DOKUMEN BUKTI HAK CIPTA

Purwarupa (Kode Program) Mesin Compiler/Interpreter untuk Sistem Augmented Reality untuk Bahasa L(gFSA)

Pengusul Hak Cipta

Dr. Aslan Alwi, S.Si., M.Cs

Dr. Azhari, M.T

Dr. Suprapto, M.Kom

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2021

Purwarupa (Kode Program) Mesin Compiler/Interpreter untuk Sistem Augmented Reality untuk Bahasa L(gFSA)

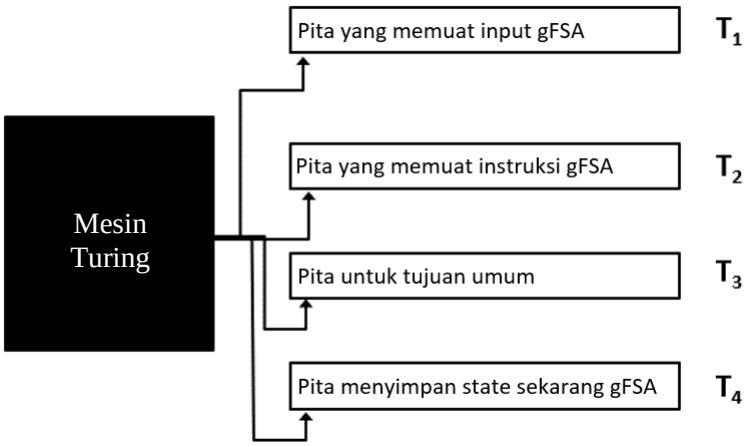
A. RINCIAN COMPILER/INTERPRETER

Bagian ini merinci *pseudocode* dan kode program sebenarnya untuk purwarupa *compiler-interpreter* untuk sistem Augmented Reality yang diusulkan sebagai bagian utama dari pengusulan hak cipta. Deksripsi arsitektur dan kode program ini dijelaskan secara rinci pada bagian klaim sebagai berikut.

PSEUDOCODE COMPILER-INTERPRETER

Misalkan *M* adalah sebuah mesin Turing dengan 4 pita yaitu T_1, T_2, T_3 dan T_4. T_1 menerima seluruh input berupa *word* input yang sebenarnya juga adalah string input (*word* penanda) dari gFSA yang disimulasikan. T_2 adalah pita tempat seluruh instruksi (fungsi transisi) gFSA ditulis. T_3 adalah sebuah pita yang disediakan untuk tujuan umum, seperti misal menulis salinan instruksi dari pita T_2. T_4 memuat *state* dari gFSA yang sedang aktif (*current state*).

Mesin Turing *M* digambarkan sebagaimana Gambar 6.1.



Gambar 6.1. Mesin Turing M untuk Mensimulasikan gFSA

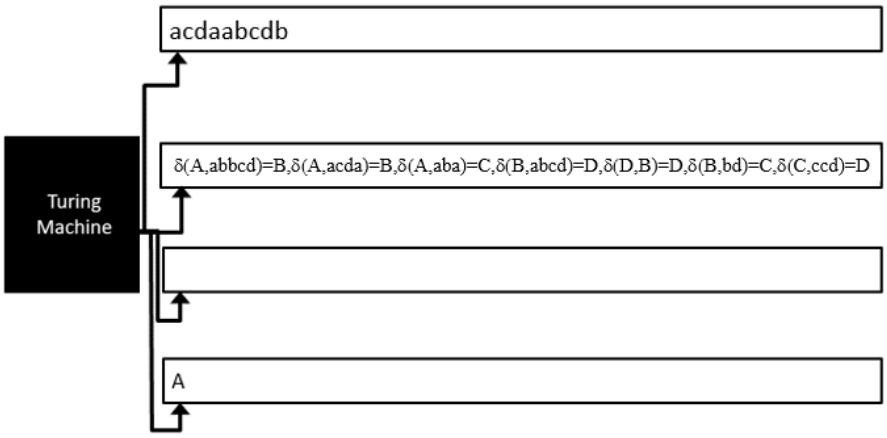
Dengan menggunakan contoh gFSA pada Gambar 5.10., operasi mesin Turing *M* adalah sebagai berikut:

a. Inisialisasi mesin Turing M:

- Step 1: *M* menulis start *state* pada sel pertama T_4. Dalam contoh ini, start *state* adalah *A*.
- Step 2: *M* menulis *word* input gFSA pada pita T_1, mulai dari posisi sel pertama. Misalkan *word* input gFSA adalah *acdaabcdb*.

