arab.
2. Membuat sebuah perangkat lunak yang mampu menterjemahkan kata berbahasa Arab walaupun sudah mengalami perubahan dari bentuk asal.

3. Sebagai untuk sarana membantu mempelajari Bahasa Arab bagi pemula terutama dalam hal At Tashrif (Perubahan bentuk kata).
4. Menjadi referensi bagi penelitian selanjutnya tentang Perangkat lunak yang digunakan sebagai sarana belajar bahasa Arab.

1.3 PERUMUSAN MASALAH

Dalam perancangan perangkat lunak Kamus Bahasa Arab Indonesia ini terdapat permasalahan-permasalahan sebagai berikut:

- Bagaimana metode yang dilakukan untuk pengenalan pola kata berbahasa Arab
- Bagaimana cara yang harus dilakukan agar pengenalan pola kata bahasa Arab bisa berkembang dan bisa dimodifikasi andaikata terdapat penyempurnaan.
- Bagaimana memilih *Parser* yang efisien untuk menangani pengenalan pola
- Bagaimana mengambil bentuk dasar dari kata masukan yang telah diketahui polanya, yang kemudian akan dicocokkan dengan data kamus.
- Bagaimana data kamus bisa berkembang serta bisa dimodifikasi sebagai langkah penyempurnaan.

1.4 PEMBATASAN MASALAH

Aplikasi ini mempunyai batasan-batasan sebagai berikut :

- Aplikasi hanya bisa dijalankan pada MS Windows 95/98 dg fasilitas Arabic enabled, ataupun Windows 2000.
- Cakupan bentuk kata yang bisa diterjemahkan, jika masukan berupa kata jadian (bukan bentuk dasar), tergantung dari database rules yang telah dimasukkan.

- Database rule untuk TA ini hanya mencakup untuk wazn tsulatsy dengan bina' shohih saja.

1.5 METODE PENYUSUNAN TUGAS AKHIR

1. Studi literatur

   Pada tahap ini akan dipelajari bahan-bahan yang diperlukan untuk proses desain. Antara lain adalah memahami pola perubahan kata (tashrif) pada bahasa arab. Termasuk juga mempelajari konsep parsing bahasa.

2. Desain program

   Pada tahap ini dilakukan desain konsep dan algoritma yang memenuhi kebutuhan dari tujuan perangkat lunak ini, untuk kemudian diterapkan dalam proses pembuatan/ coding program. Selain itu juga mendefinisikan rule-rule bahasa Arab yang akan dipakai dalam aplikasi ini.

3. Pembuatan / *coding* program

   Pada tahap ini implementasi dari tahap sebelumnya dilakukan dengan membuat/ coding program untuk perangkat lunak yang akan dihasilkan.

4. Test dan evaluasi

   Pada tahap ini konsep, algoritma, dan teknis dari pembuatan perangkat lunak diuji kebenarannya. Hasil / output dari perangkat lunak ini juga dievaluasi dengan menggunakan perangkat lunak lainnya.

5. Perbaikan dan Penyempurnaan Aplikasi

   Apabila dalam proses pengujian ditemukan kesalahan atau kekurangan maka perlu dilakukan perbaikan dan penyempurnaan. Setelah itu dilakukan lagi

pengujian sampai aplikasi bisa benar-benar sempurna dan sudah layak dipakai sesuai dengan tujuannya.

6. Pembuatan Dokumentasi

   Pada tahap akhir dibuat suatu dokumentasi lengkap tentang aplikasi yang telah dibuat, dalam hal ini dibuat dalam bentuk Buku Tugas Akhir.

1.6 SISTEMATIKA PEMBAHASAN

Penulisan tugas akhir ini dibagi menjadi 7 Bab dengan sistematika sebagai berikut:

BAB I berisi pendahuluan, menguraikan tentang latar belakang, permasalahan, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan Tugas Akhir.

BAB II berisi tentang Dasar-dasar teori pendukung dari tugas akhir. Yaitu menjelaskan konsep tentang grammer, production, parsing serta penerapannya.

BAB III berisi tentang sekilas pengetahuan tentang Bahasa Arab, mencakup kata dan perubahan kata.

BAB IV berisi tentang penjelasan permasalahan secara lebih detail serta metodologi yang digunakan sebagai pemecahan dari aplikasi tugas akhir yang dibuat.

BAB V berisi tentang perancangan dan implementasi perangkat lunak, berupa input-output, antarmuka serta proses.

BAB VI berisi tentang hasil uji coba dan analisa dari perangkat lunak yang telah dibuat.

BAB VII berisi tentang kesimpulan yang didapat dari uji coba perangkat lunak serta kemungkinan pengembangan dari perangkat lunak.

BAB II

DASAR TEORI

BAB II

DASAR TEORI

2.1 PARSER

Sebuah parser pada grammar *G* adalah suatu instruksi yang mengambil suatu string *w* dan menghasilkan parser tree lain untuk *w*, jika *w* adalah *sentence* dari *G* atau pesan salah yang menyatakan bahwa *w* bukan kalimat dari *G*. Seringkali parse tree hanya menghasilkan pengertian kiasan, pada kenyataannya parse tree merupakan sebuah rangkaian dari perintah-perintah yang mempunyai langkah-langkah yang bersambungan dari proses konstruksi. Dua tipe dasar dari parser untuk *Context Free Grammar* (CFG) adalah bottom-up parser dan top down parser. Sesuai dengan namanya maka bottom up membangun parser tree dari bawah (daun) ke atas (root), sedangkan top down parser dibangun dari akar menuju daun. Pada kedua pilihan ini maka input parser diamati dari kiri ke kanan, satu simbol setiap waktu.

Metode parsing bottom-up disebut "shift reduce" parsing karena terdiri pergeseran atau perpindahan stack sampai bagian kanan muncul production di atas stack. Sisi sebelah kanan dihapus dengan perintah (*reduced to*) simbol dari left side (sisi kiri) dari production kemudian proses diulangi.

Gambaran dari Parse Tree

Pada bagian ini menjelaskan jenis dari output yang diproduksi oleh parser dalam compiler. Pada bagian ini kita boleh menghapus output parser sebagaimana yang digambarkan sesuai dengan parse tree untuk input jika input benar. Ada dua jenis parse

tree yaitu implisit dan eksplisit. Rangkaian production digunakan dalam beberapa derivasi adalah contoh dari implisit. Sebuah struktur link list untuk parse tree merupakan gambaran eksplisit.

Pemanggilan kembali derivation yang merupakan leftmost nonterminal pada setiap langkah dikatakan menjadi leftmost. Jika $\alpha \rightarrow \beta$ dalam leftmost nonterminal dimana α selalu diganti, sehingga dapat ditulis bahwa $\alpha \rightarrow \beta$. Setiap langkah leftmost dengan menggunakan notasi konvensi mempu yai bentuk $wA\gamma \rightarrow w\delta\gamma$ dimana w adalah terminal. Jika α mendapatkan β pada leftmost derivation dapat ditulis $\alpha \rightarrow \beta$ Jila $S \rightarrow \alpha$, kemudian dapat disebut *left-sentential form* dari sisi grammar. Definisi ini bisa disebut righmost derivation yang artinya setiap nonterminal yang ada disebelah kanan pada setiap langkah selalu digantikan. Kadang-kadang rightmost derivation disebut dengan *canonical* derivation.

Setiap kalimat dalam bahasa mengandung dua kemungkinan yaitu leftmost dan rightmost derivasi. Untuk menemukan satu leftmost derivation sebuah kalimat (sentence) w, kita dapat mengambil salah satu dari w dan membentuknya sesuai dengan parse tree *T* yang berkesesuaian. Dari T kita dapat membuat sebuah leftmost derivasi dengan top down tree dengan garis melintang. Kita dapat memulainya dengan simbol awal S, yang sesuai dengan akar dari T. Sehingga dapat kita bangun sebuah derivasi leftmost berikut:

$$S = \alpha_1 \xrightarrow[lm]{} \alpha_2 \xrightarrow[lm]{} \ldots \xrightarrow[lm]{} \alpha_n = w$$

yang berkesesuaian dengan T, dengan menggunakan prosedur berikut :

Jika root diberi label S mempunyai anak A, B dan C maka dengan menggunakan teori leftmost derivasi dapat dituliskan sebagai : S → ABC dimana S adalah α_1 sedangkan ABC adalah α_2.

Jika node A mempunyai children dengan label XYZ dalam T, maka dapat dibangun langkah berikutnya yaitu dengan mengganti A sesuai dengan children-nya yaitu ABC → XYZBC dimana XYZBC adalah α_3 kita lanjutkan dengan meneruskan dengan menemukan sebuah node nonterminal D dengan α dan mengganti D dengan children-nya dalam T untuk mengisi α_{i+1}dimana i = 1, 2, …, n-1.

Jika kita ingin membangun parse tree , maka kita pilih node berikutnya sesuai dengan production dari leftmost derivasi.

Contoh

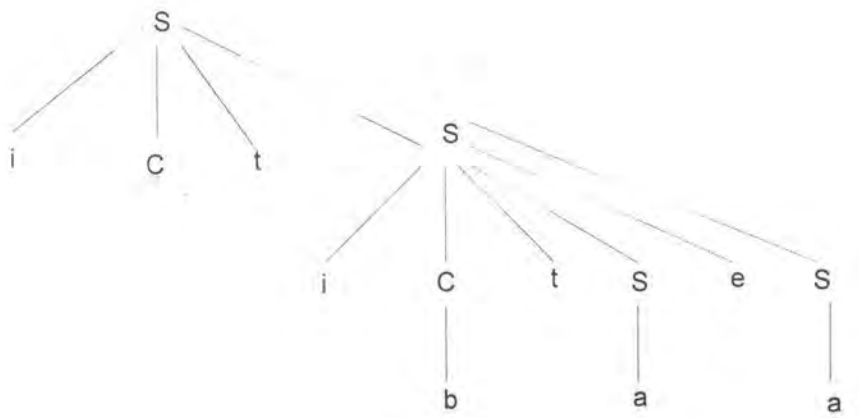
(1)S → i C t S

(2)S → i C t S e S

(3)S → a

(4)C→ b

i, t dan e adalah if, then, else sedangkan C dan S adalah 'conditional' dan 'statement'.



Gambar 2.1 Parse Tree T

Kita bisa membangun derivasi leftmost untuk sentence w= i b t i b t a e a. Sebuah parse tree T untuk w ditunjukkan dalam gambar 2.1. leftmost derivasi yang sesuai untuk parse tree diatas adalah :

$S \rightarrow i\ C\ t\ S$

$S \rightarrow i\ b\ t\ S$

$S \rightarrow i\ b\ t\ i\ C\ t\ S\ e\ S$

$S \rightarrow i\ b\ t\ i\ b\ t\ S\ e\ S$

$S \rightarrow i\ b\ t\ i\ b\ t\ a\ e\ S$

$S \rightarrow i\ b\ t\ i\ b\ t\ a\ e\ a$

Sebuah righmost derivasi dapat dibangun dari parse tree yang sama. Pada setiap langkah kita ganti satu nonterminal paling kanan dengan label children-nya. Sebagai contoh dua langkah derivasi rightmost dari figure 2.1 menjadi $S \rightarrow i\ C\ t\ S \rightarrow i\ C\ t\ i\ C\ t\ S\ e\ S$.

Sekarang menjadi jelas bahwa baik leftmost maupun righmost mempunyai parse tree yang sama, sehingga dapat dengan mudah kita mengubah leftmost (righmost) derivasi ke parse tree atau sebaliknya.

2.2 SHIFT REDUCE PARSING

Pada bagian ini akan dibahas jenis bottom-up parsing yang disebut bottom-up karena ia membentuk parse tree untuk input string dari daun (bottom) dan naik ke atas menuju root (akar). Proses ini bisa dikatakan "reducing" string menuju start simbol dari suatu grammar. Tiap-tiap langkah suatu string menyesuaikan dari sisi kanan production yang kemudian diganti dengan simbol yang sebelah kiri. Sebagai contoh diberikan grammar berikut :

$S \rightarrow aAcBe$

$A \rightarrow Ab|b$

$B \rightarrow d$

Dan string *abbcde*. Kita ingin mereduksi string ini menjadi S. Kita amati abbcde cari substring yang sesuai dengan sisi kanan pada production. Substring b dan d memenuhi syarat. Sekarang kita pilih leftmost b dan menggantikannya dengan A, sisi kiri dari production $A \rightarrow b$ kita memperoleh string aAbcde. Kita sekarang menemukan Ab, d dan e yang masing-masing sesuai dengan sisi kanan dari production. Kita dapat mengganti Ab dengan A sesuai dnegan production $A \rightarrow Ab$. Sehingga diperoleh aAcde. Kemudian d diganti menjadi B, sesuai dengan sisi kiri dari production $B \rightarrow d$, diperoleh aAcBe. Sekarang kita string ini dengan S.

Penggantian sisi kanan oleh production dengan sisi kiri diatas disebut dengan reduksi. Dengan empat kali reduksi kita dapat mengganti abbcde menjadi S. Reduksi ini pada kenyataannya adalah mengikuti rightmost derivasi dalam inverse-nya.

Substring sisi kanan productionnya diganti dengan substring dari sisi kiri pasti suatu reduksi menuju start simbol dengan kebalikan derivasi rightmost, yang disebut "handle". Proses parsing bottom-up mungkin dilihat sebagai salah satu finding dan reducing handle.

Dalam beberapa kasus substring leftmost β yang sesuai dengan sisi kanan dari beberapa production A → β tidak handle karena reduksi dengan production A→ β mungkin menghasilkan string yang tidak dapat direduksi menuju start simbol. Sebagai contoh jika kita ingin mengganti b dengan A pada string kedua aAbcde kita akan memperoleh sebuah string aAAcde yang tidak dapat direduksi menjadi S. Untuk alasan ini kita harus memberikan definisi yang tepat apa itu "handle". Kita dapat melihat bahwa jika kita menulis derivasi righmost dalam inverse-nya, kemudian dilanjutkan membuat perubahan dalam derivasi yang asli dengan peletakkan yang tepat untuk mereduce sentence menjadi start simbol.

Handles

Sebuah handle pada bentuk right-sentential γ adalah production A → β dan posisi γ dimana string β mungkin ketemu dan digantikan oleh A untuk menghasilkan bentuk right-sentential sebelumnya dalam rightmost derivasi γ, jika S → αAw → αβw, maka A → β dalam posisi α adalah handle pada αβw. String w ke kanan dari handle berisi hanya simbol terminal.

Sebagai contoh string abbcde adalah bentuk right-sentential yang mempunyai handle A → b pada posisi 2. Demikian juga aAbcde adalah bentuk right-sentential yang handlenya A → Ab pada posisi 2.

Kadang-kadang bisa dikatakan substring β adalah handle dari αβw jika posisi β dan production A → β akan kita hapus. Jika grammar satu maka setiap bentuk right-sentential merupakan grammar yang hanya punya satu handle.

Contoh 2.2. Diberikan grammar berikut :

(1) $E \rightarrow E + E$

(2) $E \rightarrow E * E$

(3) $E \rightarrow (E)$

(4) $E \rightarrow id$

Dan diberikan derivasi rightmost

$E \rightarrow \underline{E + E}$

$\rightarrow E + \underline{E * E}$

$\rightarrow E + E * \underline{id_3}$

$\rightarrow E + \underline{id_2} * id_3$

$\rightarrow \underline{id_1} + id_2 * id_3$

Kita mempunyai subscript id's untuk notasi convenience dan digarisbawahi sebuah handle pada bentuk rihgt-sentential . Sebagai contoh id_1 adalah handle dari bentuk right-sentential $id_1 + id_2 * id_3$ karena id adalah sisi kanan dari production E → id dan penggantian id1 oleh produces E sebelum bentuk right sentential $E + \underline{id_2} * id_3$ .Dengan catatan string kelihatan ke kanan dari handle yang hanya berisi simbol terminal.

Karena grammar 2.2 rangkap, ada rightmost derivasi lain pada string yang sama. Derivasi dimulai E * E dan menghasilkan handle lainnya. Pada faktanya E + E adalah

handle dari E + E * id_3 yang sesuai dengan derivasi ini, hanya id3 sendiri adalah handle pada bentuk righ-sentential yang sama untuk derivasi di atas.

2.3 IMPLEMENTASI STACK PADA PARSING SHIFT-REDUCE

Ada 2 masalah yang harus diselesaikan jika kita akan melakukan *automata* parsing dengan handle purning, pertama adalah bagaimana menempatkan handle dalam bentuk right-sentential dan kedua adalah memilih satu production dari sisi kanan yang sesuai.

Cara cepat untuk mengimplementasikan parser shift-reduce dengan menggunakan sebuah stack dan sebuah input buffer, kita bisa menggunakan $ untuk menandai 'bottom' stack dan akhir input string.

Stack	Input
$	*w* $

Operasi parser dengan *shifting* nol (zero) atau lebih input simbol ke stack sampai sebuah handle β berada di atas stack. Parser kemudian mereduksi *β* sesuai dengan sisi kiri dari production yang berkesesuaian. Parser mengulangi proses ini sampai ditemukan error atau sampai stack berisi start simbol dan inputnya kosong :

Stack	Input
$ S	$

Pada susunan ini parser berhenti dan proses parsing sudah selesai.

Contoh 2.4

Diberikan sebuah input string $id_1 + id_2 * id_3$ yang sesuai dengan grammar (2.2), menggunakan derivasi pada contoh 2.2. Maka langkah-langkahnya ditunjukkan pada figure berikut. Dengan catatan bahwa grammar (2.2) mempunyai dua rightmost derivasi untuk input ini sedangkan lainnya urutan langkah shift-reduce parser.

	Stack	Input	Action
(1)	\$	$id_1 + id_2 * id_3$\$	shift
(2)	\$ id_1	$+ id_2 * id_3$\$	reduce by E → id
(3)	\$ E	$+ id_2 * id_3$\$	shift
(4)	\$ E +	$id_2 * id_3$\$	shift
(5)	\$ E + id_2	$* id_3$\$	reduce by E → id
(6)	\$ E + E	$* id_3$\$	shift
(7)	\$ E + E *	id_3\$	shift
(8)	\$ E + E * id_3	\$	reduce by E → id
(9)	\$ E + E * E	\$	reduce by E → E * E
(10)	\$ E + E	\$	reduce by E → E + E
(11)	\$ E	\$	accept

Tabel 2.1 Shift-reduce parsing action

Pada saat operasi penting dari parser adalah shift dan reduce, ada empat kemungkinan action shift-reduce parser yang dapat dibuat: (1) shift, (2) reduce, (3) accept, (4) error.

1. shift, memasukkan input simbol berikutnya ke dalam stack
2. reduce, mengganti isi stack paling atas dengan handlenya yang berupa non terminal.
3. Accept, proses parsing telah selesai.
4. Error, parser menemukan *syntax error*, yang menyebabkan pemanggilan routine error.

Ada hal penting yang harus diperhatikan dalam shift-reduce parsing bahwa handle pada akhirnya selalu berada di stack paling atas, tidak pernah di bawah. Kenyataan ini menjadi jelas ketika kita mengetahui kemungkinan dari dua urutan step pada derivasi rihgtmost. Ada dua step dapat dibentuk.

(1) $S \rightarrow \alpha A z$

$\rightarrow \alpha \beta B y z$

$\rightarrow \alpha \beta \gamma y z$

(2) $S \rightarrow \alpha B x A z$

$\rightarrow \alpha B x y z$

$\rightarrow \alpha \gamma x y z$

Pada pilihan (1), A digantikan oleh $\beta B y$, dan selanjutnya rightmost nonterminal B pada sisi kanan digantikan oleh γ. Pada pilihan (2), A yang pertama digantikan, tetapi pada waktu ini sisi kanan adalah string y yang hanya berupa terminal saja. Rightmost selanjutnya adalah nonterminal B, tempatnya disebelah kiri y.

Kebalikan dari pilihan (1), dimana shift-reduce parser hanya menjangkau konfigurasi:

Stack	Input
$ α β γ	y z $

Parser sekarang mereduksi handle γ menjadi B untuk memperoleh konfigurasi

Stack	Input
$ α β B	y z $

B adalah rightmost non terminal dalam α β B y z, adalah handle kanan paling akhir dari α β B y z yang tidak terjadi di sebelah stack. Parser dapat melakukan shift string *y* ke stack dengan untuk memperoleh konfigurasi.

Stack	Input
$ α β B y	z $

dimana β B y adalah sebuah handle, dan digunakan untuk mereduksi A.

Pada pilihan (2), konfigurasinya adalah.

Stack	Input
$ α y	x y z $

Handle *y* berada di stack teratas. Setelah mereduksi handle γ menjadi B, parser dapat memasukkan string *xy* dan mendapatkan handle *y* selanjutnya di stack paling atas.

Stack	Input
$ α B x y	z $

Sekarang parser mereduksi *y* menjadi A.

Pada kedua kasus ini, setelah melakukan reduksi parser shift zero atau lebih simbol untuk mendapatkan handle selanjutnya di stack. Parser tidak menuju stack sebelum menemukan handle. Ini yang membuat handle purning sesuai dengan struktur data pada implementasi shift-reduce parser.

FIRST AND FOLLOW

Sekarang kita pelajari bagaimana mengisi entry ke dalam *predictive parsing table.* Kita memerlukan dua fungsi yang tergabung dalam grammar *G.* Fungsi ini adalah FIRST dan FOLLOW, yang akan membuat indikasi entry yang sesuai ke dalam tabel untuk *G,* jika parsing *G* ada. Jika α adalah sebuah string simbol grammar, FIRST (α) adalah sekumpulan terminal yang memulai string diturunkan dari α. Jika $\alpha \rightarrow \epsilon$, maka ϵ juga dalam FIRST (α).

Definisi *FOLLOW* (*A*), untuk nonterminal *A* adalah sekumpulan terminal α yang dapat menuju ke kanan dari *A* dalam beberapa bentuk sentential. $S \rightarrow \alpha A \alpha \beta$ untuk beberapa α dan β. Jika *A* menjadi right simbol dalam beberapa bentuk sentential, maka tambahkan $ ke FOLLOW (*A*).

Untuk mencari FIRST (X) pada semua grammar simbol X, tambahkan aturan sampai tidak ada lagi terminal atau ϵ yang dapat ditambahkan ke himpunan FIRST.

1. Jika X adalah terminal, maka FIRST(X) adalah (X)
2. Jika X adalah nonterminal dan X $\rightarrow$ a α adalah production, maka tambahkan a ke FIRST (X). Jika $x \rightarrow \epsilon$ adalah sebuah production, maka tambahkan ϵ ke FIRST (X).
3. Jika $X \rightarrow Y_1Y_2 \ldots Y_k$ adalah production maka untuk semua I yang termasuk di dalam $X \rightarrow Y_1Y_2 \ldots Y_{i-1}$ adalah nonterminal dan FIRST (Y_j) berisi ϵ untuk j =1,2,..., i-1 (misal $Y_1Y_2 \ldots Y_{i-1} \rightarrow \epsilon$) tambahkan setiap non - ϵ simbol dalam FIRST (Y_i) ke FIRST (X). Jika ϵ dalam FIRST (Y_j) untuk semua j = 1,2,...,k, maka tambahkan ϵ dalam FIRST (Y_j) ke FIRST (X).

Sekarang kita hitung FIRST untuk setiap string X_1 X_2 ... X_n berikut. Tambahkan FIRST (X_1 X_2 ... X_n) semua non ϵ simbol dari FIRST (X_1). Juga tambahkan non ϵ simbol dari FIRST (X_2) jika ϵ berada dalam FIRST (X_1), non ϵ simbol dari FIRST (X_3) jika ϵ berada dalam FIRST (X_1) dan FIRST (X_2), dan seterusnya. Pada akhirnya tambahkan ϵ ke FIRST (X_1 X_2 ... X_n) jika untuk semua i. FIRST (X_i) berisi ϵ, atau jika $n = 0$.

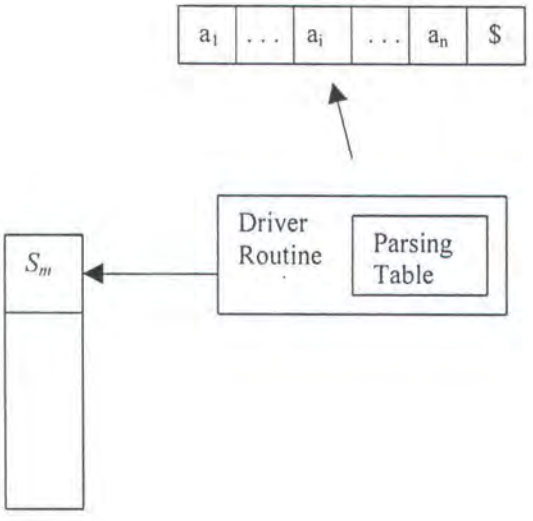
Untuk menghitung FOLLOW (A) untuk semua nonterminal A, tambahkan aturan sampai tidak ada yang ditambahkan dalam himpunan FOLLOW.

1. $ berada dalam FOLLOW (S), jika S adalah start simbol.

2. Jika ada production $A \rightarrow \alpha B \beta, \beta \neq \epsilon$, maka setiap FIRST ($\beta$) adalah FOLLOW ($B$) kecuali ϵ.
3. Untuk Production $A \rightarrow \alpha \beta$ atau $A \rightarrow \alpha B \beta$, ϵ adalah elemen FIRST (β) maka FOLLOW (A) adalah dalam FOLLOW (B).

2.4 LR PARSER

Gambar dibawah ini melukiskan sebuah LR Parser. Parser mempunyai input, stack dan parsing table. Input dibaca dari kiri ke kanan satu simbol satu waktu. Stack berisi string dengan bentuk $s_0X_1s_1X_2 \ldots X_ms_m$, dimana s_m adalah isi stack paling atas. X adalah grammar simbol dan masing-masing s adalah sebuah simbol yang disebut *state*. Masing-masing state simbol mengandung informasi untuk stack dibawahnya dan digunakan sebagai penunjuk keputusan *shift-reduce*. Dalam implementasinya grammar simbol tidak memerlukan stack. Kita memasukkan grammar simbol hanya untuk membantu menjelaskan jalannya LR parser. Parsing table berisi dua bagian yaitu fungsi ACTION dan fungsi GOTO.



Gambar 2.2 LR Parser

Program driving LR parser diatas mempunyai maksud berikut. Program ini menentukan S_m, state saat ini yang berada di atas stack dan a_i, input simbol saat ini. Program ini kemudian meminta ACTION $[S_m, a_i]$, parsing action table memasukkan state S_m dan input a_i. Masukan ACTION $[S_m, a_i]$ mempunyai salah satu dari nilai berikut :

1. shift s
2. reduce $A \rightarrow \beta$
3. accept
4. error

Fungsi GOTO mengambil sebuah state dari grammar simbol sebagai argumen dan menghasilkan sebuah state. Sesungguhnya tabel transisi deterministic finite automaton input simbolnya berupa terminal dan nonterminal dari grammar.

Konfigurasi LR parser adalah gabungan antara isi stack yang merupakan komponen pertama dengan komponen kedua adalah input keluaran berikut :

$(s_0 X_1 s_1 X_2 s_2 \ldots X_m s_m, a_i a_{i+1} \ldots a_n \$)$

Perubahan parser berikutnya diputuskan dengan membaca a_i , input simbol sekarang dan s_m , state di stack teratas dan action parsing table ACTION $[s_m, a]$. Konfigurasi ini menghasilkan empat jenis perubahan berikut :

1. Jika ACTION $[s_m, a_i]$ = shift s, parser melakukan shift, dan memasukkan konfigurasi

   $(s_0 X_1 s_1 X_2 s_2 \ldots X_m s_m a_i s, a_{i+1} \ldots a_n \$)$

   Disini parser sekarang tershift pada input simbol a_i dan state berikutnya s = GOTO$[s_m, a_i]$ ke stack; a_i+1 menjadi input simbol berikutnya.

2. Jika ACTION $[s_m, a_i]$ = reduce $A \rightarrow \beta$, maka parser mengerjakan perintah reduce dan memasukkan konfigurasi $(s_0 X_1 s_1 X_2 s_2 \ldots X_m s_m a_i s, a_{i+1} \ldots a_n \$)$ dimana s = GOTO$[s_{m-r}, A]$ dan r adalah panjang β, sisi kanan dari production. Disini parser pertama diambil sebanyak $2r$ simbol dari stack (r state simbol dan r grammar simbol), ditampilkan state s_{m-r}. Parser dipush dengan A, sisi kiri dari production dan s, entry untuk ACTION $[s_{m-r}, A]$, dimasukkan ke stack. Input simbol belum digerakkan. Untuk LR Parser kita boleh membangun $X_{m-r+1} \ldots Xm$, rangkaian grammar simbol dikeluarkan dari stack, sesuai dengan β, sisi kanan dari reducing production.

3. Jika ACTION$[s_m, a_i]$ = accept, parsing telah selesai.

4. Jika ACTION $[s_m, a_i]$ = error, terjadi kesalahan dan memanggil rutin error.

Algoritma LR parsing sangat sederhana. Mula-mula LR parser dalam konfigurasi $(s_0, a_1 a_2 \ldots a_n \$)$ dimana s adalah state awal yang telah ditandai dan $a_1 a_2 \ldots a_n$ adalah

string yang akan diparsing. Kemudian parser meng-execusi sampai diterima atau terjadi kesalahan. Semua parser prosesnya adalah seperti itu. Perbedaan antara satu LR parser dengan LR Parser lainnya terletak pada informasi action dan goto pada tabel parsing.

2.5 KONSTRUKSI LR(0) PARSING TABLE

Algoritma:

1. Inisialisasi:

State 0 adalah state awal. Biasanya diisi suatu item LR(0) yang terdiri dari start production dan dan sebuah dot yang terletak paling kiri pada bagian kanan. Himpunan awal dari item-item pada state ini (dan state yang lainnya) disebut item kernel atau seed. Himpunan kernel item untuk state awal sebagai berikut:

0
s → .e

2. Tutup item kernel.

Item kernel pada state-state selanjutnya dibuat dengan sebuah operasi closure, yang menghasilkan sebuah himpunan item-item closure. Closure dilakukan pada item-item kernel tersebut yang memiliki suatu simbol nonterminal di sebelah kanan sebuah dot. Tambahkan pada himpunan item closure untuk produk-produk, yang mana ketika direduksi, dapat kembali kepada state yang bersangkutan.

i. Inisialisasi proses closure dengan menambahkan pada himpunan closure semua produksi yang memiliki simbol closure pada bagian kiri. Perlu diingat, jika dot berada pada bagian kiri suatu nonterminal, maka ada perpindahan keluar dari state tsb (yang berlabel dengan nonterminal itu), dan perpindahan tersebut menggambarkan bagian push pada suatu reduksi. Maka, produksi-produksi dengan nonterminal itu pada bagian kiri dapat mengembalikan pada state yang bersangkutan pada waktu suatu reduksi dengan produksi tersebut dilakukan.

Item baru, dibentuk dengan menempatkan sebuah dot di sebelah kiri pada bagian kanan produksi. Pada grammer, karena e berada di sebelah kanan dari dot pada semua item-item kernel, item-item yang berisi semua produksi dengan suatu e pada bagian kiri ditambahkan. Himpunan closure selanjutnya tampak sebagai berikut:

0
s → .e
e → .e + t
e → .t

ii. Ulangi proses closure pada item-item baru yg ditambahkan. Jika suatu item baru yg ditambahkan memiliki suatu nonterminal di sebelah kanan dari dot, dan closure belum dilakukan pada nonterminal itu, tambahkan pada himpunan closure semua produksi yang memiliki nonterminal tersebut sebagai bagian kiri mereka. Pada

contoh, produksi kedua memiliki sebuah t pada bagian kanan dot, maka kita tambahkan semua produksi dengan t pada bagian kiri, sebagai berikut:

0
s → .e
e → .e + t
e → .t
t → .t * f
t → .f

iii. Ulangi proses (b) sampai tidak ada item yang bisa ditambahkan. Suatu produksi dengan sebuah f pada bagian kanan sebuah dot ditambahkan pada langkah sebelumnya, maka himpunan closure ditambah dengan semua produksi yang memiliki f pada bagian kiri, sebagaimana berikut:

0
s → .e
e → .e +t
t → .t * f
t → .f
f → .(e)
f → .NUM

Karena tidak ada produk baru yang memiliki dot pada bagian kiri nonterminal, prosedur ini selesai.

3. Susun item kernel untuk state selanjutnya

i. Pemisahan item-item.

Kelompokkan semua item-item pada current state (baik item kernel dan closure) yang memiliki simbol yang sama di sebelah kanan dot. Kita akan menyebut simbol ini sebagai simbol transisi, dan semua item yang memiliki sebuah simbol transisi yang membentuk partisi. Gambar di bawah ini menunjukkan item-item yang dipisahkan oleh simbol di sebelah kanan dot.

0
s → .e
e → .e + t
e → .t
t → .t * f
t → .f
f → .(e)
f → .NUM

Partisi-partisi yang memiliki dot pada bagian paling kanan produk tidak membentuk sebuah state partisi tersebut menggambarkan reduksi dari state yang sedang

berlangsung. (Paling tidak, ada satu dari partisi-partisi ini dan semua produk e adalah bagian dari partisi ini). Semua partisi lain membentuk kernel dari state baru.

ii. Bentuk item kernel untuk state selanjutnya. Modifikasi item-item dalam partisi tersebut yang membentuk state baru dengan memindahkan dot satu karakter ke kanan, dan:

iii. Tambahkan transisi next-state: Jika sebuah state sudah ada yang memiliki himpunan kernel items yang sama sebagai partisi yang sudah dimodifikasi, tambahkan sebuah garis yang berlabel dengan simbol transisi yang berasal dari state yang sedang berlangsung ke state yang tersedia. Selain itu, tambahkan sebuah state baru, menggunakan kelompok item yang termodifikasi sebagai item-item kernel. Tambahkan sebuah garis dari state yang sedang berlangsung ke simbol transisi yang baru.

4. Ulangi langkah (2) dan (3) pada item-item kernel di setiap state baru sampai tidak ada lagi state yang bisa ditambahkan ke tabel.

BAB III

SEKILAS TENTANG ISIM, FI'IL DAN HURUF

BAB III

SEKILAS TENTANG ISIM, FI'IL DAN HURUF

Di dalam bab ini dibahas mengenai kata-kata yang ada dalam bahasa Arab. Dalam bahasa Arab kata dikenal dengan istilah kalam, di mana kalam memiliki arti lafazh yang tersusun dan bermakna lengkap. Kata (kalam) dalam bahasa Arab terdiri dari tiga bagian, yaitu:

1. **Kata benda (isim)**
2. **Kata kerja (fi'il)**
3. **Huruf**

Pembahasan dalam bab ini lebih ditekankan pada pembahasan mengenai kata kerja (fi'il), karena perubahan bentuk dan penambahan huruf paling banyak terjadi pada jenis kata kerja. Ilmu yang mempelajari mengenai perubahan-perubahan kata biasa dikenal dengan istilah Ilmu Sharaf.