- Step 3: *M* mengosongkan pita T_3.
- Step 4: *M* menulis semua instruksi gFSA pada pita T_2, mulai sel pertama. Instruksi gFSA pada contoh ini, Gambar 5.10, himpunan instruksi gFSA adalah: *{δ(A, abbcd) = B, δ(A, acda) = B, δ(A, aba) = C, δ(B, abcd) = D, δ(D, B) = D, δ(B, bd) = C, δ(C, ccd) = D}*.
- Step 5: *M* tempatkan semua *head* keempat pita (T_1, T_2, T_3 dan T_4) pada posisi sel pertama masing-masing.

Ilustrasi inisialisasi mesin Turing adalah seperti ditujukkan Gambar 6.2..



Gambar 6.2. Inisialisasi Mesin Turing M

a. Simulasi gFSA oleh M:

- Step 1: *M* membaca sel yang sedang ditunjuk oleh *head* pada pita T_1.

  Pada contoh ini, manakala baru saja terjadi inisialisasi, *head* menunjuk sel pertama pita pada karakter *a*.
- Step 2a: *M* membaca sel yang sedang ditunjuk oleh *head* pada pita T_4.

  Jika *state* yang ditunjuk adalah *state* FINISH dan sel yang ditunjuk oleh *head* pada pita T_1 adalah sel kosong maka mesin HALT.

  Jika *state* yang ditunjuk bukan *state* FINISH dan sel yang ditunjuk pada pita T_1 adalah sel kosong maka FAIL dan HALT. Dalam contoh ini, inisialisasi menunjuk sel pertama berisi *state A*.
- Step 2b: Jika pita T_3 tidak kosong, bandingkan *state* yang terbaca pada step 2a dan karakter yang terbaca pada step 1 dengan instruksi pertama pada T_3.

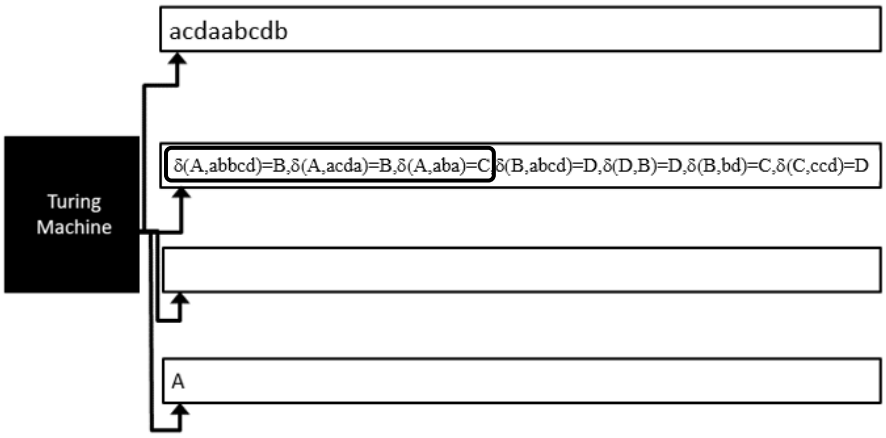
Jika *state* pada argumen instruksi pertama adalah sama dengan *state* yang sedang terbaca pada (*current state*) step 2a dan karakter pertama pada *word* dalam argumen instruksi pertama adalah sama dengan karakter yang terbaca pada step 1 maka lanjut ke step 5.

- Step 3: *M* mencari semua instruksi dalam pita T_2 yang karakter pertama input *word* nya adalah sama dengan karakter yang sedang ditunjuk oleh *head* pada pita T_1 dan *state* pada argumen instruksinya adalah sama dengan *state* yang sedang ditunjuk oleh *head* pada pita T_4.

  Jika tidak ketemu maka satupun instruksi maka M menjadi FAIL and HALT.

  Dalam contoh ini, setelah pencarian instruksi di T_2 dan membandingkannya dengan karakter *a* dan *state A* yang terbaca pada step 1 dan step 2a, diperoleh himpunan instruksi yang *match* sesuai dengan step 3 yaitu *{δ(A, abbcd) = B, δ(A, acda) = B, δ(A, aba) = C}*.

  Ilustrasi untuk step 3 digambarkan oleh Gambar 6.3:



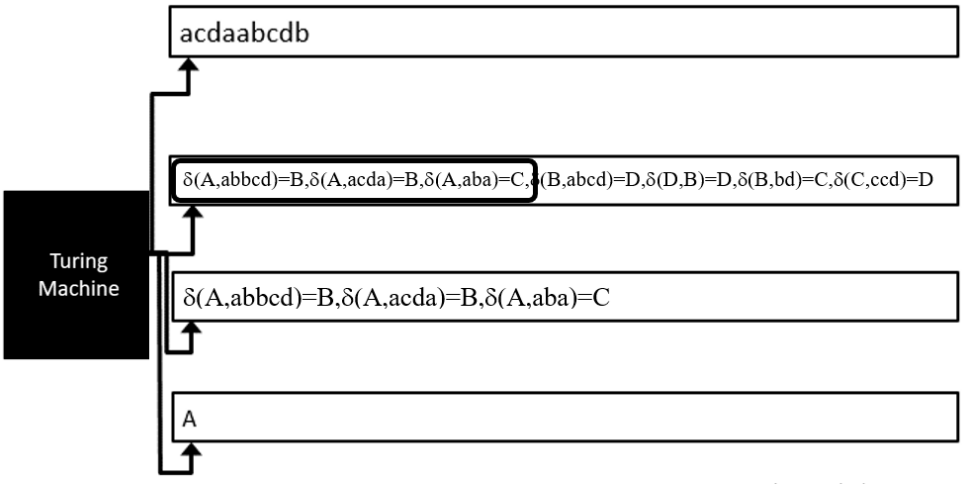
Gambar 6.3. Ilustrasi Step 3 Simulasi gFSA Mesin Turing M

- Step 4: *M* mengosongkan T_3 dan *M* menyalin semua instruksi yang ditemukan pada step 3 kemudian menulisnya secara terurut pada pita T_3 mulai sel pertama T_3. Urutan penulisan dimulai dari instruksi yang memiliki input *word* terpanjang kemudian *head* pita T_2 kembali ke posisi sel pertama dan *head* pada posisi T_3 juga direset ke sel pertama pita T_3.

  Pada contoh ini, semua instruksi pada pita T_3 telah selesai diurutkan, yaitu terurut sebagai berikut:

  δ(A, abbcd) = B, δ(A, acda) = B, δ(A, aba) = C.

  Ilustrasi untuk step 4 ini adalah sebagaimana Gambar 6.4.



Gambar 6.4. M Menulis Semua Instruksi yang Memenuhi δ(A, a ....) pada pita T_3

- Step 5: *M* mulai membaca instruksi-instruksi pada pita T_3 secara terurut mulai dari posisi sel pertama dan membandingkan isinya dengan isi T_1. (Catatan: pada pita T_3, *M* tidak lagi melakukan pencarian instruksi, tetapi membaca dalam urutan mulai instruksi pertama).

  Jika input *word* pada instruksi pertama tidak *match* dengan potongan *word* yang terbaca pada T_1 maka *head* berpindah ke instruksi kedua dan *head* pada pita T_1 direset posisinya pada karakter pertama potongan *word*. Input *word* pada instruksi kedua dibandingkan dengan potongan *word* yang terbaca pada pita T_1, jika tidak *match* maka *head* berpindah ke instruksi ketiga dan *head* pada pita T_1 direset posisinya pada karakter pertama potongan *word* demikian seterusnya sampai ditemukan bahwa input *word* instruksi *match* dengan potongan *word* yang terbaca pada pita T_1.