3.1 ISIM (KATA BENDA)

Isim (kata benda) adalah kata yang menunjukkan arti benda atau yang dibendakan. Perubahan susunan dan penambahan huruf tidak banyak terdapat dalam jenis kata benda (isim). Perubahan-perubahan ini hanya terjadi berkisar pada makna jumlah, yaitu untuk mendapatkan makna jumlah benda yang berbeda-beda. Berdasarkan makna jumlah, kata benda terdiri dari:

i. **Kata benda tunggal (isim mufrod)**, yaitu kata benda yang menunjukkan arti tunggal contoh:

- مُسْلِمَةٌ (seorang wanita muslim)

- مُسْلِمٌ (seorang lelaki muslim)

ii. **Kata yang menunjukkan arti dua (isim tatsniyah)**, yaitu kata benda yang menunjukkan dua benda. Dalam keadaan rafa' ditandai dengan penambahan huruf alif di akhir kata. Contoh:

- مُسْلِمَتَانِ (dua orang wanita muslim)

- مُسْلِمَانِ (dua orang lelaki muslim)

iii. **Kata benda jamak**, yaitu kata benda yang menunjukkan jumlah tiga atau lebih. Kata benda jama' ada tiga macam:

i. *Jama' muannats salim*, yaitu kata benda jama' yang tidak berubah dari bentuk mufrodnya, menunjukkan jenis perempuan. Ditandai dengan penambahan huruf alif dan ta, contoh:

مُسْلِمَاتٌ (wanita wanita muslim) bentuk mufrodnya:

ii. *Jama' mudzakar salim*, yaitu kata benda jama' yang tidak berubah dari bentuk mufrodnya, menunjukkan jenis laki-laki. Ditandai dengan penambahan huruf wawu jama'. Contoh:

مُسْلِمُوْنَ (beberapa orang laki-laki muslim) bentuk mufrodnya مُسْلِمٌ

iii. *Jama' taksir*, kata benda jama' yang susunannya berubah dari bentuk mufrodnya. Untuk kata benda jenis ini sifatnya sima'i sehingga tidak bisa dikiaskan dan harus dihapalkan. Contoh:

كُتُبٌ (beberapa kitab) bentuk mufrodnya كِتَابٌ

مَسَاجِدٌ (beberapa masjid) bentuk mufrodnya مَسْجِدٌ

Jenis isim yang lain adalah:

1. ***Ma'rifat*** adalah lafazh yang menunjukkan benda tertentu misal lafazh Zaid, menunjukkan orang yang bernama Zaid.

   Isim ma'rifat ada lima macam, yaitu:

   i. *Isim dhamir (kata ganti)*, isim dhamir dibagi menjadi dua macam yaitu:

   أ. *Dhomir Muttasil*, yaitu dhommir yang bersambung di tengah-tengah ataupun di akhir kata, meliputi:

   نَا يَ، كُنَّ، كُمَا، كِ، كُمْ، كُمَا، كَ، هُنَّ، هُمَا، هَ، ، هُمْ هُمَا، هُ،

   ب. *Dhomir munfashil*, yaitu dhomir yang penulisannya selalu dipisah dan ditulis di awal kata, meliputi:

   نَحْنُ اَنَا، اَنْتُنَّ، اَنْتُمَ، اَنْتِ، اَنْتُمْ، اَنْتُمَا، اَنْتَ، هُنَّ، هُمَا، هِيَ، هُمْ، هُمَا، هُوَ،

   ii. *Isim 'alam (nama)*, seperti lafazh: زَيْدٌ (Zaid-nama orang), مَكَّةٌ (Mekkah-nama kota)

iii. *Isim Mubham* yaitu isim yang tidak diketahui dengan pasti, isim mubham ini mencakup isim isyarah (kata penunjuk) yaitu isim yang mengisyaratkan sesuatu dan isim maushul (kata ganti penghubung). Contoh: هذَا، هذِهِ، الَّذِيْنَ الَّذِى

iv. *Isim yang diberi alif lam (isim ma'rifah)*, untuk menunjukkan benda tertentu seperti lafazh:

اَلكِتَابُ artinya kitab itu

الرَّجَالُ artinya seorang lelaki itu

v. *Isim yang diidhofatkan (disandarkan)* kepada salah satu di antara yang empat bagian (yaitu isim mudhmar (dhamir), isim 'alam, isim mubham dan isim ma'rifah). Contoh:

كِتَـابُكَ (kitabmu) asalnya كِتَـابٌ (kitab) dan اَنْتَ (kamu) digabungkan menjadi satu. اَنْتَ berubah menjadi كَ karena posisinya di akhir kata.

2. *Isim Nakiroh*

Yaitu isim yang jenisnya bersifat umum yang tidak menentukan sesuatu perkara dan lainnya, setiap isim yang layak dimasuki alif dan lam. Contoh:

رَجُلٌ (seorang laki-laki → secara umum)

كِتَابٌ (sebuah kitab → secara umum)

3.2 FI'IL (KATA KERJA)

Kata kerja (fi'il) merupakan kata yang menunjukkan suatu pekerjaan dan disertai dengan makna waktu. Fi'il dalam bahasa Arab kaya akan bentuk-bentuk turunan, yang merupakan perubahan atau pengembangan arti makna dari yang dikandung oleh bentuk asal / dasar, dalam hal ini fi'il mujarrad (kata kerja dasar).

Kata kerja dasar dalam bahasa Arab ada dua macam, yang pertama jumlahnya terdiri atas tiga huruf dan yang kedua terdiri atas empat huruf. Kebanyakan kata kerja dalam bahasa Arab terdiri atas tiga huruf yang merupakan akarnya, sedangkan kata kerja yang akarnya terdiri atas empat huruf jumlahnya sedikit dan jarang digunakan. Sedangkan apabila ditinjau dari jenisnya, kata kerja terdiri dari tiga macam yaitu kata kerja bentuk lampau (fi'il madhi), kata kerja bentuk sekarang./akan datang (fi'il mudhari) dan kata kerja bentuk perintah (fi'il amr).

Berikut ini akan dijelaskan mengenai pengembangan makna dari fi;il madhi. Kemudian akan dibahas tentang pengembangan dari bentuk fi'il mujarrad.

i. kata kerja bentuk lampau (fi'il madhi) contoh:

(telah menulis) كَتَبَ

(telah menolong) نَصَرَ

ii. Kata kerja bentuk sedang / akan datang (fi'il mudhori'), contoh:

(sedang / akan menulis) يَكْتُبُ

(sedang / akan menolong) يَنْصُرُ

iii. Kata kerja bentuk perintah (fi'il amr). Contoh:

(tulislah) اُكْتُبْ

(tolonglah) اُنْصُرْ

Fi'il mudhari dan fi`il amr merupakan turunan dari fi'il madhi, turunan-turunan yang lain adalah mashdar, isim fail, isim maful, fi'il nahi, isim zaman, isim makaan, isim alat. Bentuk perubahan dari fi'il madhi menjadi kata-kata yang lain secara keseluruhan seperti dalam contoh di bawah ini:

1. نَصَرَ: fi'il madhi, artinya sudah menolong.
2. يَنْصُرُ: fi'il mudhorik, artinya sedang / akan menolong
3. نَصْرًا: mashdar ghoiru mim, artinya pertolongan
4. مَنْصَرًا: mashdar mim, artinya pertolongan
5. نَاصِرٌ: isim fail, artinya yang menolong
6. مَنْصُوْرٌ: isim maf'ul artinya yang ditolong.
7. اُنْصُرْ: fi'il amr, artinya tolonglah
8. لَاتَنْصُرْ: fi'il nahi, artinya jangan ditolong

9. مَنْصَر: isim zaman / makan, artinya tempat / waktu menolong

10. مِنْصَر: isim alat, artinya alat untuk menolong.

Perubahan fi'il yang lain berupa pemberian kata ganti (*dhamir muttasil*) pada akhiran fi'il yang menerangkan bilangan dan jenis, biasa dikenal dengan istilah *mabni ma'lum*, sedangkan apabila berupa kata kerja pasif disebut mabni majhul. Berikut ini fi'il mabni ma'lum yang terdapat pada fi'il madhi, mudhari' amr dan nahi yang jumlahnya ada empat belas macam.

Untuk fi'il amr dan nahi susunan orang pertama tunggal dan jama' tidak ada karena posisinya sebagai pemberi perintah / larangan

1. نَصَرَ artinya **dia seorang lelaki** telah menolong

   يَنْصُرُ artinya dia seorang lelaki sedang / akan menolong

   لِيَنْصُرَ artinya dia seorang lelaki supaya menolong

   لَا يَنْصُرْ artinya dia seorang lelaki jangan menolong

2. نَصَرَا artinya **mereka dua lelaki** telah menolong

   يَنْصُرَانِ artinya mereka dua lelaki sedang/akan menolong.

   لِيَنْصُرَا artinya mereka dua lelaki supaya menolong

لَايَنْصُرَا artinya mereka dua lelaki jangan menolong

3. نَصَرُوا artinya **mereka (lelaki)** telah menolong

يَنْصُرُوْنَ artinya mereka (lelaki)sedang/akan menolong

لِيَنْصُرُوا artinya mereka (lelaki)supaya menolong

لَايَنْصُرُوا artinya mereka (lelaki)jangan menolong

4. نَصَرَتْ artinya **dia seorang perempuan** telah menolong

تَنْصُرُ artinya dia seorang perempuan sedang/akan menolong.

لِتَنْصُرْ artinya dia seorang perempuan supaya menolong

لَاتَنْصُرْ artinya dia seorang perempuan jangan menolong

5. نَصَرَتَا artinya **dua orang perempuan** telah menolong

تَنْصُرَانِ artinya dua orang perempuan sedang/akan menolong.

لِتَنْصُرَا artinya dua orang perempuan supaya menolong

لَاتَنْصُرَا artinya dua orang perempuan jangan menolong

6. نَصَرْنَ artinya **mereka (perempuan)** telah menolong

يَنْصُرْنَ artinya mereka (perempuan) sedang/akan menolong.

لِيَنْصُرْنَ artinya mereka (perempuan) supaya menolong

لَايَنْصُرْنَ artinya mereka (perempuan) jangan menolong

7. نَصَرْتَ artinya kamu (lelaki) telah menolong

تَنْصُرُ sedang/akan menolong.

اُنْصُرْ supaya menolong

لَاتَنْصُرْ jangan menolong

8. نَصَرْتُمَا artinya kamu (dua lelaki) telah menolong

تَنْصُرَانِ sedang/akan menolong.

اُنْصُرَا supaya menolong

لَاتَنْصُرَا jangan menolong

9. نَصَرْتُمْ artinya kalian (3 lelaki/lebih) telah menolong

تَنْصُرُوْنَ sedang/akan menolong.

اُنْصُرُوْا supaya menolong

لاَتَنْصُرُوْا jangan menolong

10. نَصَرْتِ artinya kamu (perempuan) telah menolong

تَنْصُرِيْنَ sedang/akan menolong.

اُنْصُرِى supaya menolong

لاَتَنْصُرِى jangan menolong

11. نَصَرْتُمَا artinya kamu (dua perempuan) telah menolong

تَنْصُرَانِ sedang/akan menolong.

اُنْصُرَا supaya menolong

لاَتَنْصُرَا jangan menolong

12. نَصَرْتُنَّ artinya kalian (tiga/lebih perempuan) telah menolong

تَنْصُرْنَ sedang/akan menolong.

اُنْصُرْنَ supaya menolong

لاَتَنْصُرْنَ jangan menolong

13. نَصَرْتُ artinya saya (lelaki/perempuan) telah menolong

أَنْصُرُ sedang/akan menolong.

14. نَصَرْنَا artinya kami telah menolong

نَنْصُرُ sedang/akan menolong.

Untuk menyatakan pola kata kerja, ahli tata bahasa menggunakan konsonan kata kerja فَعَلَ (fa'ala) dan فَعْلَـلَ (fa'lala). Fa'ala digunakan untuk fi'il yang jumlah huruf asalnya tiga huruf (tsulatsy), dan fa'lala digunakan untuk fi'il yang jumlah huruf asalnya ada empat huruf (ruba'i). Untuk fi'il tsulatsy, huruf ف (fa') menggambarkan akar pertama biasa disebut dengan fa' fi'il. Huruf ع ('ain) akar kedua atau biasa di sebut 'ain fi'il dan ل (lam) akar ketiga atau lam fi'il. Sedangkan untuk ruba'I huruf lam kedua sebagai akar keempat atau lam fi'il kedua. Pola-pola dasar dari kata kerja ini dikenal dengan istilah wazan. Huruf-huruf tambahan yang memasuki fi'il mujarrod ada sepuluh macam,

yaitu: ء س، ل، ت، م، و، ن، ي، ه، ا

Berikut ini pembagian fi'il berdasarkan jumlah huruf asalnya:

1) Fi'il Tsulatsy, fi'il yang tersusun dari tiga huruf. Fi'il ini terbagi lagi menjadi dua bagian yaitu:

   i) Fi'il Tsulatsy Mujarrad, yaitu fi'il yang terdiri dari tiga huruf dan di dalamnya tidak didapatkan huruf tambahan, contoh: كَسَرَ وَعَدَ، كَرَمَ، قَتَلَ، نَصَرَ،

   ii) Fi'il tsulatsy Mazied, yaitu fi'il yang huruf asalnya terdiri dari tiga huruf. Yang mendapatkan tambahan satu huruf disebut Fi'il Tsulatsy Mazied Ruba'i. Yang mendapatkan tambahan dua huruf disebut Fi'il Tsulatsy Mazied Khumasy. Dan yang mendapatkan tiga huruf disebut Fi'il Tsulatry Mazied Sudasy. Contoh:

   أ) Tsulatsy Mazied Ruba'I:

      (1) قَاتَلَ asalnya قَتَلَ

      (2) أَكْرَمَ asalnya كَرَمَ

      (3) أَوْعَدَ asalnya وَعَدَ

   ب) Tsulatsy Mazied Khumasy

      (1) اِنْكَسَرَ asalnya كَسَرَ

      (2) تَوَاعَدَ asalnya وَعَدَ

      (3) تَكَسَّرَ asalnya كَسَرَ

   ج) Tsulatsy Mazied Sudasy

إِسْتَغْفَرَ asalnya غَفَرَ

2) Fi'il Ruba'i, yaitu fi'il yang huruf asalnya tersusun dari empat huruf. Fi'il ini terbagi lagi menjadi:

i) Fi'il Ruba'i Mujarrad, yaitu fi'il yang terdiri dari empat huruf dan tidak didapatkan huruf tambahan, contoh: تَرْجَمَ دَحْرَجَ،

ii) Fi'il Ruba'i Mazied, yaitu fi'il yang huruf asalnya terdiri dari empat huruf dan mendapatkan tambahan satu huruf disebut Fi'il Ruba'i Mazied Khumasy dan yang mendapatkan tambahan dua huruf disebut Fi'il Ruba'I Mazied Sudasy.

أ) Fi'il Ruba'i Mazied Khumasy تَدَحْرَجَ

ب) Fi'il Ruba'i Mazied Sudasy اِحْرَنْجَمَ

iii) Fi'il Ruba'i Mulhaq, yaitu fi'il yang asalnya berasal dari Tsulatsy Mujarrad akan tetapi diikutsertakan kepada Ruba'i Mujarrad. Fi'il yang mendapatkan tambahan satu huruf disebut Fi'il Ruba'i Mulhaq Mazied Khumasy. Contoh:

تَجَحْوَرَ تَحَلْبَبَ، جَهْوَرَ، جَلْبَبَ،

BINA (BANGUNAN) FI'IL

Bina Fi'il ada tujuh macam, yaitu:

1) Fi'il Bina Shoheh.

Tiap-tiap fi'il yang fa' fi'il, ain fi'il dan lam fi'il tidak berupa huruf hamzah dan huruf illat (wawu, alif dan ya') serta 'ain dan lam fi'ilnya tidak terdiri dari huruf sejenis. Contoh: ضَرَبَ نَصَرَ،

2) Fi'il Bina' Ajwaf.

Tiap-tiap fi'il yang 'ain fi'ilnya berupa huruf illat. Kalau huruf illatnya berupa wawu maka disebut Bina' Ajwaf Wawi. Jika berupa huruf ya' maka disebut Bina Ajwaf Ya'i. Contoh:

أ) صَانَ asalnya صَوَنَ , 'ain fi'il berupa huruf wawu

ب) سَارَ asalnya سَيَرَ , 'ain fi'il berupa huruf ya'

3) Fi'il Bina Mahmuz

Tiap-tiap fi'il yang fa', 'ain atau lam fi'ilnya berupa hamzah. Kalah fa' fi'ilnya berupa huruf hamzah disebut Bina Mahmuz Fa'. Kalau 'ain fi'il nya yang berupa huruf hamzah disebut Mahmuz 'Ain. Dan kalau yang berupa huruf hamzah lam fi'il nya disebut Mahmuz Lam. Contoh:

أ) أَكَلَ mahmuz fa', fa' fi'il nya berupa huruf hamzah

ب) رَأَي mahmuz 'ain, 'ain fi'il nya berupa huruf hamzah

ج) نَشَأَ mahmuz lam, lam fi'il nya berupa huruf hamzah

4) Fi'il Bina Mudlo'af, dibagi menjadi dua bagian yaitu:

i) Fi'il Bina Mudlo'af Tsulatsy ialah tiap-tiap fi'il yang 'ain dan lam fi'il terdiri dari huruf yang sejenis. Contoh:

(1) مَدَّ asalnya مَدَدَ

(2) فَرَّ asalnya فَرَرَ

ii) Fi'il Bina Mudlo'af Ruba'i, ialah tiap-tiap fi'il yang fa' fi'il dan lam fi'il yang pertama terdiri dari huruf yang sejenis. 'Ain fi'il dan lam fi'il yang kedua terdiri dari huruf yang sejenis juga. Contoh:

قَلْقَلَ وَسْوَسَ،

5) Fi'il Bina' Naqis

Tiap-tiap fi'il yang lam fi'il nya berupa huruf illat. Kalau berupa huruf illat wawu disebut Fi'il Bina' Naqis Wawi, kalau huruf illat ya' disebut Fi'il Bina Naqis Ya'i. Contoh:

أ) غَزَا naqis wawi. Asalnya غَزَوَ, lam fi'il nya berupa huruf wawu

ب) سَرَى naqis ya'i . Asalnya سَرَيَ , lam fi'il nya berupa huruf ya'

6) Fi'il Bina' Lafif, dibagi menjadi dua bagian yaitu:

i) Fi'il Bina Lafif Maqrun, ialah tiap-tiap fi'il yang 'ain dan lam fi'il nya terdiri dari huruf illat. Contoh: قَـــوِيَ ، شَوَى 'ain fi'il berupa huruf wawu dan lam fi'il berupa huruf ya'.

ii) Fi'il Bina' Lafif Mafruq ialah tiap-tiap fi'il yang fa' fi'il dan lam fi'il nya terdiri dari huruf illat. Contoh: وَلَى ، وَقَى lam fi'il berupa huruf wawu dan lam fi'il berupa huruf ya'.