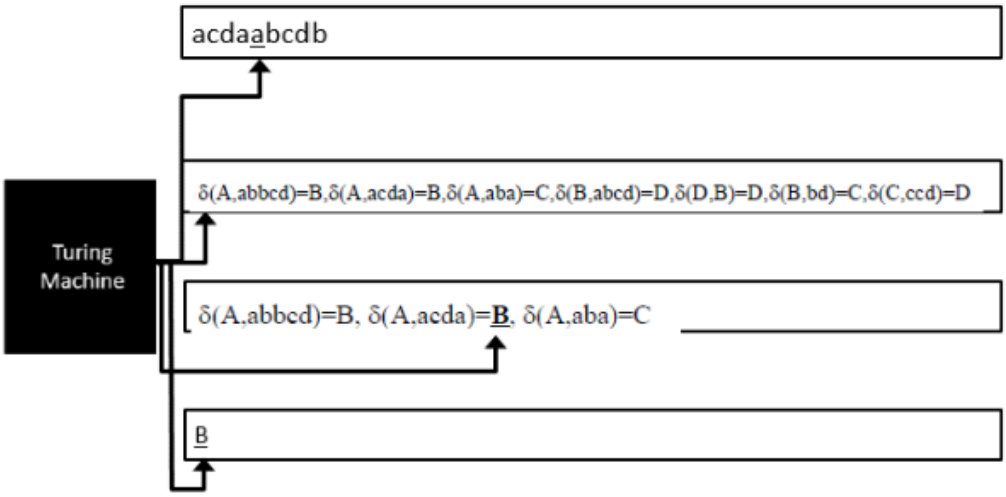
  Jika seluruh instruksi pada T_3 telah terbaca habis dan tak ada lagi instruksi yang *match* maka *M* menjadi FAIL and HALT.

  Jika instruksi yang *match* ditemukan maka instruksi tersebut dieksekusi sehingga transisi terjadi dan *state* berikut (*current state* berikut) adalah *state* berikut pada instruksi tersebut, selanjutnya *state* baru itu ditulis pada sel pertama pita T_4 menimpa isi sel pertama T_4 sebelumnya.

  Pada contoh sebelumnya, instruksi yang input *word*-nya *match* dengan potongan *word* pada pita T_1 adalah *δ(A, acda) = B maka M* mengeksekusi *δ(A, acda) = B* sehingga diperoleh

state berikut yaitu *state B*. *State B* kemudian ditulis pada sel pertama pita T_4 menimpa *state A* pada sel pertama pita T_4.

Ilustrasi untuk contoh ini adalah sebagaimana Gambar 6.5.



Gambar 6.5. Mesin Turing M Mengeksekusi δ(A, acda) = B

- Step 6: Ulangi step 1 sampai 5.
- SELESAI.

Sampai disini, algoritma ini secara intuitif menunjukkan bahwa di sana ada sebuah mesin Turing yang dapat mensimulasikan sebuah gFSA. Ini berarti bahwa secara intuitif mesin Turing itu dapat digunakan sebagai *interpreter* atau *compiler* untuk semua bahasa yang dibangun menggunakan gFSA, yaitu L(gFSA).

Konstruksi Pseuodecode secara Formal

Untuk usaha ini, definisi gFSA diberikan sebagaimana definisi 6.1 yang dikemukakan sebagai berikut:

> ***Definisi 6.1*** *Sebuah generalized-FSA (gFSA) adalah sebuah 6-tuple* $(Q, \Sigma, q_0, L_\Sigma, \delta, F)$ *dan:*
>
> Q *adalah himpunan berhingga state*
> Σ *adalah himpunan simbol*
> q_0 *adalah state awal*
> $L_\Sigma = \{ a_1a_2a_3...a_n \mid a_i \in \Sigma \}$
> $\delta: Q \times L_\Sigma \rightarrow Q$
> $F \subseteq Q$ *adalah himpunan state akhir*

Definisi gFSA pada definisi 6.1 kemudian diperumum ke dalam bentuk mesin Turing yang mengeksekusi *word* pada pita. Dalam kerja ini, pertama kali dibuat sebuah himpunan simbol untuk input dan pita untuk mesin Turing. Misalkan *C* adalah simbol alphabet pada keyboard (papan ketika komputer) dengan tanpa simbol blank, yaitu bahwa *C = {a, b, c, d, e, f, g, ..., y, z, A, B, C, D ..., Z, 1,2,3,4 ..., 9, + _) (*, &, %, $, £, @, ...}*, andaikan simbol blank adalah ∅. Definisi 6.2. menunjukkan definisi formal mesin Turing yang dimaksud.

> ***Definisi 6.2*** *Sebuah mesin Turing, sebut sebagai MW, yang mengeksekusi sebuah word didefinisikan sebagai 7-tuple* $(Q, \Sigma, I, L_I, q_0, \delta, F)$ *dan:*
>
> Q *adalah himpunan state*
> $F \subseteq Q$ *adalah himpunan state accept*
> q_0 *adalah state start*
> $I = C$ *alphabet*
> $\Sigma = C \cup \{\varnothing\}$ *adalah himpunan simbol pita*
> $L_I = \{ a_1a_2a_3...a_g \mid a_i \in I , i=1,2,3,...,g\}$
> $\delta: Q \times L_I \rightarrow Q \times L_I \times \{-m,0,n\}$
>
> *MW hanya memiliki satu pita, membaca dari kiri ke kanan dan menulis dari kiri ke kanan.*

Kerja daripada mesin transisi *MW* dapat dijelaskan dengan cara misalkan sebuah transisi *δ(A, abcd) = (B, cda, -2)* adalah dibaca sebagai paragraf berikut.

Head pita membaca *word abcd* pada pita sehingga *MW* mengubah *state* dari *state A* ke *state B* kemudian *head* direset ke posisi awal dari *word abcd* dan mulai menulis *word cda* dan menimpa *word abcd* kemudian *head* berpindah 2 sel ke kiri.

Andaikan sebuah transisi *δ(A, aaac) = (B, aadcda, 2),* transisi ini dibaca sebagai paragraf berikut.

Head membaca *word aaac* pada pita kemudian *MW* mengubah *state* dari A ke B kemudian *head* mereset posisinya pada karakter awal *word aaac* dan mulai menulis *aadcda* dan menimpa semua *aaac* kemudian *head* berpindah 2 sel ke kanan.

MW didefinisikan sebagai $(Q, \Sigma, I, L_I, q_0, \delta, F)$ kemudian transisi dalam *MW* didefinisikan secara umum dalam bentuk $\delta(A_i, x) = (A_j, x, 1)$ dan *x* adalah sebuah *word* maka *MW* adalah sebuah gFSA yang memenuhi definisi 6.1. Akan tetapi, mungkin gFSA memiliki satu atau lebih simbol diluar alphabet *C*. Untuk mengatasinya, *x* dienkoding ke *word* dan *word* itu adalah menggunakan simbol-simbol *C* semata sehingga transisi menjadi $\delta(A_i, encode(x)) = (A_j, encode(x), 1)$.

Encoding untuk MW dan inputnya

MW dan inputnya dienkoding ke dalam alphabet *C*. Enkoding ini adalah sebuah deskripsi *MW* dan juga deskripsi input *MW* ke dalam bentuk string alphabet *C* untuk diumpankan sebagai input pada sebuah mesin Turing (dalam hal ini mesin Turing *M*). Proses enkoding itu adalah sebagai paragraf berikut:

Semua *state* dari *MW* dipetakan ke bilangan bulat 0, 1, ... *k*., pemetaan ini dilakukan secara *bijective* dan *state* start dipetakan sebagai $q_0 \equiv 0$ dan *state accept* dipetakan sebagai *f*, *f* + 1, ... *k* for $0 < f \leq k$.

Ini berarti jika hasil pemetaan himpunan state Q = {0,1,2, 3, ...k} maka berlaku sifat $q_0 = 0$ and F = {f, f + 1, ...k} $k \geq 0$, $f \leq k$.

Word yang menjadi input bagi *MW* adalah dienkoding ke dalam string yang dibangun oleh alphabet $C \cup \{\varnothing\}$ juga secara *bijective* sehingga diperoleh sebuah bahasa yang telah dienkoding yaitu $L_{encode} = encode(L_I) = \{b_1 b_2 b_3 \ldots b_g \mid b_i = encode(a_i), a_i \in I, b_i \in C \cup \{\varnothing\}\ i = 1,2,3, \ldots, g\}$.

Fungsi transisi diekspresikan sebagai δ-function: δ(q, s) = (q ', s', d) dan s, s' ∈ L_{encode}, $0 \leq q \leq f$ dan d∈ {-m, 0, n}, $0 \leq q' \leq k$ dan m dan n menyatakan banyaknya pergeseran *head* di atas pita, jika negatif, misal -m maka bergeser ke kiri, jika positif, misal n maka bergeser ke kanan.

Panjang dari *word* $s \in L_{encode}$ adalah |*s*|.