7) Fi'il Mu'tal Misal

Tiap-tiap fi'il yang fa' fi'il nya berupa huruf illat. kalau berupa huruf illat wawu disebut Fi'il Bina' Mu'tal Misal Wawi dan kalau berupa huruf ya' disebut Fi'il Bina' Mu'tal Misal Ya'i. Contoh:

أ) وَعَدَ fa' fi'il berupa huruf illat wawu

ب) يَسَرَ fa' fi'il berupa huruf illat ya'

Fi'il Tsulatsy

Fi'il yang tersusun dari tiga huruf dikenal dengan istilah Fi'il Tsulatsy. Fi'il Tsulatsy yang tidak mendapatkan huruf tambahan disebut dengan fi'il tsulatsy mujarrad. Pembahasan mengenai pola perubahan (wazan) fi'il tsulatsy mujarrad terbagi menjadi enam pembahasan (enam wazan). Keenam wazan tersebut adalah:

1) Wazan Fa'ala Yaf'ulu فَعَلَ - يَفْعُلُ contoh: كَتَبَ - يَكْتُبُ

2) Wazan Fa'ala Yaf'ilu فَعَلَ - يَفْعِلُ, contoh: ضَرَبَ - يَضْرِبُ

3) Wazan Fa'ala Yaf'alu فَعَلَ - يَفْعَلُ , contoh: فَتَحَ - يَفْتَحُ

4) Wazan fa'ila Yaf'alu فَعِلَ - يَفْعَلُ . contoh: عَلِمَ - يَعْلَمُ

5) Wazan fa'ila Yaf'ilu فَعِلَ - يَفْعِلُ . contoh: حَسِبَ - يَحْسِبُ

6) Wazan fa'ula yaf'ulu فَعُلَ - يَفْعُلُ , contoh: حَسُنَ - يَحْسُنُ

Fi'il Tsulatsy yang mendapatkan tambahan satu huruf disebut dengan fi'il Tsulatsy mazied Ruba'i, terdapat tiga wazan, yaitu:

1) Wazan Fa' 'ala - فَعَّلَ , ditambah 'ain fiilnya.

Makna fi'il wazan fa'ala adalah:

i) Sebagai fi'il muta'adi yaitu fi'il yang menuntut keberadaan maf'ul / pelaku.

ii) Menunjukkan arti jamak

iii) Menisbatkan maf'ul kepada arti fi'il

iv) Meniadakan / memiliki makna lain (kias)

v) Menjadikan fi'il sebagai isim.

Contoh: فَرَّحَ artinya menggembirakan, asalnya فَرِحَ (gembira)

2. Wazan Faa'ala - فَاعَلَ, ditambahkan huruf alif setelah fa' fi'il.

Makna fi'il wazan faa'ala adalah:

أ) Menunjukkan arti saling berbuat

ب) Memiliki makna taksir (banyak)

ج) Memiliki makna kesengajaan

د) Makna tsulatsy mujarrad (tidak berubah)

Contoh: قَاتَلَ (saling membunuh / berperang) asalnya قَتَلَ (membunuh)

3. Wazan Af'ala - اَفْعَلَ ditambahkan huruf hamzah sebelum fa' fi'il.

Makna wazan af'ala adalah:

أ) Sebagai fi'il muta'adi yaitu fi'il yang menuntut keberadaan pelaku / maf'ul.

ب) Menjadi sesuatu

ج) Memiliki makna hilang

د) Menuju tempat tujuan, dalam hal ini huruf asalnya bukan berupa fi'il akan tetapi berupa isim.

ه) Memasuki sesuatu (waktu)

و) Memiliki kesan bersungguh-sungguh

ز) Memiliki makna diambil dari fiilnya.

Contoh: أَكْرَمَ (memuliakan) asalnya كَرَمَ (mulia)

Fi'il Tsulatsy Mazied Khumasy

Fi'il Tsulatsy yang mendapatkan tambahan dua huruf disebut dengan fi'il tsulatsy mazied khumasy, terdapat lima wazan, yaitu:

1. Wazan Infa'ala - اِنْفَعَلَ mendapat tambahan hamzah dan nun

Makna fi'il wazan infa'ala adalah:

(1) Hasil dari suatu perbuatan

(2) Menunjukkan arti pengaruh dari suatu perbuatan

Contoh: اِنْكَسَرَ (menjadi pecah) asalnya كَسَرَ (pecah)

2. Wazan Ifta'ala - اِفْتَعَلَ mendapat tambahan huruf hamzah dan ta'

Makna fi'il wazan ifta'ala adalah:

i) Hasil dari suatu perbuatan

ii) Isim yang dijadikan fi'il

iii) Menunjukkan arti sungguh sungguh

iv) Menunjukkan suatu kesulitan

v) Menunjukkan makna fa'ala

vi) Memiliki makna saling

Contoh: اجتمع (menjadi berkumpul) asalnya جمع (berkumpul)

3. Wazan if'alla افعـــل mendapat tambahan huruf hamzah dan huruf fa' fiilnya diberi tasydid

Makna wazan if'alla adalah:

i) Menunjukkan arti sifat / warna

ii) Menunjukkan arti sangat

Contoh: احمر (sangat merah) asalnya حمر (merah)

4. Wazan tafa' 'ala - تفعل mendapat tambahan huruf ta' dan huruf 'ain fi'il diberi tasydid.

Makna fi'il wazan tafa' 'ala:

i) Memperoleh / mendapatkan beban suatu pekerjaan

ii) Memaksakan sesuatu

iii) Hasil suatu perbuatan

iv) Fa'il menjadikan fi'il sebagai maf'ul

v) Memiliki makna menjadi

vi) Hasil dari suatu perbuatan secara beruntun

vii) Menuntut sesuatu

viii) Isim dihukumi fi'il

ix) Menisbatkan kepada sesuatu

Contoh: تَكَسَّرَ (menjadi pecah) asalnya كَسَرَ (pecah)

5. Wazan Tafaa'ala - تَفَاعَلَ, mendapat tambahan huruf ta' dan alif

Makna wazan Tafaa'ala adalah:

i) menunjukkan makna saling melakukan

ii) Hasil dari suatu perbuatan

iii) Untuk menampakkan apa-apa yang tidak ada / pura-pura

iv) Memiliki makna berangsur-angsur / bertahap

v) Memiliki makna mujarrad (dinisbatkan kepada nama Allah / sifat Allah)

Contoh: تَبَاعَدَ (saling menjauh) asalnya بَعُدَ (jauh)

Fi'il Tsulatsy Mazied Sudasy

Fi'il tsulatsy yang mendapatkan tambahan tiga huruf disebut dengan fi'il tsulatsy mazied sudasy, terdapat enam wazan, yaitu:

1. Wazan Istaf'ala - اِسْتَفْعَلَ mendapat tambahan huruf hamzah, sin dan ta'

Makna wazan istaf'ala:

i) Menunjukkan makna mencari sesuatu / menuntut sesuatu

ii) Memiliki tujuan maf'ul sebagai sifat atau maf'ul itu hendak disifati

iii) Perubahan

iv) Makna terpaksa

v) Sebagai akibat dari suatu perbuatan

Contoh: اِسْتَغْفَرَ (meminta maaf) asalnya غَفَرَ (memaafkan)

2. Wazan If'au'ala - اِفْعَوْعَـلَ mendapat tambahan huruf hamzah, wau dan huruf 'ain fi'il.

Makna wazan If'au'ala:

i) Memiliki makna sangat

ii) Makna mujarrad

Contoh: اِحْقَوْقَفَ - حَقَفَ

3. Wazan If'awwala - اِفْعَوَّل mendapatkan huruf hamzah dan wau yang bertasydid.

Makna wazan if'awwala adalah memiliki arti sangat

Contoh: اِجْلَوَّذَ (lalu lalang dengan cepat) asalnya جَلَذَ

4. Wazan If'aalla - اِفْعَالَّ, mendapat tambahan huruf hamzah, alif dan lam fi'il nya diberi tasydid.

Makna Wazan If'aalla adalah memiliki arti sangat

Contoh: اِحْمَرَّ (menjadi merah) asalnya حَمَرَ (merah)

5. Wazan If'anlaa - اِفْعَنْلَى, mendapat tambahan huruf hamzah, nun dan alif manqus.

Contoh: اِسْلَنْقَى (telentang) asalnya سَلَقَ (merebus)

6. Wazan if'anlala - اِفْعَنْلَلَ, mendapat tambahan huruf hamzah, nun dan lam fi'il kedua.

Contoh: اِقْعَنْسَسَ (terlambat dan mundur kebelakang) قَعَسَ

Fi'il Ruba'i

Fi'il ruba'i merupakan fi'il yang huruf asalnya tersusun dari empat huruf. Fi'il ruba'i yang yang tidak mendapatkan tambahan disebut fi'il ruba'i mujarrad. Yang memiliki hanya satu wazan yaitu wazan fa'lala.

Contoh: دَحْرَجَ (_____)

Fi'il ruba'i mazied khumasy, merupakan fi'il yang asalnya dari empat huruf kemudian ditambah satu huruf sehingga jumlahnya menjadi lima huruf, memiliki satu wazan, yaitu wazan tafa'lala (mendapat tambahan huruf ta'). Makna wazan tafa'lala adalah sebagai akibat dari suatu perbuatan.

Contoh: تَدَحْرَجَ (menjadi terguling) asalnya دَحْرَجَ (menggulingkan)

Fi'il Ruba'i Mazied sudasy, merupakan fi'il ruba'i yang mendapatkan tambahan dua huruf sehingga jumlahnya menjadi enam huruf, memiliki dua wazan, yaitu:

أ) if'anlala (mendapat tambahan huruf hamzah dan nun)

ب) if'alalla (mendapat tambahan huruf hamzah dan huruf lam fi'il kedua bertasydid).

Makna fi'il ruba'I mazied sudasy adalah memiliki arti sangat.

Contoh: اِحْرَنْجَمَ (berdesakan) asalnya حَرَجَمَ (sempit)

Fi'il Ruba'i Mulhaq, merupakan fi'il yang asalnya terdiri dari tiga huruf kemudian ditambah satu huruf disamakan wazannya dengan ruba'i mujarrad. Terdapat enam wazan, yaitu:

أ) Fau'ala

ب) Fa'wala

ج) fai'ala

د) fa'yala

ه) fa'laa

و) fa'lala

Sedangkan fi'il ruba'I mulhaq yang mendapatkan tambahan satu huruf disebut fi'il ruba'I mulhaq mazied khumasy. Fi'il ini diserupakan dengan wazan fi'il ruba'i mazied khumasy. Memiliki enam wazan, yaitu:

أ) Tafau'ala

ب) Tafa'wala

ج) tafai'ala

د) tafa'yala

ه) tafa'lala

3.3 HURUF

Huruf adalah kalam / kata yang tidak dapat berdiri sendiri. Maksudnya huruf merupakan kata yang hanya dapat menunjukkan arti apabila dirangkaikan dengan kata yang lain. Macam-macam huruf ada banyak sekali seperti huruf jarr, huruf athaf, huruf nashab, huruf jazm dan sebagainya.

Berikut ini huruf-huruf Jarr, Nashab, Jazm

Huruf Jar, meliputi: مِنْ ، اِلَى ، عَنْ ، فِى ، عَلَى ، اللاَّمُ ، الكَافَ ، البَاءُ ، رُبَّ ، حَتَّى

Huruf Qosam (sumpah) seperti: وَاوٌ ، تَاءٌ ، بَاءٌ

Huruf Nashab, meliputi: اَنْ ، لَنْ ، لِكَى ، اِذَنْ

Huruf Jazm, meliputi: النَّهِى لاَمَ ، لَمْ ، اَلَمْ ، لَمَّا ، اَلَمَّا ، اْلاَمَّا لاَمُ

Contoh pemakaian huruf:

لِزَيْدٍ اَلْمَالُ= harta milik zaid, huruf lam artinya untuk, karena, milik

كَالْبَدْرِ زَيْدٌ = Zaid itu bagaikan bulan purnama, huruf kaf artinya seperti

وَاللهِ = demi Allah, huruf qosam artinya sumpah.

Huruf sengaja tidak dibahas secara lebih detail karena begitu luasnya pembahasan bahasa arab mengenai huruf, sampai-sampai ada suatu ilmu tersendiri yang khusus membahas tentang huruf. Selain itu huruf juga tidak

mempunyai pola perubahan sehingga bisa diketahui artinya secara langsung. Dalam tugas akhir ini lebih ditekankan kepada perubahan yang terjadi pada suatu kata kerja / fi'il.

BAB IV

MODEL / PERMASALAHAN SECARA DETAIL

BAB IV

MODEL / PERMASALAHAN SECARA DETAIL

Pada bab terdahulu sudah dijelaskan tentang point-point permasalahan, yang mana point-point tersebut dan metodologi pemecahan akan dijelaskan secara detail pada bab ini. Permasalahan-permasalahan tersebut adalah bahwa aplikasi kamus bahasa arab ini diharapkan mampu untuk menangani hal-hal sebagai berikut:

1. Aplikasi diharapkan mampu menambahkan aturan-aturan (grammer) beserta keterangan tanpa harus mengubah source program, karena grammer tersebut disimpan dalam database.
2. Aplikasi diharapkan mampu menambahkan data kamus yang hanya berupa pola dasar (wazn fa'ala), karena data kamus tersebut juga disimpan dalam database.
3. Aplikasi diharapkan mampu membentuk State Machine dan LR(0) Parsing Table dari Database Grammer.
4. Aplikasi diharapkan mampu mendeteksi pola yang dipakai pada kata yang diinputkan user, baik yang sudah mengalami perubahan bentuk ataupun tidak serta mendapatkan bentuk dasarnya.
5. Aplikasi diharapkan mampu memberikan arti dan keterangan pada kata yang diinputkan user setelah diubah menjadi pola dasar jika kata tersebut sudah benar dan terdapat pada kamus.

Masing-masing point di atas akan dijabarkan sebagai berikut:

4.1 DATABASE GRAMMER DAN KAMUS

Di dalam bahasa Arab, suatu kata bisa berubah hingga puluhan bahkan ratusan bentuk. Akan tetapi masing-masing tidak lepas dari kaidah-kaidah atau pola-pola tertentu yang masing-masing bentuk memiliki pola dasar tersendiri. Di dalam aplikasi ini, pola-pola tersebut perlu disimpan dalam database, sehingga mampu untuk dikembangkan dan dimodifikasi untuk menghasilkan hasil yang lebih akurat.

Pola-pola tadi diterjemahkan menjadi suatu production rules, yang tentu saja hanya bisa dilakukan oleh seorang yang pakar dalam tehnik kompilasi dan Bahasa Arab. Dengan alasan tersebut, di dalam aplikasi ini, belum semua pola dimasukkan dalam database, akan tetapi diharapkan mampu untuk dikembangkan lebih lanjut dengan penambahan data pada database rule tersebut.

Adapun teknik penyimpanan rule adalah sebagai berikut:
Masing-masing huruf hijaiyyah, harakat, fa' fi'il, 'ain fi'il, lam fi'il, dan sebagainya dikodekan menjadi suatu simbol. Sehingga gabungan dari pola akan membentuk suatu *right production* pada suatu grammer.

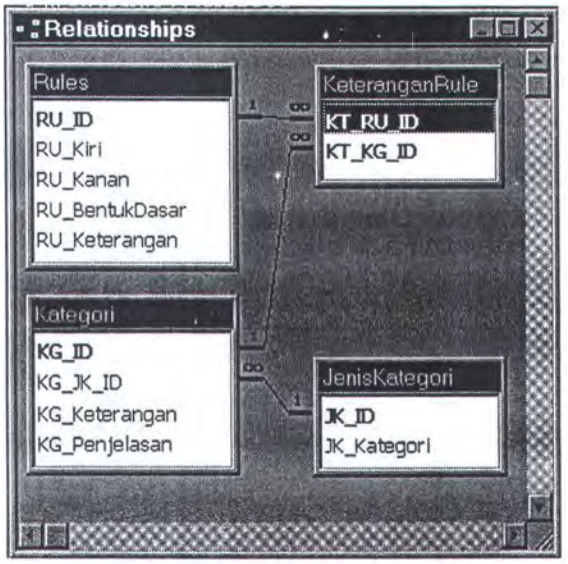
Keterangan tentang mapping kode dari huruf/harakat ke symbol, bisa dilihat pada Appendix 1.

Misalkan untuk menyimpan pola dari wazn fa'ala-yaf'ulu digunakan grammer sebagai berikut:

No	Pola Wazn	Keterangan	Grammer
1	فَعَلَ	fiil madhi	K → akbkck
2	يَفْعُلُ	fiil mudhorik	K → jkaqbmcm
3	فَعْلًا	mashdar	K → akbqcle
4	مَفْعَلًا	mashdar mim	K → hkaqbkcle
5	فَاعِلٌ	fa'il (pelaku)	K → akebocn
6	مَفْعُوْلٌ	maf'ul	K→ hkaqbmiqcn
7	أُفْعُلْ	fi'il amr	K → emaqbmcq
8	لَاتَفْعُلْ	fi'il nahi	K → gkefkaqbmcq
9	مَفْعَلٌ	isim makan / zaman	K → hkaqbkcn
10	مِفْعَلٌ	isim alat	K → hoaqbkcn

Tabel 4.1 Production Rules untuk Wazn Fa'ala Yaf'ulu

Adapun struktur database yang dipergunakan untuk menyimpan aturan-aturan pola grammer adalah sebagai berikut:



Gambar 4.1 Struktur Database

Tabel RULES

digunakan untuk menyimpan data pola-pola / aturan bahasa arab dalam bentuk production rule.

Field	tipe	Keterangan
RU_ID	long integer	kode rules (autoincrement)
RU_Kiri	text[1]	bagian kiri dari production
RU_Kanan	text[30]	bagian kanan production
RU_BentukDasar	text[30]	bentuk pola dasar dari production yang akan dicocokkan dengan kamus
RU_Keterangan	memo	Keterangan khusus dari rule

Tabel 4.2 Struktur Tabel Rules

Tabel KETERANGANRULE

merupakan tabel transaksi 1-N untuk menyimpan keterangan untuk setiap rule.