Selanjutnya, *δ-function* yang diumpankan sebagai input ke dalam mesin Turing adalah ditulis dalam bentuk 6-tuples *(q, s, |s|, q ', s', d)*.

Selanjutnya, penulisan deskripsi *MW* ke dalam pita mesin Turing adalah dalam urutan *k, f, t_1, t_2, t_3,* ... dan *t_1, t_2, t_3,*... menyatakan barisan instruksi yang sudah dienkoding, dan *k* adalah jumlah *state*, *f* adalah *state accept* pertama dan t_i adalah *(q, s, | s |, q ', s', d)*. Ini berarti *encoding(MW) = k, f, (q, s, |s|, q ', s', d), ..., (q, s, |s|, q ', s', d)* dan seterusnya hingga seluruh instruksi *MW* tertuliskan, dan n, f, q, q', d adalah bilangan-bilangan bulat. $0 \leq f, q < f, q' \leq k$, d ∈ {-m, 0, n} dan s, s' ∈ L_{encode}.

Konstruksi UTM untuk MW

Pada bagian ini akan dikonstruksikan sebuah mesin Turing universal (UTM) yang sebenarnya merupakan versi mesin Turing *M* yang disuplai enkoding dari *MW* dan input *MW*. Pada versi ini terdapat modifikasi algoritma versi intuitif sebelumnya. Mesin Turing *M* versi ini adalah juga seperti Gambar 6.1, yaitu mesin dengan 4 pita, T_1, T_2, T_3 dan T_4. T_1 adalah sebuah pita dengan input *MW*, T_2 adalah pita yang deskripsi *MW* di dalam bentuk *encoding(MW)* diletakkan, T_3 adalah

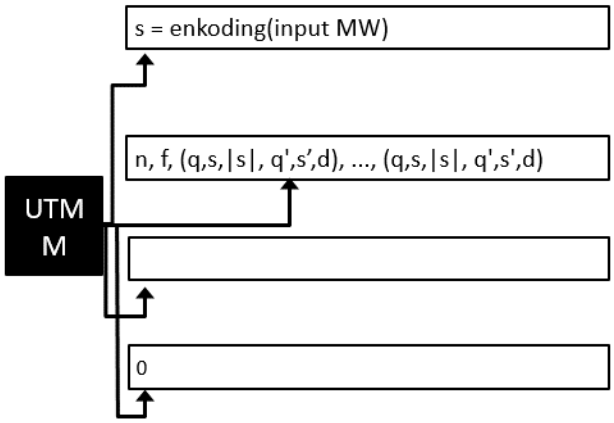
pita bebas yang ditujukan untuk tujuan umum dan pita T_4 adalah pita yang *state* yang telah dieksekusi ditulis (*current state*).

Operasi-operasi mesin Turing *M* untuk *MW* yang telah dienkoding adalah sebagai berikut:

a. Tahapan Inisialisasi:

- Step 1: *M* menulis 0 pada sel pertama pita T_4. (0 adalah *state* start)
- Step 2: *M* menulis encoding *word* input *MW* pada pita T_1, mulai dari sel pertama.
- Step 3: *M* mengosongkan pita T_3.
- Step 4: *M* menulis seluruh deskripsi mesin Turing *MW* yaitu *encoding(MW)* pada pita T_2, mulai pada sel pertama.
- Step 5: *M* menempatkan seluruh *head* keempat pita, T_1, T_2, T_3 dan T_4, pada sel pertama tiap pita.

Ilustrasi langkah inisiliasi ini disajikan sebagaimana Gambar 6.6.



Gambar 6.6. Inisialisasi mesin Turing *M* sebagai UTM

a. Tahapan simulasi mesin Turing *MW*:

- Step 1: *M* membaca sel yang ditunjuk oleh *head* pada pada pita T_4.

  Jika $q \geq f$ dan head pada pita T_1 telah menunjuk sel yang melampaui posisi sel-sel input maka HALT dan Instruksi mencapai *state accept*

  Jika $q < f$ maka lanjut ke step 2a.
- Step 2a: *M* membaca sel yang sedang ditunjuk oleh *head* pada pita T_1.
- Step 2b: Jika T_3 tidak kosong, bandingkan *state* yang terbaca pada step 1 dan karakter yang terbaca pada step 2a dengan instruksi pertama pada pita T_3.