Field	tipe	Keterangan
KT_RU_ID	long integer	foreign key dari Tabel Rule
KT_KG_ID	long integer	foreign key dari Tabel Kategori

Tabel 4.3 Struktur Tabel KeteranganRule

Tabel KATEGORI

digunakan untuk menyimpan data kategori yang merupakan definisi umum pada klasifikasi sebuah kata dalam bahasa Arab.

Field	tipe	Keterangan
KG_ID	long integer	Identifier kategori (autoincrement)
KG_JK_ID	byte	ID jenis kategori, foreign key dari tabel jenis kategori
KG_Keterangan	Text[30]	Keterangan /judul istilah umum yang dikenal untuk klasifikasi
KG_Penjelasan	Memo	Penjelasan dari klasifikasi

Tabel 4.4 Struktur Tabel Kategori

Tabel JENISKATEGORI

digunakan untuk menyimpan data pengelompokan pada masing-masing klasifikasi pada sebuah kata dalam bahasa Arab.

Field	Type	Keterangan
JK_ID	Byte	identifier Jenis Kategori
JK_Kategori	Text[30]	Jenis kategori

Tabel 4.5 Struktur Tabel JenisKategori

DATABASE KAMUS

Sebagaimana telah dijelaskan sebelumnya, di dalam aplikasi Kamus Bahasa Arab ini, data kamus yang disimpan cukup hanya pola dalam bentuk dasar saja untuk kata yang merupakan perubahan bentuk dari fi'il. Adapun untuk kata-kata selain yang dibentuk dari fi'il, hanya didata dalam bentuk asalnya, yakni bentuk tunggal (mufrod) dan umum (nakiroh).

Struktur tabel untuk mencatat kamus adalah sebagai berikut:

Tabel KAMUS

Field	Type	Keterangan
KA_ID	long integer	identitas
KA_Arab	varchar[30]	kosakata bahasa arab
KA_Indonesia	varchar[50]	makna dalam bahasa Indonesia

Tabel 4.6 Struktur Tabel Kamus

4.2 KONSTRUKSI STATE MACHINE DAN LR(0) PARSING TABLE

Sebagaimana dijelaskan sebelumnya, untuk membentuk suatu Parser Driver, diperlukan suatu machine untuk menyusun database rule tadi sehingga membentuk DFA

State Machine yang kemudian akan dibentuk menjadi LR(0) Parsing Table. Pada bagian ini akan dijelaskan mengapa memilih pemakaian metode LR(0) dan bagaimana contoh penerapannya. Adapun penjelasan algoritma pembentukan Parsing Tabel LR(0) ini telah dijelaskan pada bab Dasar Teori.

Mengapa Menggunakan Parsing Tabel LR(0) ?

Dalam kompilasi, banyak dikenal metode untuk melakukan parsing. Salah satunya adalah LR(0). LR(0) dikenal merupakan metode yang paling mudah untuk membuat parsing tabel. Namun di sisi lain, memiliki keterbatasan, yaitu pada kasus adanya konflik antara aksi reduce dengan shift.

Misalnya contoh sederhana keterbatasan LR(0) adalas seperti pada grammer di bawah ini:

$X \rightarrow (X)$

$X \rightarrow \in$

Parsing LR(0) tabel dari grammer di atas adalah sebagai berikut:

	ACTION			GOTO
State	**(**	**)**	**$**	**X**
0	s2/r3	r3	r3	1
1	-	-	acc	-
2	s2/r3	r3	r3	3
3	-	s4	-	-
4	r2	r2	r2	-

Tabel 4.7 Tabel Konflik pada LR(0) Parsing Table

Pada state 0 dan 2 terdapat konflik jika terdapat input symbol '('. Action tidak bisa ditentukan apakah yang dilakukan reduce ataukah shift, karena keduanya berada pada state yang sama.

Hal ini disebabkan karena adanya satu atau lebih grammer yang memiliki ciri sebagai berikut:

- Memiliki produksi yang rekursif baik *left recursive* ataupun *right recursive*, dengan bagian kiri produksi yang sama.

Point di atas inilah yang merupakan ciri grammer yang tidak bisa diparsing oleh parser LR(0). Adapun dalam aplikasi Kamus Bahasa Arab yang menggunakan metode parsing ini, semua production dipastikan tidak memiliki pola yang rekursif.

Contoh Penerapan dalam Aplikasi Kamus Bahasa Arab

Berikut ini adalah sebuah contoh untuk penerapan algoritma membentuk LR(0) State Machine dan Tabel Parsing LR(0) dalam aplikasi Kamus Bahasa Arab.

Contoh: di bawah ini merupakan production rules untuk perubahan bentuk kata dengan aturan (wazan) Fa'ala Yaf'ulu

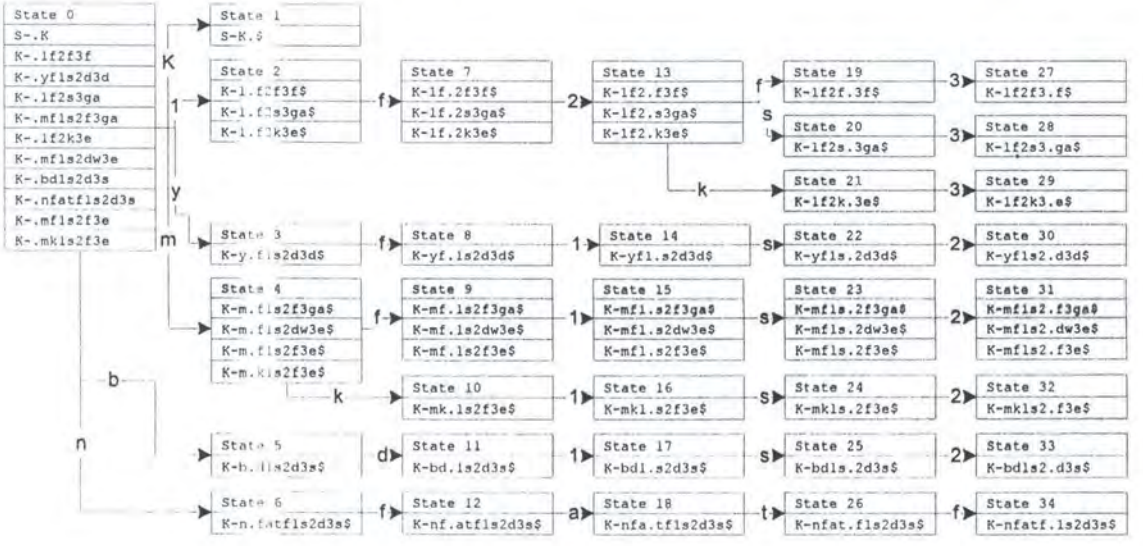
No	Keterangan	Produksi
1	fiil madhi	K → akbkck
2	fiil mudhorik	K → jkaqbmcm
3	mashdar	K → akbqcle
4	mashdar mim	K → hkaqbkcle
5	fa'il (pelaku)	K → akebocn
6	maf'ul	K→ hkaqbmiqcn

7	fi'il amr	K → emaqbmcq
8	fi'il nahi	K → gkefkaqbmcq
9	isim makan / zaman	K → hkaqbkcn
10	isim alat	K → hoaqbkcn

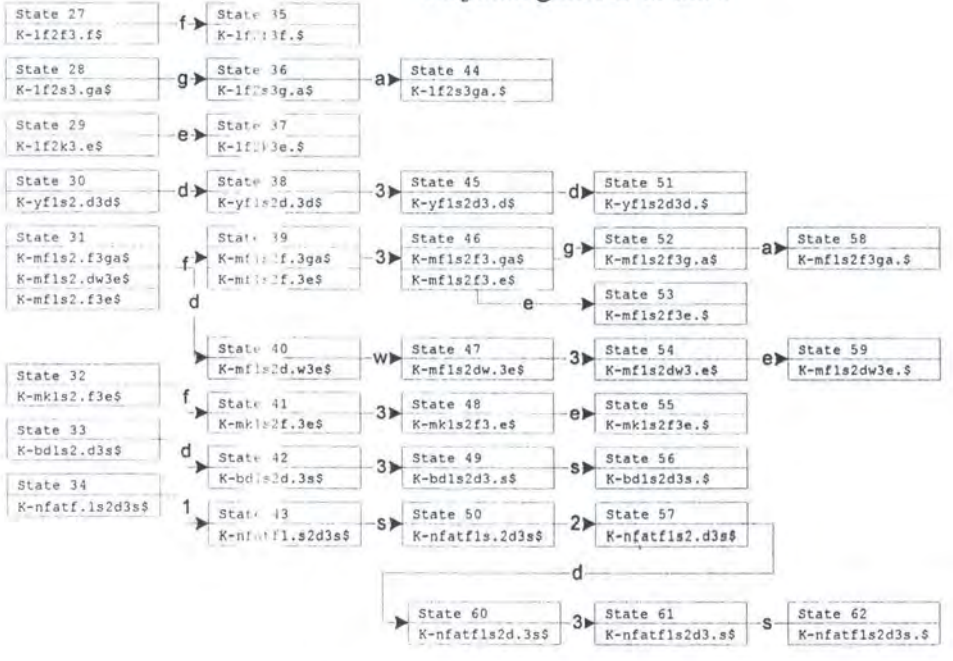
Tabel 4.8 Production Rules untuk Wazn Fa'ala Yaf'ulu

(lihat Appendix 1 untuk keterangan symbol)

Dari grammer di atas dibentuk state machine sbb:



lanjutan gambar di atas:



Gambar 4.2 State Machine dari contoh

4.3 DETEKSI POLA INPUT

Pada waktu melakukan parsing dari masukan yang diinputkan user, parser akan mampu mendeteksi masukan yang diinputkan oleh user, apakah terjadi error ataupun tidak. Jika terjadi error pada waktu parsing, maka masukan dari user akan langsung dicocokkan dengan database yang terdapat pada kamus. Akan tetapi jika 'accepted', maka production yang digunakan untuk reduksi yang terakhir dijadikan petunjuk untuk menentukan pola yang dipakai.

Namun bisa saja ada suatu production direduksi tidak pada akhir dari parsing. Hal ini akan menjadikan catatan tersendiri pada penjelasan kata yang diinputkan. Misalnya pada production yang mendefinisikan *dhomir* (kata ganti) yang mana merupakan bagian dari kata dan tidak bisa berdiri sendiri untuk membentuk kata.

4.4 MENENTUKAN KATA DALAM BENTUK DASAR

Sebagaimana kita ketahui bahwa di dalam kamus hanya tersimpan data dalam bentuk pola dasar (wazn fa'ala), maka untuk mencocokkan data dengan kamus, harus diambil dahulu karakter-karakter yang berhubungan dengan fa'fiil, 'ain fi'il dan lam fi'il (pada grammer ditandai dengan symbol 1, 2, dan 3).

Misalnya diinputkan kata sebagai berikut:

منصر (mim – fathah – nun – sukun – shot – dhommah – ro –dhommatain)

parsing untuk input kata di atas, production yang digunakan untuk reduksi terakhir adalah:

K → hkaqbmiqcn

Dalam database rule, production tersebut memiliki pola dasar:

akbkck

Untuk mendapatkan bentuk dasar dari kata yang diinputkan user tadi, maka diambil huruf yang diwakili simbol **a** (fa' fi'il), **b** ('ain fi'il) dan **c** (lam fi'il). Didapatkan:

Simbol:	a	b	c
Huruf:	ن	ص	ر

Tabel 4.9 Simbol ke huruf Arab dari Contoh

Kemudian, huruf dari simbol di atas dimasukkan ke dalam production pola dasarnya, sehingga didapatkan sebagai berikut:

Pola dasar:	a	f	b	f	c	f
Bentuk Dasar	ن		ص		ر	
	Num	Fathah	Shot	fathah	Ro'	fathah

Tabel 4.10 Pola dasar ke Bentuk Dasar dari Contoh

Maka bentuk dasar dari kata: منصر

Adalah: نصر

4.5 PEMBERIAN ARTI DAN PENJELASAN

Kata dalam bentuk dasar yang didapatkan dari proses sebelumnya dicocokkan ke dalam kamus. Jika ketemu, maka akan diterjemahkan ke dalam bahasa Indonesia. Jika tidak ketemu, maka hanya dijelaskan bahwa kata tersebut masuk ke dalam pola yang didapatkan tadi, namun belum terdapat dalam database kamus.

Untuk mendapatkan penjelasan dari kata tersebut, dicari query pada tabel KeteranganRule yang memiliki transaksi 1-N pada rule yang didapatkan.

BAB V

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

BAB V

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

Dalam bab ini dibahas tentang perancangan dan implementasi input dan output system, diagram alir data, struktur data dan proses.

5.1 Hasil Perangkat Lunak yang diharapkan

Sesuai dengan tujuan dibuatnya tugas akhir ini, maka kriteria yang nanti akan menjadi spesifikasi perangkat lunak yang dibangun adalah sebagai berikut:

1. Perangkat lunak mampu untuk mengembangkan aturan-aturan yang belum ada pada database rule ataupun memodifikasi yang telah ada berdasarkan aturan-aturan rule yang lebih tepat untuk diterapkan dalam pengenalan pola kata Bahasa Arab.
2. Perangkat lunak mampu untuk menambah atau memodifikasi database Kamus yang digunakan untuk menterjemahkan masukan dari user. Adapun dalam database kamus, andaikata merupakan kata jadian, hanya bentuk dasar (bentuk fi'il madhi) dari kata-kata dalam bahasa Arab saja yang disimpan.
3. Perangkat lunak mampu untuk mengkonstruksi *parsing table* dari database production rule yang ada.
4. Perangkat lunak mampu untuk mengenali pola dasar kata yang diinputkan user, andaikata yang diinputkan user tersebut merupakan bentuk kata jadian.
5. Perangkat lunak mampu untuk memberikan penjelasan / makna dari kata yang diinputkan user andaikata kata tersebut terdapat di dalam database kamus.

5.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak yang dilakukan meliputi perancangan data, perancangan proses, dan perancangan antarmuka. Perancangan data terdiri dari perancangan data masukan, data proses, dan data keluaran. Perancangan proses meliputi dari perancangan diagram alir data (DAD). Perancangan antarmuka terdiri dari perancangan hirarki menu, *form* masukan, *form* proses, dan *form* keluaran.

5.2.1 Perancangan Data

Perancangan data bertujuan untuk mengetahui kebutuhan perangkat lunak yang akan dibangun. Pada bagian ini ditentukan data-data yang akan terlibat langsung dalam sistem yang akan dibangun. Perancangan data dapat digolongkan dalam tiga bagian, diantaranya adalah :

- Data Masukan
- Data Proses
- Data Keluaran

a. Data Masukan

Data masukan yang digunakan adalah tabel dari suatu basisdata non-transaksi serta masukan manual dari user. Dalam aplikasi Kamus Bahasa Arab ini berupa:

- Data Kamus
- Data aturan-aturan pola (production rule)
- Masukan manual dari user yang merupakan kata yang akan dicari penjelasan / artinya ke dalam bahasa Indonesia.

b. Data Proses

Data proses adalah data yang digunakan oleh sistem selama proses berlangsung. Di dalam aplikasi Kamus Bahasa Arab ini berupa:

- State machine yang dikonstruksi dari database pola (production rule)
- LR(0) Parsing table yang dikonstruksi dari State Machine.

c. Data Keluaran

Data keluaran dari sistem yang akan dibangun adalah berupa informasi tentang penjelasan / makna dari kata bahasa Arab yang diinputkan oleh user. Penjelasan tersebut bisa meliputi:

- Bentuk / pola kata
- Wazn yang dipakai
- Bentuk dasar dari input yang dimasukkan
- Makna Kata

Keluaran di atas tergantung dari pola yang dikenali dari input yang dimasukkan user. Bisa jadi kurang dari atau lebih dari itu.

5.2.2 Perancangan Proses

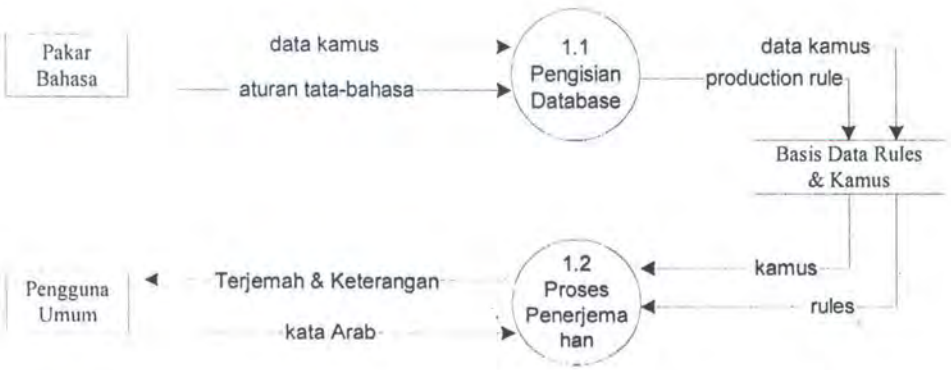
Untuk menggambarkan perancangan proses sistem, akan digunakan suatu *Data Flow Diagram* atau Diagram Aliran Data (DAD).

Pembuatan DAD disini menggunakan software Visio Technical 4.5. Diagram alir data yang dibuat akan menjelaskan proses dari level 0 sampai level 2

Gambar 5.1 DFD Level 0

Pada DFD level 0, (gambar 4.1) proses dimulai dengan penggunaan aplikasi kamus Bahasa Arab oleh pengguna dengan penggunaan database untuk kebutuhan sarana penerjemahan dan pengisian data.

Kemudian, proses 1 dijabarkan pada DFD level 1 dengan membagi proses menjadi proses pengisian database dan proses penerjemahan.

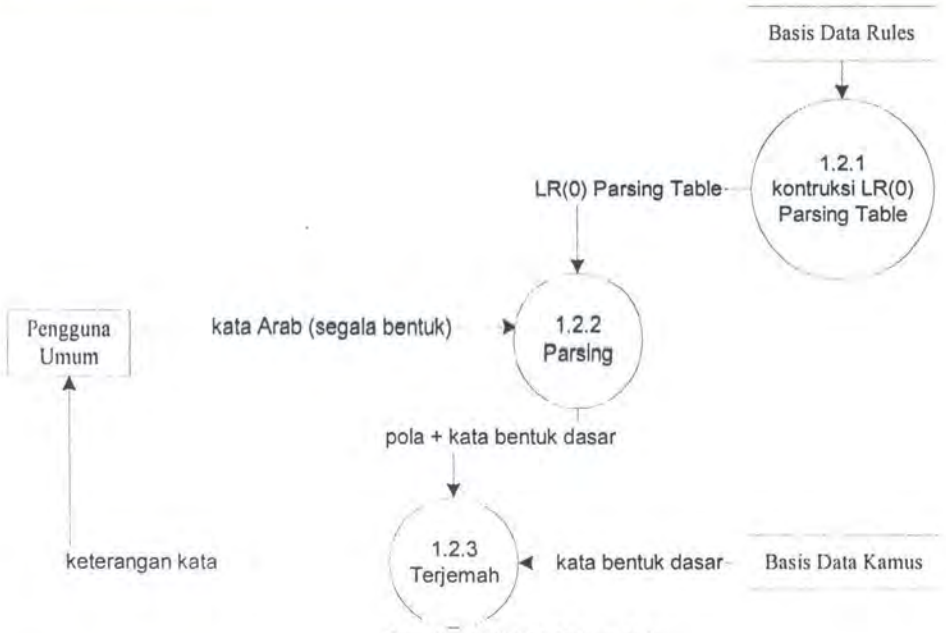


Gambar 5.2 DFD Level 1

Dalam diagram level 1 ini, pengguna dibedakan antara pakar bahasa, yang tentunya mengerti tata bahasa Bahasa Arab dan implementasinya ke production rules dengan pengguna pada umumnya, yang hanya menggunakan aplikasi untuk penerjemahan bahasa

Arab ke dalam bahasa Indonesia. Proses pengisian database dilakukan dengan memasukkan data kamus bahasa Arab dalam bentuk pola dasar dan production rule yang mewakili pola susunan kata bahasa arab. Sedangkan proses penerjemahan dilakukan dengan mengambil data rules dan kamus dari database yang akan diolah dengan masukan dari user untuk menghasilkan keterangan dari masukan tersebut.

Kemudian pada level 2, proses proses penerjemahan (1.2) dijabarkan menjadi sebagai berikut:



Gambar 5.3 DFD Level 2

Untuk penerjemahan, proses dimulai dengan pengambilan seluruh database rules yang akan diolah untuk mendapatkan *LR(0) parsing table*. Kata yang diinputkan dari user (yang bisa merupakan segala perubahan bentuk kata), dilakukan parsing untuk mendapatkan pola dan bentuk dasarnya. Pola dan bentuk dasar dari proses parsing akan dicocokkan dengan basisdata kamus. Jika hasil parsing menghasilkan error, masukan

langsung dicocokkan dengan database kamus dengan mengabaikan pola dan masukan dianggap sebagai bentuk dasar.

5.2.3 Perancangan Struktur Data

Pada bagian ini akan dijelaskan struktur data sistem yang digunakan selama proses penerjemahan dalam Aplikasi Kamus Bahasa Arab.

Struktur rProduksi

Struktur data ini dipergunakan untuk menyimpan dan memproses Production rule yang diambil dari database.

```
rProduksi = record
              id    : word;
              Kiri  : char;
              Kanan : string[32];
end;
```

Keterangan:

Id : no identitas dari production yang disamakan dari database

Kiri : bagian kiri dari suatu production

Kanan : bagian kanan dari suatu production

Struktur rState

Struktur data ini dipergunakan untuk menyimpan dan memproses State Machine yang dikonstruksi dari *production rule*.

```
rState = record
           Nomor          : word;
```

```
        JumlahKernel    : word;
        Kernel          : array[1..32] of rProduksi;
        JumlahItem      : word;
        Item            : array[1..32] of rProduksi;
        GotoChar        : string;
        GotoState       : array[1..32] of word;
end;
```

Keterangan:

Nomor : nomor state

JumlahKernel : jumlah kernel item dari state

Kernel : kumpulan Production dari kernel pada state

JumlahItem : jumlah item dari state

Item : Kumpulan production dari Item pada state

GotoChar : kumpulan karakter untuk goto yang berelasi indeks dengan goto state

GotoState : nomor state yang dituju dari setiap GotoChar

5.2.4 Perancangan Antarmuka

Pada bagian ini akan dijelaskan sruktur rancangan antarmuka dari sistem. Bagian-bagian antarmuka adalah sebagai berikut:

- **Form Tampilan Utama**

Pada form ini merupakan layar utama antarmuka dari aplikasi ini. Di mana user akan memasukkan data dan mendapatkan hasil keterangan (output) nya di layar ini.

- **Form Pengisian database Rule**

  Pada form ini merupakan antarmuka dari sistem untuk penambahan atau modifikasi dari aturan-aturan (production rule) yang digunakan untuk aplikasi Kamus Bahasa Arab. Di dalamnya terdapat pengkodean antara karakter arab menjadi karakter latin untuk mempermudah pembuatan rule.

- **Form Pengisian Database Kamus**

  Pada layar ini merupakan antarmuka dari sistem untuk penambahan atau modifikasi dari kamus data, yang berupa kata dalam bentuk dasar ataupun kata lain yang bukan merupakan kata jadian beserta penjelasannya dalam bahasa Indonesia.

- **Form Pengguna Tingkat Lanjut**

Pada form ini merupakan antarmuka bagi pengguna tingkat lanjut yang ingin mengetahui informasi tentang *state machine* yang telah terbentuk, *parsing table* dan proses parsing dari masukan yang telah diproses.

5.3 PEMBUATAN PERANGKAT LUNAK

Pada bagian sub bab ini akan dibahas tentang lingkungan implementasi, implementasi proses dan implementasi antarmuka .

5.3.1 Lingkungan Implementasi

Dalam pembuatan perangkat lunak ini menggunakan perangkat keras dan perangkat lunak sebagai berikut:

Perangkat Lunak:

- Sistem Operasi Windows 98 with Arabic Support
- Borland Delphi 4.0 untuk pembuatan program
- Microsoft Access sebagai database engine.
- ODBC driver untuk menghubungkan database MS Access ke aplikasi.

Perangkat Keras:

- Komputer dengan Processor Pentium II / 400 Mhz
- Memory 64Mb
- Monitor Super VGA dengan resolusi 800x600, warna 16 bit.

Kebutuhan Sistem

Adapun aplikasi ini, memiliki kebutuhan sistem minimal sebagai berikut:

- Menggunakan Sistem Operasi Windows dengan Arabic Support

Adapun kebutuhan yang lain, cukup dengan komputer standar yang sering dipakai oleh masyarakat pada umumnya selama perangkat tersebut mendukung sistem operasi di atas. Adapun untuk kebutuhan memori tergantung dari banyaknya jumlah database rule dan kamus, namun 32Mb dianggap maksimal karena aplikasi ini tidak membutuhkan banyak memori.

5.3.2 Implementasi Antarmuka

Pada Sub bab ini akan dibahas tentang implementasi antarmuka yang dibuat untuk aplikasi Kamus Bahasa Arab ini.

5.3.2.1 Form Utama

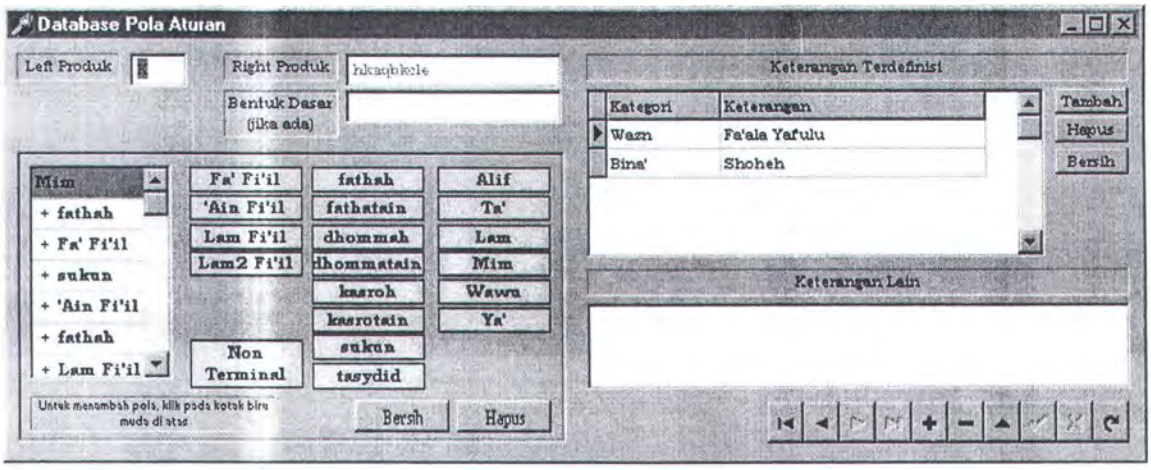
Form ini merupakan tampilan utama program yang mana pengguna umum akan berinteraksi dengan program untuk memasukkan kata yang akan dicari terjemah atau penjelasannya.

Bagian atas merupakan pengisian kata arab yang akan dicari penjelasan/maknyanya sedangkan bagaian bawah merupakan hasil proses penerjemahan yang berupa keterangan dari kata yang dimasukkan. Untuk pengguna khusus, terdapat popup (menu yang muncul ketika diklik kanan) untuk menampilkan pilihan untuk modifikasi database kamus ataupun rules.

Gambar 5.4 Form Utama Aplikasi

5.3.2.2 Form Pengisian database Rules

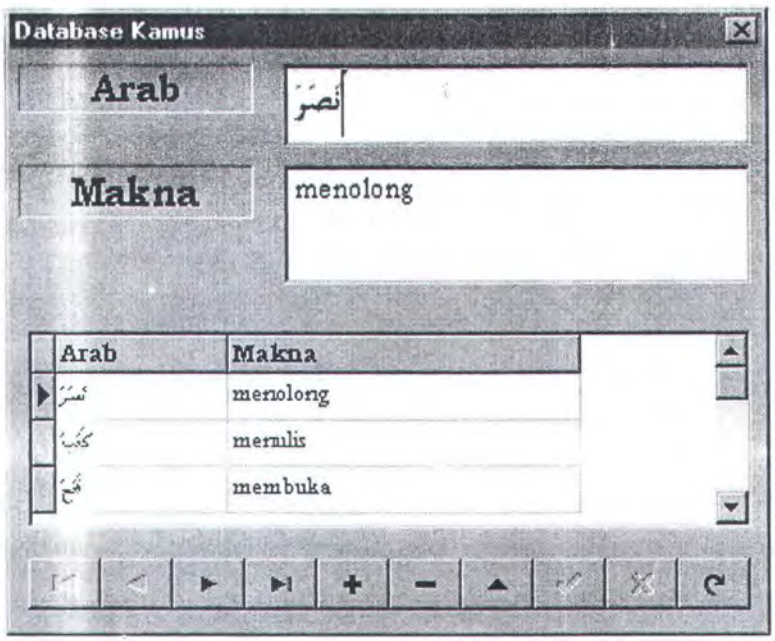
Form ini merupakan antarmuka untuk melakukan pengisian ataupun modifikasi database untuk aturan-aturan bahasa arab yang berupa production rule.



Gambar 5.5 Form Pengisian Database Rules

5.3.2.3 Form Pengisian Database Kamus

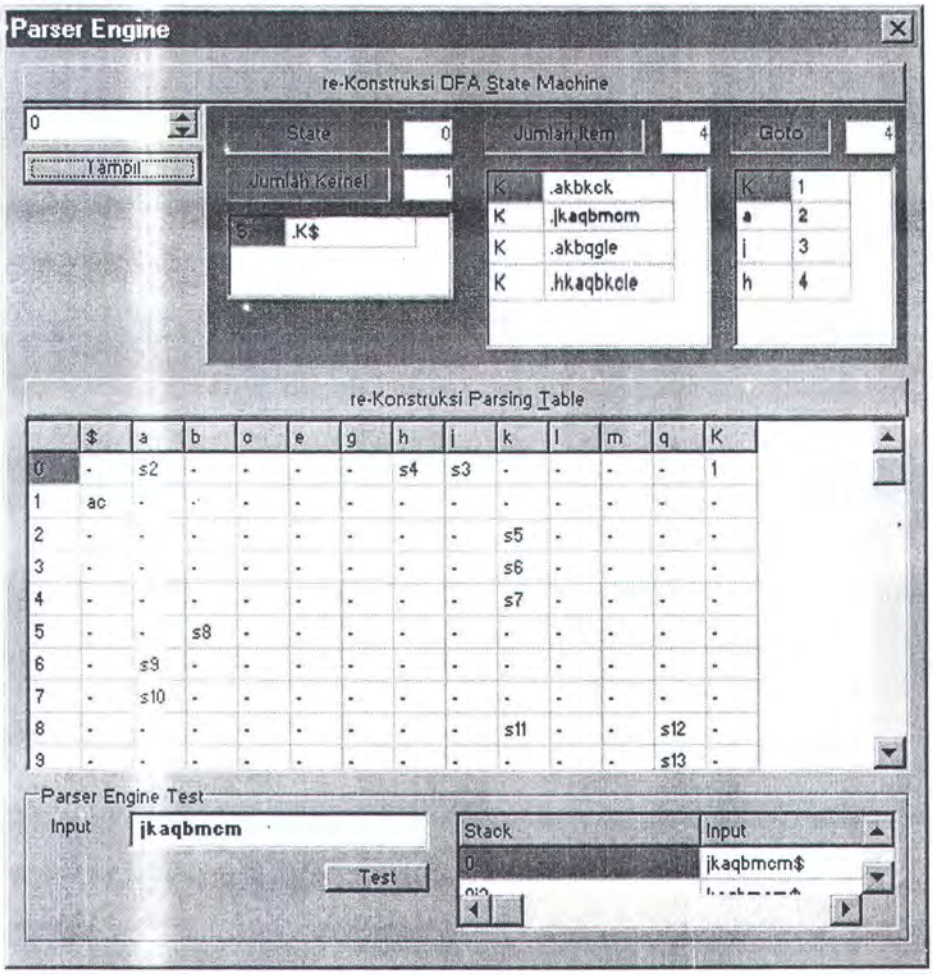
Form ini merupakan antarmuka untuk melakukan proses pengisian data atau modifikasi dari kamus. Masukan berupa kata kerja bentuk dasar ataupun kata bentuk umum (nakiroh).



Gambar 5.6 Form Database Kamus

5.3.2.4 Form Pengguna Tingkat Lanjut

Form ini merupakan antarmuka bagi user tingkat lanjut, yang ingin mengetahui informasi bagaimana state machine, parsing table dan proses parsing yang dilakukan terhadap kata masukan yang telah diproses.



Gambar 5.7 Form Pengguna Tingkat Lanjut

5.3.3 Implementasi Proses

Pada sub bab ini akan dibahas tentang implementasi proses yang digunakan dalam aplikasi Kamus Bahasa Arab.

5.3.3.1 Proses Pembacaan Production Rules

Proses ini merupakan proses untuk membaca production dari database ke array produksi. Yang dilakukan adalah membaca setiap record pada database, yang kemudian dimasukkan ke dalam array produksi.

(Coding Delphi beserta keterangan, terdapat dalam lampiran 3, dg nama procedure TFUtama.BacaRuleProduksi)

5.3.3.2 Proses konstruksi LR(0) State Machine

Proses ini merupakan proses pembentukan state machine dengan masukan (input) berupa *production rules* yang disimpan dalam variabel **Produksi**. Proses ini membutuhkan sub proses lain, yaitu proses kontruksi StateItem dan kontruksi Goto Item. Dalam prosedur ini dilakukan:

- Penentuan symbol secara unik. Yaitu mengambil simbol-simbol apa saja yang dipakai dengan menghilangkan duplikasi.
- Simbol urut tadi diurutkan untuk menstandarkan tampilan Parsing Table.
- Menambahkan setiap non terminal secara unik
- Menambahkan simbol dengan karakter $ yang merupakan simbol yang bermakna accepted.
- Melakukan iterasi dengan membentuk state baru beserta kernelnya maupun goto item sehingga closure yang terakhir (yakni closure yang tidak menemukan lagi simbol setelah titik).

(Coding Delphi beserta keterangan, terdapat dalam lampiran 3, dg nama procedure TFParsingTable.BuildStateMachine)

5.3.3.3 Proses Kontruksi State Item dan Goto Item

Pada proses ini dilakukan kontruksi state pada state machine untuk setiap satu state. Proses ini dilakukan secara rekursif hingga tidak ada lagi state machine baru yang terbentuk.

(Coding Delphi beserta keterangan, terdapat dalam lampiran 3, dg nama procedure procedure TFParsingTable.BuildStateItem(StateNum: word) dan procedure TFParsingTable.BuildGotoItem(noState: word))

5.3.3.4 Proses Kontruksi LR(0) Parsing Table

Pada proses ini dilakukan pembentukan LR(0) Parsing Table dengan input state machine yang telah dibentuk sebelumnya.

(Coding Delphi beserta keterangan, terdapat dalam lampiran 3, dg nama procedure TFParsingTable.BuildParsingTabel)

5.3.3.5 Proses Parsing

Proses ini merupakan proses pengenalan pola dari masukan pengguna. Masukan berupa masukan dari pengguna dan parsing table. Sedangkan keluaran berupa nomor-nomor produksi yang telah digunakan untuk reduksi dalam proses parsing. Jika parsing menghasilkan error, maka keluaran menghasilkan 0.

(Coding Delphi beserta keterangan, terdapat dalam lampiran 3, dg nama function TFParsingTable.Parsing: word)

BAB VI

UJI COBA PERANGKAT LUNAK

BAB VI
UJI COBA PERANGKAT LUNAK

Dalam bab ini dijelaskan tentang hasil uji coba perangkat lunak dengan berbagai macam masukan.