Jika *state* pada argumen instruksi pertama adalah sama dengan *current state* yang terbaca pada pita step 1 dan karakter pertama input *word* argumen instruksi pertama adalah sama dengan karakter yang terbaca pada step 2a maka lanjut ke step 5.

- Step 3: *M* mencari semua instruksi pada pita T_2 yang karakter pertama *word* inputnya adalah sama dengan yang sedang ditunjuk oleh *head* pada pita T_1 dan *state* yang sedang aktif (*current state*) adalah *state* yang sama dengan yang sedang ditunjuk oleh *head* pada pita T_4.

  Jika tidak ada yang *match* maka mesin *M* menjadi HALT dan FAIL.

  Jika ketemu instruksi yang match maka lanjut ke step 4.

- Step 4: *M* mengosongkan pita T_3 dan *M* menyalin semua instruksi yang ditemukan *match* pada step 3 dan kemudian menulisnya secara berurut (sekuensial) pada pita T_3. Penulisan berurut dimulai dengan instruksi yang memiliki panjang *word* input terbesar dengan membaca nilai |*s*| pada *(q, s, |s|, q', s', d)*.

  Selanjutnya, *head* pada pita T_2 direset ke posisi sel pertama pada T_2 dan *head* pada pita T_3 juga direset ke posisi sel pertama pita T_3.

- Step 5: *M* mulai menulisi instruksi pertama sesuai urutan. Pembacaan dimulai pada instruksi pertama sesuai urutan dalam pita T_3 dan membandingkan input *word* instruksi pertama dengan potongan *word* yang terbaca pada pita T_1.

  Catatan: Pada pita T_3, *M* tidak lagi melakukan pencarian, tetapi hanya membaca dalam urutan mulai instruksi pertama dalam urutan tersebut.

  Jika input *word* pada instruksi pertama tidak *match* dengan potongan input *word* yang sedang terbaca pada pita T_1 maka *head* pada pita T_3 berpindah ke instruksi kedua dalam urutan kemudian melakukan perbandingan antara input *word* pada instruksi kedua dengan input *word* yang terbaca pada pita T_1. Jika tidak *match* maka *head* pada pita T_3 ke instruksi ketiga dalam urutan demikian seterusnya hingga seluruh instruksi habis terbaca atau *head* menemukan instruksi yang sesuai.

  Jika instruksi pada T_3 telah terbaca habis dan tidak ditemukan instruksi yang *match* maka mesin Turing *M* mengalami FAIL dan HALT.

  Jika ditemukan instruksi yang *match*, misal instruksi tersebut *(q, s, |s|, q ', s', d)* yang *word s match* dengan *word* yang terbaca pada pita T_1 maka transisi terjadi dari *state q* ke *state q'* dan hasil transisi ini yaitu *q'* ditulis pada sel pertama pita T_4 menimpa *current state* sebelumnya. *Current state* bertransformasi menjadi *state q'*.

Head kemudian direset ke karakter pertama *word s* yang terbaca pada pita T_1 kemudian *head* mulai menulis dan menimpa *s* dengan menulis *s'* sampai selesai.

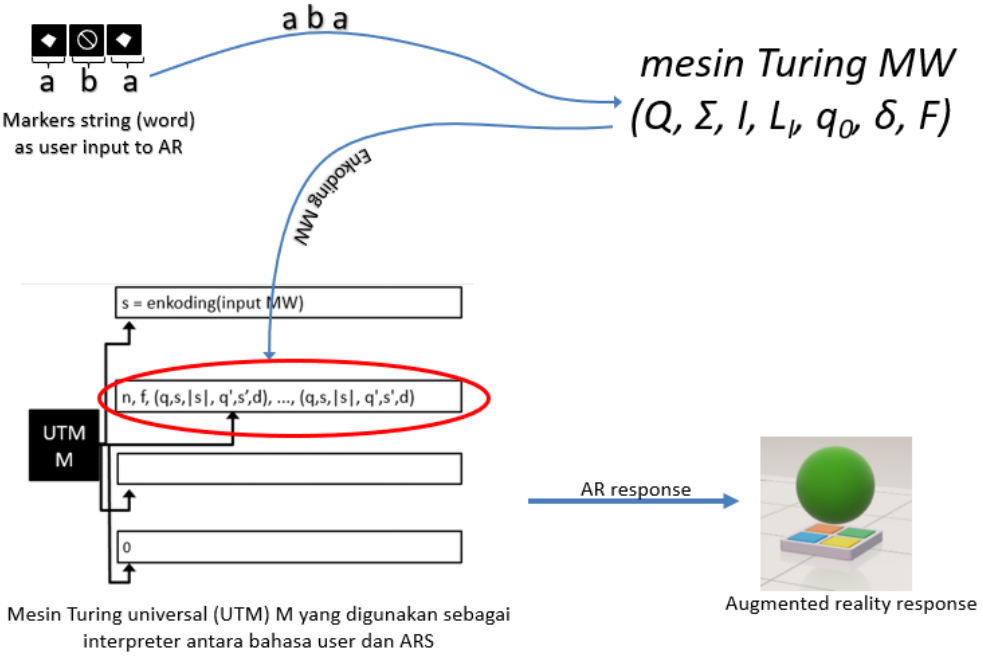
Setelah *s'* selesai ditulis maka *head* berpindah sejauh *d* sel mulai tepat setelah akhir dari *word* s'.