6.1 LINGKUNGAN IMPLEMENTASI

Uji coba dilakukan dengan spesifikasi perangkat keras dan lunak sebagai berikut:

Perangkat Keras:

- Komputer processor Pentium III/500 Mhz
- Memory 64 Mb
- Monitor SVGA resolusi 800x600 warna 16 bit

Perangkat Lunak

- Sistem Operasi Microsoft Windows 98 with Arabic Support

6.2 SPESIFIKASI DATABASE RULES

Dalam database rules telah dimasukkan aturan-aturan dalam bentuk production rule yang mewakili tashrif wazn fa'ala yaf'ulu dengan bina' shohih (tanpa terdapat huruf illat), yaitu sebagai berikut:

Left	Right	Bentuk Dasar	Keterangan
K	akbkck	akbkck	fiil madhi

K	jkaqbmcm	akbkck	fiil mudhorik
K	akbqgle	akbkck	mashdar ghoiru mim
K	hkaqbkcle	akbkck	mashdar mim

Tabel 6.1 Spesifikasi Data Rules untuk Ujicoba

6.3 SPESIFIKASI CONTOH DATA UNTUK KAMUS

Dalam kamus diisikan data yang berupa bentuk dasar -jika kata tersebut bisa mengalami perubahan- ataupun kata yang berbentuk umum dan tunggal (mufrod-nakiroh) -jika kata tersebut merupakan kata benda murni-.

Dalam database kamus diisikan sebagai berikut:

Arab	Indonesia	Keterangan
نصر	menolong	kata kerja dasar
كتب	menulis	kata kerja dasar
قمر	bulan	kata benda umum-tunggal

Tabel 6.2 Spesifikasi Data Kamus untuk Ujicoba

6.4 HASIL UJICOBA

Data masukan di atas yang telah dimasukkan dalam database akan diujicoba dengan beberapa masukan dengan kriteria sebagai berikut:

- dianggap accepted oleh parser dalam bentuk yang berbeda dengan bentuk dasar.
- dianggap error oleh parser

Sebelum dilihat hasil dari input tersebut, perlu diketahui pula hasil sementara sebelum diproses dengan input tersebut seperti hasil kontruksi parsing tabel.

6.4.1 Hasil LR(0) Parsing Table

Berikut ini adalah hasil LR(0) Parsing Table yang dihasilkan dari production rule dengan spesifikasi di atas:

	$	a	b	c	e	g	h	j	k	l	m	q	K
0	-	s2	-	-	-	-	s4	s3	-	-	-	-	1
1	ac	-	-	-	-	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-	s5	-	-	-	-
3	-	-	-	-	-	-	-	-	s6	-	-	-	-
4	-	-	-	-	-	-	-	-	s7	-	-	-	-
5	-	-	s8	-	-	-	-	-	-	-	-	-	-
6	-	s9	-	-	-	-	-	-	-	-	-	-	-
7	-	s10	-	-	-	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-	s11	-	-	s12	-
9	-	-	-	-	-	-	-	-	-	-	-	s13	-
10	-	-	-	-	-	-	-	-	-	-	-	s14	-
11	-	-	-	s15	-	-	-	-	-	-	-	-	-
12	-	-	-	-	-	s16	-	-	-	-	-	-	-
13	-	-	s17	-	-	-	-	-	-	-	-	-	-
14	-	-	s18	-	-	-	-	-	-	-	-	-	-

15	-	-	-	-	-	-	-	-	s19	-	-	-	-
16	-	-	-	-	-	-	-	-	-	s20	-	-	-
17	-	-	-	-	-	-	-	-	-	-	s21	-	-
18	-	-	-	-	-	-	-	-	s22	-	-	-	-
19	r1	-	-	-	-	-	-	-	-	-	-	-	-
20	-	-	-	-	s23	-	-	-	-	-	-	-	-
21	-	-	-	s24	-	-	-	-	-	-	-	-	-
22	-	-	-	s25	-	-	-	-	-	-	-	-	-
23	r3	-	-	-	-	-	-	-	-	-	-	-	-
24	-	-	-	-	-	-	-	-	-	-	s26	-	-
25	-	-	-	-	-	-	-	-	-	s27	-	-	-
26	r2	-	-	-	-	-	-	-	-	-	-	-	-
27	-	-	-	-	s28	-	-	-	-	-	-	-	-
28	r4	-	-	-	-	-	-	-	-	-	-	-	-

Tabel 6.3 LR(0) Parsing Table Hasil Uji Coba

6.4.2 Hasil dengan input yang dianggap accepted oleh parser

Input dengan kriteria ini misalnya adalah sebagai berikut:

ينصر	bentuk fiil mudhorik dari kata نصر
منصرا	bentuk mashdar mim dari kata نصر
نصرا	bentuk mashdar ghoiru mim dari kata نصر

Tabel 6.4 Masukan untuk Ujicoba

yang dicoba adalah dengan masukan ينصر

Hasil translate ke simbol production adalah: jkaqbmcm

Dengan input di atas, hasil parsing adalah sebagai berikut:

Stack	Input	Action
0	jkaqbmcm$	s3
0j3	kaqbmcm$	s6
0j3k6	aqbmcm$	s9
0j3k6a9	qbmcm$	s13
0j3k6a9q13	bmcm$	s17
0j3k6a9q13b17	mcm$	s21
0j3k6a9q13b17m21	cm$	s24
0j3k6a9q13b17m21c24	m$	s26
0j3k6a9q13b17m21c24m26	$	r2
0K1	$	ac

Tabel 6.5 Proses Parsing untuk yang berakhir dengan Accepted

dari database didapatkan bentuk dasarnya adalah: akbkck

sehingga bentuk dasar dari ينصر adalah نصر

dari query berdasarkan nomor id rule yang dipakai, didapatkan keterangan sebagai berikut:

- Wazn = fa'ala yaf'ulu
- bina' = shohih
- bentuk = fiil mudhorik → Arti = *sedang / akan*

dari database kamus didapatkan arti dari نصر adalah *menolong*

sehingga arti/makna dari kata ينصر adalah: *sedang / akan menolong*

6.4.3 Hasil dengan input yang dianggap error oleh parser

Input yang dicoba adalah: قمر

Translate ke simbol: akhkbn

Hasil Parsing:

Stack	Input	Action
0	akhkbn$	s2
0a2	khkbn$	s5
0a2k5	hkbn$	-

Tabel 6.6 Proses Parsing yang Berakhir dengan Error

Karena hasil parsing menghasilkan error, maka kata قمر langsung dicocokkan ke dalam kamus.

Hasil pencocokan dengan hasil kamus menghasilkan, bahwa arti dari kata قمر adalah *bulan.*

6.5 ANALISA

Dari uji coba di atas terlihat bahwa untuk production yang semodel dengan aturan-aturan bahasa Arab bisa diterapkan dalam LR(0) parser, karena dalam perubahan bentuk kata dalam bahasa Arab tidak terdapat suatu production yang rekursif.

BAB VII

PENUTUP

BAB VII
PENUTUP

Pada bab ini berisi beberapa kesimpulan dari Tugas Akhir yang dibuat berdasarkan hasil ujicoba perangkat lunak. Selain itu dibahas pula kemungkinan pengembangan lebih lanjut dari Tugas Akhir ini.

7.1 KESIMPULAN

Dari hasil ujicoba perangkat lunak dapat disimpulkan sebagai berikut:

- Metode parsing bisa digunakan untuk pengenalan pola bentuk kata pada bahasa Arab.
- LR(0) Parsing Table dianggap cukup untuk mengimplementasikan production rule yang mewakili perubahan bentuk kata pada bahasa Arab.
- Penyimpanan pola perubahan kata bahasa Arab yang disimpan dalam bentuk production rule ke dalam suatu database yang bisa dimodifikasi, memungkinkan untuk bisa dilakukan pengembangan lebih lanjut tanpa mengubah source program. Demikian pula untuk penyimpanan data kamus.
- Pengambilan bentuk dasar dari suatu kata berbahasa Arab yang telah diketahui polanya dalam bentuk production rule bisa diketahui dengan mengambil token-token tertentu yang mewakili fa' fi'il, 'ain fi'il ataupun lam fi'il.

7.2 KEMUNGKINAN PENGEMBANGAN

Beberapa kemungkinan pengembangan pada aplikasi Kamus Bahasa Arab ini adalah sebagai berikut:

- Melengkapi database rules dengan berkonsultasi kepada pakar Bahasa Arab, sehingga lingkup perubahan kata yang bisa dicakup lebih luas lagi.
- Menambahkan database kamus sebanyak mungkin sehingga menjadi kamus yang lebih lengkap.
- Tidak membatasi aplikasi harus berjalan dengan Sistem Operasi Windows dengan support Arabic, namun bisa berjalan pula tanpa Arabic Support.
- Menambahkan keterangan/help yang selengkap mungkin sehingga bisa pula dijadikan sarana mempelajari Bahasa Arab dengan mudah.

DAFTAR PUSTAKA

DAFTAR PUSAKA

1. *Aunur Rofiq Ghufron; Qowaaidul Lughotul Arobiyah, Ma'had Al Furqon Gresik.*
2. *Yunus, Mahmud, Prof. H, Kamus Arab Indonesia, Yayasan Penyelenggara Penterjemah Pentafsiran Al Qur'an Jakarta, 1973.*
3. *Wahyu Hidayat, Herlambang, Perancangan dan Pembuatan Perangkat Lunak Kamus Bahasa Arab Indonesia*
4. *V. Aho, Alfred, "Principles of Compiler Design", Princeton University*
5. *Allen I. Holub, "Compiler Design in C", Prentince Hall International, Inc.*

LAMPIRAN

Lampiran 1

Keterangan Simbol

Simbol	Keterangan
a	huruf yang mewakili fa' fi'il
b	huruf yang mewakili 'ain fi'il
c	huruf yang mewakili lam fi'il
d	huruf yang mewakili lamlam fi'il
e	huruf alif
f	huruf ta'
g	huruf lam
h	huruf mim
i	huruf wawu
j	huruf ya'
k	fathah
l	fathatain
m	dhommah
n	dhommatain
o	kasroh
p	kasrotain
q	sukun
r	tasydid

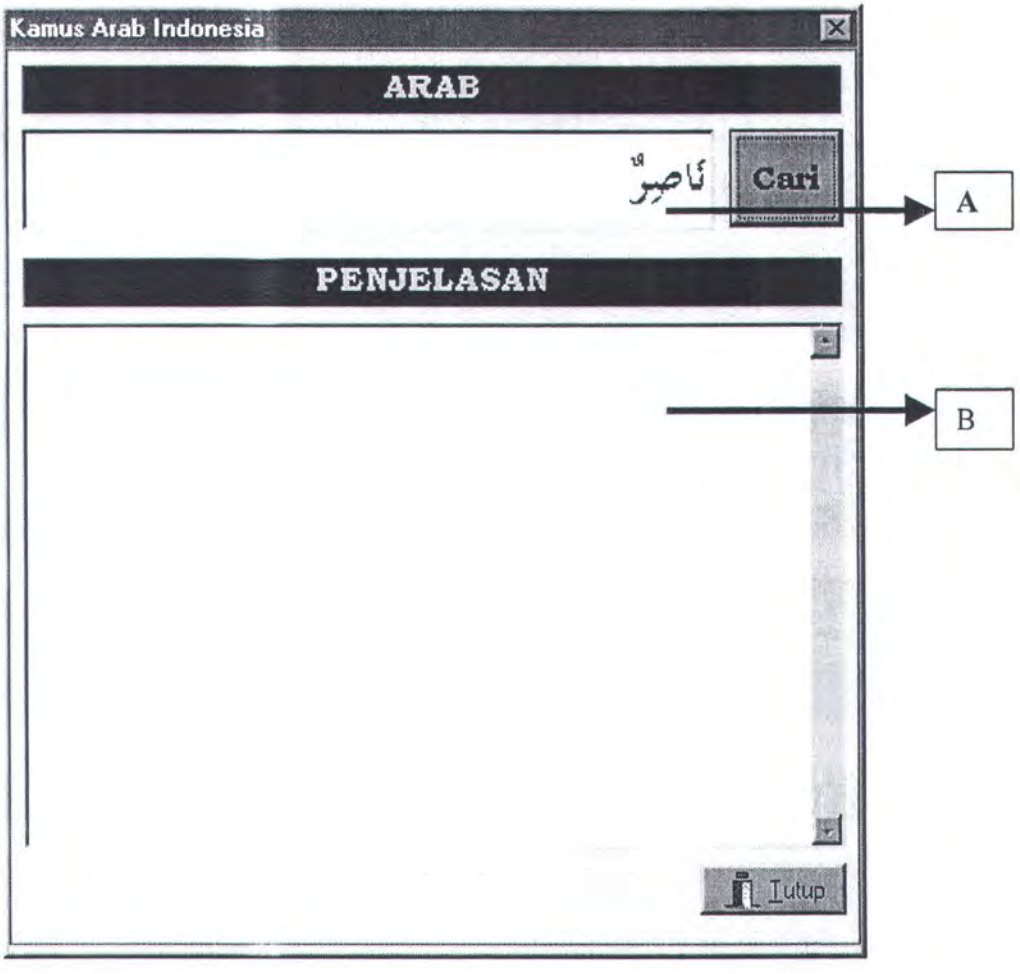
Lampiran 2:

Petunjuk Pemakaian

Pencarian / Penerjemahan Kata

Antarmuka dari aplikasi ini adalah sebagaimana antarmuka program kamus pada umumnya, yaitu berbentuk sebuah window kecil yang berjenis tools window. Karena memang aplikasi ini berupa program bantu untuk menerjemahkan kata-kata yang mungkin terdapat pada aplikasi lain di window lain yang lebih besar.

Tampilan utama adalah sebagai berikut:



Bagian A adalah kotak edit tempat mengisikan kata berbahasa Arab yang ingin diterjemahkan. Sedangkan bagian B adalah kotak keterangan hasil dari proses penerjemahan dari kata berbahasa Arab yang dimasukkan pada bagian A.

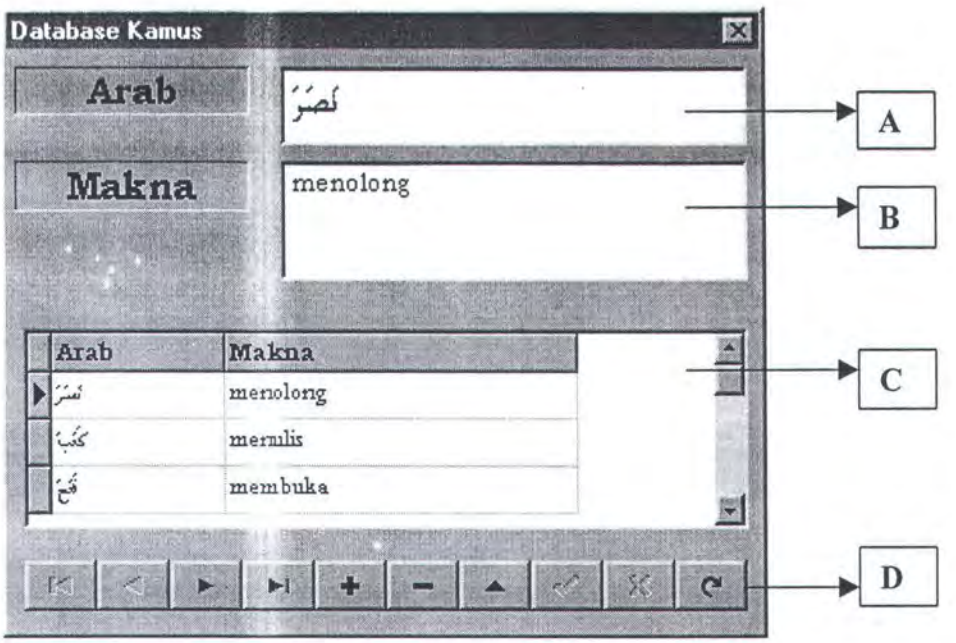
Untuk melakukan penerjemahan, masukkan kata berbahasa Arab yang dimaksudkan pada bagian A, kemudian klik buttin "Cari" pada samping kotak tersebut.

Misalnya dengan mengetikkan kata: نَاصِرٌ

Akan mendapatkan hasil penerjemahan sebagai berikut:

Pemasukan dan Modifikasi Data Kamus

Untuk menambahkan data di kamus, pada layar utama, lakukan klik kanan untuk menampilkan pilihan untuk pengguna khusus. Kemudian pilih: Dictionary. Maka akan muncul tampilan sebagai berikut:



Untuk menambahkan data pada kamus, klik button + pada bagian D. Kemudian anda diminta untuk mengisikan setiap field pada bagian A dan B. Bagian A anda harus memasukkan kata berbahasa Arab dan bagian B adalah makna dari kata tersebut.

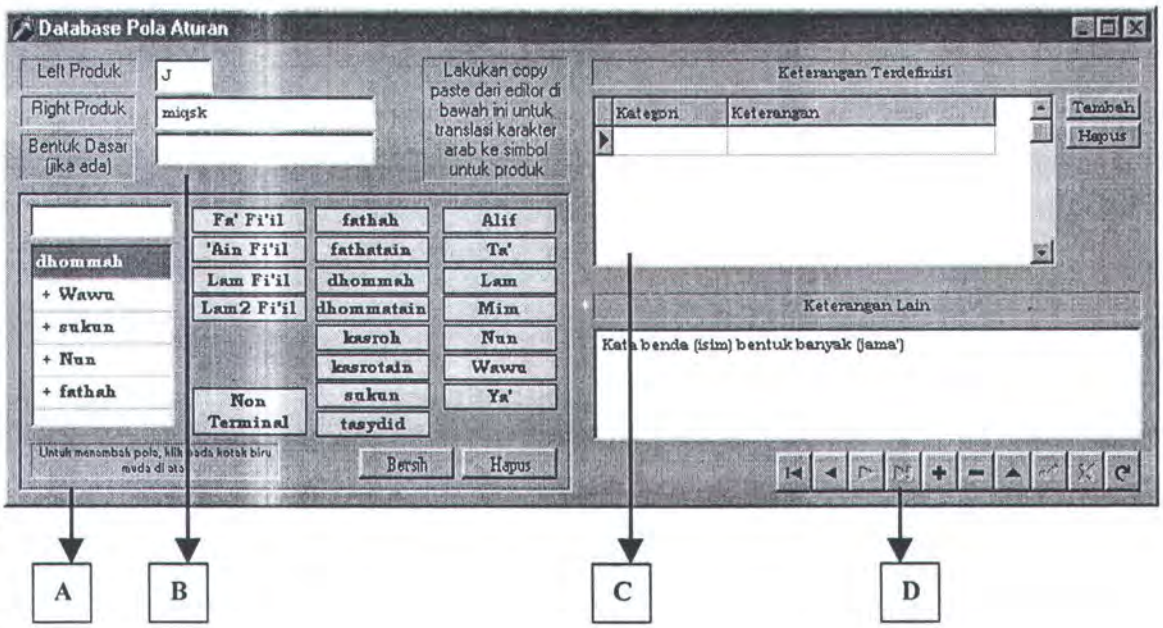
INGAT: Bahwa dalam aplikasi ini, data kamus hanya berupa kata dalam bentuk dasar ataupun kata umum yang bukan berupa kata yang bisa berubah bentuk.

Bagian C dipergunakan untuk navigasi yang berbentuk *grid* dari data yang terpilih untuk dilakukan modifikasi, baik untuk diubah ataupun dihapus. Button – untuk menghapus, sedangkan button ^ untuk mengubah.

Selain itu navigasi bisa dilakukan pada button-button yang terdapat pada bagian D.

Penambahan dan Modifikasi Data Rules / Production

Pada Layar utama, klik kanan lalu klik pilihan Production Rules. Maka akan muncul tampilan sebagai berikut:



Untuk melakukan penambahan aturan, klik button + pada bagian D. Setelah itu anda diminta untuk mengisikan data production rule. Bagian A merupakan simbol editor untuk mempermudah coding dari karakter Arab ke simbol. Untuk menambah sebuah token/simbol klik pada kotak bagian kanan, maka akan muncul keterangan di kirinya tentang urutan simbol beserta keterangannya yang bisa diubah berdasarkan keperluan. Jika selesai, lakukan copy dan paste dari field edit simbol ke bagian field yang akan dimasukkan (apakah itu bagian field bentuk dasar ataupun left production).

Setelah itu, berikan keterangan yang sesuai untuk keterangan khusus tentang rule tersebut (keterangan terdefinisi jangan diisikan dulu !). Jika sudah selesai dan akan menyimpannya, klik button yang berbentuk check (√). Setelah itu, untuk menambah keterangan terdefinisi, klik button 'Tambah' pada bagian C sebelah kanan. Maka akan muncul layar sebagai berikut:

Pertama kali, pilih kategori pada combo-box pada bagian bawah. Daftar pada sub kategori pada grid tergantung dari nilai yang terdapat pada kategori. Kemudian keterangan bisa dilihat pada sebelah kanannya. Jika sudah benar, klik button 'Tambah' untuk menambahkan pada daftar keterangan terdefinisi pada aturan yang tadi. Untuk menambah keterangan terdefinisi yang lain lakukan hal yang sama seperti di atas.

LAMPIRAN 3: LISTING PROSEDUR

```
procedure TFUtama.BacaRuleProduksi;
var I     : word;
   st     : string;
begin
   JumlahProduksi:=DM.Rules.RecordCount;
   DM.Rules.First;
   for i:=1 to DM.Rules.Recordcount do begin
      st:=DM.Rules['RU_Kiri'];
      Produksi[i].Kiri := st[1];
      Produksi[i].Kanan := DM.Rules['RU_Kanan'];
      DM.rules.next;
   end;
end;
```

```
procedure TFParsingTable.BuildStateMachine;
var i,j   : word;
   tc    : char;
begin

   // cari ItemUnik
   ItemUnik:='';
   for i:=1 to JumlahProduksi do begin
      for j:=1 to length(Produksi[i].Kanan) do
         if not(Produksi[i].Kanan[j] in ['A'..'Z']) then
            if pos(Produksi[i].Kanan[j], ItemUnik)=0 then ItemUnik:=ItemUnik+Produksi[i].Kanan[j];
   end;
```

```
// Urutkan Item Unik
for i:=1 to Length(ItemUnik) do
   for j:=i+1 to Length(ItemUnik) do
      if ItemUnik[i]>ItemUnik[j] then begin
        tc := ItemUnik[i];
        ItemUnik[i] := ItemUnik[j];
        ItemUnik[j] := tc;
      end;

// Tambahkan Symbol NT
for i:=1 to JumlahProduksi do
      if pos(Produksi[i].Kiri, ItemUnik)=0 then ItemUnik := ItemUnik + Produksi[i].Kiri;

// Tambahkan $
ItemUnik := '$' + ItemUnik;


with state[0] do begin
    Nomor:=0;
    JumlahKernel:=1;
    Kernel[1].Kiri:='S';
    Kernel[1].Kanan:='.K$';
end;
JumlahState:=1;
SpinEdit1.MaxValue:=JumlahState;

i:=0;
```

```
    while (i<JumlahState) do begin
//    while (i<50) do begin
        BuildStateItem(i);
        BuildGotoItem(i);
        inc(i);
    end;
end;
```

```
    procedure TFParsingTable.BuildStateItem(StateNum: word);
    var i,j: word;
      dotPosition: word;
      CharRightofTheDot: char;
    begin
       with State[StateNum] do begin
          JumlahItem:=0;
          for i:=1 to JumlahKernel do begin
            dotPosition := pos('.',Kernel[i].Kanan);
            if dotPosition < length(Kernel[i].Kanan) then begin
             CharRightofTheDot:= Kernel[i].Kanan[dotPosition+1];
              if CharRightofTheDot in ['A'..'Z'] then
                for j:=1 to JumlahProduksi do
                  if Produksi[j].Kiri = CharRightofTheDot then begin
                    inc(JumlahItem);
                          Item[JumlahItem].Kiri:=CharRightofTheDot;
                    Item[JumlahItem].Kanan:= '.' +Produksi[j].Kanan;
              end;
```

```
        end;
      end;

      i:=1;
      while i<=JumlahItem do begin
        dotPosition := pos('.',Item[i].Kanan);
        if dotPosition < length(Item[i].Kanan) then begin
          CharRightofTheDot := Item[i].Kanan[dotPosition + 1];
          if (CharRightofTheDot in ['A'..'Z']) AND (CharRightofTheDot <> Item[i].Kiri) then
            for j:=1 to JumlahProduksi do
              if Produksi[j].Kiri = CharRightofTheDot then begin
                inc(JumlahItem);
                Item[JumlahItem].Kiri:=CharRightofTheDot;
                Item[JumlahItem].Kanan:= '.' + Produksi[j].Kanan;
              end;
        end;
        inc(i);
      end;
    end;
  end;
-------------------------------------------------------------------------

  procedure TFParsingTable.BuildGotoItem(noState: word);
  var i, s: word;
    dotPosition, idGoto: word;
    CharRightDot: char;
    NewKernel: string;
    stateSudahAda    : boolean;
```

```
// Menentukan Goto item
// sekaligus membentuk state baru beserta kernel itemnya
begin
   with State[noState] do begin
   // Cari GotoChar
      GotoChar:='';
      for i:=1 to JumlahKernel do begin
        dotPosition := pos('.', Kernel[i].Kanan);
        if dotPosition<length(Kernel[i].Kanan) then begin
          CharRightDot := Kernel[i].Kanan[dotPosition+1];
          if CharRightDot<>'$' then begin
            if (pos(CharRightDot, GotoChar)=0) then begin
              // Menambah state jika belum ada, indeksGoto = 1
              GotoChar := GotoChar + CharRightDot;
              inc(JumlahState);
              idGoto := Length(GotoChar);
              GotoState[idGoto]:=JumlahState-1;
              State[GotoState[idGoto]].JumlahKernel:=1;
            end
            else begin
              // Jika state sudah ada, ambil indeksGoto dari
              // yg sudah dicatat sebelumnya
              idGoto := pos(CharRightDot, GotoChar);
              inc(State[GotoState[idGoto]].JumlahKernel);
            end;
            // Menambahkan Kernel Item di state yang dituju berdasarkan
            // indeksGoto
```

```
            stateSudahAda:=false;
            NewKernel:=copy(Kernel[i].Kanan,1,dotPosition-1)  +  CharRightDot  +  '.'  +
copy(Kernel[i].Kanan,dotPosition + 2, length(Kernel[i].Kanan)-dotPosition - 1);
            for s := 1 to JumlahState do
              if (State[s].Kernel[1].Kiri = Kernel[i].Kiri)
                AND (State[s].Kernel[1].Kanan = NewKernel) then begin
                  GotoState[idGoto]:=s;
                  dec(JumlahState);
                  StateSudahAda:=true;
                  break;
              end;

            if not(StateSudahAda) then begin

State[GotoState[idGoto]].Kernel[State[GotoState[idGoto]].JumlahKernel].Kiri:=Kernel[i].Kiri;

State[GotoState[idGoto]].Kernel[State[GotoState[idGoto]].JumlahKernel].Kanan:=NewKernel;
             end;
           end;
         end;
       end;

       // Lakukan hal yang sama untuk item selain kernel
       for i:=1 to JumlahItem do begin
         dotPosition := pos('.', Item[i].Kanan);
         if dotPosition<length(Item[i].Kanan) then begin
           CharRightDot := Item[i].Kanan[dotPosition+1];
           if CharRightDot<>'$' then begin
```

```
if (pos(CharRightDot, GotoChar)=0) then begin
  // jika State belum ada, ditambahkan, indeksGoto = 1
  GotoChar := GotoChar + CharRightDot;
  inc(JumlahState);
  idGoto := Length(GotoChar);
  GotoState[idGoto]:=JumlahState-1;
  State[GotoState[idGoto]].JumlahKernel:=1;
end
else begin
  // Jika state sudah ada, ambil indeksGoto dari
  // yg sudah dicatat sebelumnya
  idGoto := pos(CharRightDot, GotoChar);
  inc(State[GotoState[idGoto]].JumlahKernel);
end;
// Menambahkan Kernel Item di state yang dituju berdasarkan
// indeksGoto

StateSudahAda:=false;
NewKernel:=copy(Item[i].Kanan,1,dotPosition-1) + CharRightDot + '.' +
copy(Item[i].Kanan,dotPosition+2, length(Item[i].Kanan)-dotPosition-1);
for s:=1 to JumlahState do
  if (State[s].Kernel[1].Kiri = Item[i].Kiri)
    AND (State[s].Kernel[1].Kanan = NewKernel) then begin
      GotoState[idGoto]:=s;
      dec(JumlahState);
      StateSudahAda:=true;
      break;
    end;
```

```
            if StateSudahAda=false then begin
State[GotoState[idGoto]].Kernel[State[GotoState[idGoto]].JumlahKernel].Kiri:=Item[i].Kiri;
State[GotoState[idGoto]].Kernel[State[GotoState[idGoto]].JumlahKernel].Kanan:=NewKernel;
            end;


          end;
        end;
      end;
    end;
end;
```

```
procedure TFParsingTable.BuildParsingTabel;
var i, j, k     : word;
  CharTerakhir : char;
  indeksKolom  :  byte;
  ts           : string;
  FollowItems, KananKini    : string;
  noProduksi    : word;
begin
   noProduksi:=0;
   for i:=1 to length(ItemUnik) do BarisPT[i,0]:=ItemUnik[i];

   // Inisialisasi dg error
   for i:=1 to JumlahState do
      for j:=1 to length(ItemUnik) do
         BarisPT[j, i] := '-';
```

```
// Pengisian shift dan goto
indeksKolom:=0;
for i:=0 to JumlahState-1 do begin
   BarisPT[0,i+1] := inttostr(i);
   for j:=1 to length(State[i].GotoChar) do begin
      for k:=1 to length(ItemUnik) do
         if State[i].GotoChar[j] = ItemUnik[k] then begin
            indeksKolom:=k;
            break;
         end;

      case State[i].GotoChar[j] of
         'A'..'Z' : ts:='';
         else ts:='s'
      end;

      BarisPT[indeksKolom,i+1] := ts + inttostr(State[i].GotoState[j]);
   end;
end;

//Pengisian reduce dan accept
for i:=1 to JumlahState-1 do
   with State[i] do begin
      if JumlahKernel=1 then begin
         CharTerakhir := Kernel[1].Kanan[length(Kernel[1].Kanan)];
         if CharTerakhir = '$' then BarisPT[pos('$',ItemUnik), i+1] := 'ac';
         if CharTerakhir = '.' then begin
```

```
                FollowItems := GetFollowItems(Kernel[1].Kiri);
                for j:=1 to length(FollowItems) do begin
                  KananKini := Kernel[1].Kanan;
                  delete(KananKini, pos('.', KananKini), 1);
                  for k:=1 to JumlahProduksi do
                    if Produksi[k].Kanan = KananKini then begin
                      noProduksi := k;
                      break;
                    end;
                  BarisPT[pos(FollowItems[j],ItemUnik), i+1] := 'r' + inttostr(NoProduksi);
                end;

              end;
            end;
          end;
end;
```

```
function TFParsingTable.Parsing: word;
var Action, Stack, Input  : string;
  Stacks    : array[1..64] of string[32];
  nStacks   : word;
  i         : word;
  CharInput : char;
  TopStack  : string;
  ParsingStep: word;
  noProduksi : word;
```

```
begin
  for i:= 1 to length(SGTest.cells[1,1]) do
    if pos(SGTest.cells[1,1][i], ItemUnik) = 0 then begin
      messagebox(0,'Simbol tidak dikenal !','Error',0);
      parsing:=0;
      exit;
    end;

  nStacks:=1;
  Stacks[1]:='0';
  Input:=eInput.Text+'$';
  Action:='';
  ParsingStep:=0;
  SGTest.RowCount:=1;
  nProdukAccepted:=0;
  noProduksi:=0;

  repeat
   inc(ParsingStep);
   if SGTest.RowCount=2 then SGTest.FixedRows:=1;
   SGTest.RowCount:=SGTest.RowCount+1;

   Stack:='';
   for i:=1 to nStacks do Stack:=Stack+Stacks[i];
   SGTest.cells[0,ParsingStep]:=Stack;
   SGTest.Cells[1,ParsingStep]:=Input;

  // Ambil Action
```

```
    TopStack:=Stacks[nStacks];

    CharInput:=Input[1];
    Action:=BarisPT[pos(CharInput, ItemUnik), strtoint(TopStack)+1];
    SGTest.Cells[2,ParsingStep]:=Action;

    if Action[1] = 's' then begin
      inc(nStacks);
      Stacks[nStacks]:=Input[1];
      inc(nStacks);
      Stacks[nStacks]:=copy(Action,2,length(Action)-1);
      delete(Input,1,1);
    end;

    if Action[1] = 'r' then begin
      noProduksi := strtoint(copy(Action,2,length(Action)-1));
      nStacks:=nStacks-(length(Produksi[noProduksi].Kanan)*2);
      inc(nStacks);
      Stacks[nStacks]:=Produksi[noProduksi].Kiri;
      inc(nStacks);
      Stacks[nStacks]:=BarisPT[pos(Stacks[nSTacks-1],      ItemUnik),strtoint(Stacks[nStacks-
2])+1];
      inc(nProdukAccepted);
      ProdukAccepted[nProdukAccepted]:=
noProduksi;
    end;

  until (Action='ac') OR (Action='-');
```

```
        if Action='ac' then Parsing:=noProduksi
          else Parsing:=0;
end;
```

LAMPIRAN 4: DAFTAR CONTOH PRODUCTION RULE

No	Pola Wazn	Keterangan	Grammer
1	فَعَلَ	fiil madhi	K → akbkck
2	يَفْعُلُ	fiil mudhorik	K → jkaqbmcm
3	فَعْلًا	mashdar	K → akbqcle
4	مَفْعَلًا	mashdar mim	K → hkaqbkcle
5	فَاعِلٌ	fa'il (pelaku)	K → akebocn
6	مَفْعُوْلٌ	maf'ul	K→ hkaqbmiqcn
7	أُفْعُلْ	fi'il amr	K → emaqbmcq
8	لَاتَفْعُلْ	fi'il nahi	K → gkefkaqbmcq
9	مَفْعَلٌ	isim makan / zaman	K → hkaqbkcn
10	مِفْعَلٌ	isim alat	K → hoaqbkcn

ABSTRAK

Pada aturan kata dalam Bahasa Arab, sebuah kata bisa berubah menjadi puluhan hingga ratusan bentuk, yang masing-masing memiliki arti sendiri-sendiri. Pola perubahan bentuk kata tersebut memiliki aturan-aturan yang jelas dan pasti. Untuk menghemat penyimpanan database kamus penerjemah, perlu dirancang bagaimana agar kamus tersebut cukup hanya menyimpan dalam bentuk dasar setiap kata yang bisa mengalami perubahan dalam bahasa Arab.

Metode parsing, bisa digunakan untuk mengenal pola kata Bahasa Arab dengan mengambil informasi *action* reduksi *production* mana saja yang telah sukses dilakukan untuk suatu masukan kata berbahasa Arab yang telah dikodekan menjadi token-token. Penggunaan database baik untuk *production rules* maupun kamus, memungkinkan aplikasi kamus Arab Indonesia untuk dikembangkan tanpa harus mengubah *source program*.

Ujicoba dilakukan dengan data sampling production rules yang mewakili perubahan kata berpola *wazn fa'ala* dengan *bina' shohih* serta beberapa kata bentuk dasar yang dijadikan data kamus. Dengan memasukkan suatu kata yang telah berubah bentuk dari bentuk asalnya –yang mana bentuk asalnya terdapat dalam kamus-, perangkat lunak ini mampu mendefinisikan kata tersebut serta memberikan keterangan-keterangan yang berhubungan dengan pola kata tersebut.