- Step 6: Ulangi step 1 sampai step 5.

Selanjutnya, secara umum ukuran *word* pada setiap input *word MW* dibuat menjadi berukuran 1, yaitu setiap *word s* memiliki ukuran $|s| = 1$ maka *MW* menjadi sebuah bentuk mesin Turing standar, ini secara trivial membuktikan bahwa mesin Turing *M* adalah *Turing complete*, yaitu sebuah *universal Turing machine* (UTM).

Model Interpreter

Mesin Turing *M* dapat menjadi sebuah *interpreter* bagi semua bahasa yang dibangun menggunakan gFSA atau *MW*, yaitu sebagai *interpreter* bagi bahasa L(MW) atau L(gFSA). Gambar 6.7. memperlihatkan ilustrasi bagaimana mesin Turing *M* menjadi *interpreter* bagi bahasa pengguna yang dibangun menggunakan *MW*. Karena *MW* adalah perumuman dari gFSA, yaitu bahwa $L(gFSA) \subseteq L(MW)$ maka *interpreter M* adalah juga *interpreter* bagi bahasa yang dibangun menggunakan gFSA.



Gambar 6.7. Bahasa Pengguna yang Dibangun Menggunakan MW Ditengahi oleh Interpreter M (Sumber gambar: pribadi)

Model Kompiler

Mesin Turing *M* dapat menjadi sebuah *compiler* dalam bentuk yang sederhana, yaitu sebagai mesin lexer (mesin yang memeriksa token) dan sebagai mesin parser (mesin yang memeriksa tata bahasa, tetapi karena bahasa pengguna tidak menggunakan gramatika, tetapi mesin Turing *MW* atau gFSA maka pemeriksaan sintaks adalah berarti pemeriksaan automata gFSA atau mesin Turing *MW*). Dalam konteks ini, buat dua buah mesin *M*, pertama sebagai pemeriksa token, kedua sebagai pemeriksa automata).

Sebagai pemeriksa token, input *word* dari pengguna dapat dilihat sebagai barisan token, yaitu $s=s_1s_2s_3...s_h=\alpha(w_1)\alpha(w_1)\alpha(w_1)...\alpha(w_h)=\alpha(w)$ maka $w_1, w_2, w_3,..., w_h$ adalah token-token dan $s_1, s_2, s_3,..., s_h$ adalah enkoding dari token-token tersebut. Definisi 6.2. tentang *MW* menunjukkan bahwa $w_1, w_2, w_3,..., w_h \in L_I$ dan $L_I = \{a_1a_2a_3...a_g \mid a_i \in I\ ,\ i=1,2,3,...,g\}$.

Jika L_I hanya semata sebuah himpunan *word* yang berhingga, tidak memiliki gramatika maka di sana dapat dibuat pemetaan yang dapat digunakan untuk memeriksa token, cukup dengan memeriksa apakah tiap-tiap token adalah elemen dari L_I atau di sana dapat dibuat sebuah FSA yang bisa digunakan untuk memeriksa setiap token kemudian himpunan instruksi atau transisi dari FSA di enkoding masuk ke dalam mesin Turing *M* sehingga mesin Turing *M* dapat bertindak sebagai mesin lexer (pemeriksa token) menggunakan enkoding FSA.

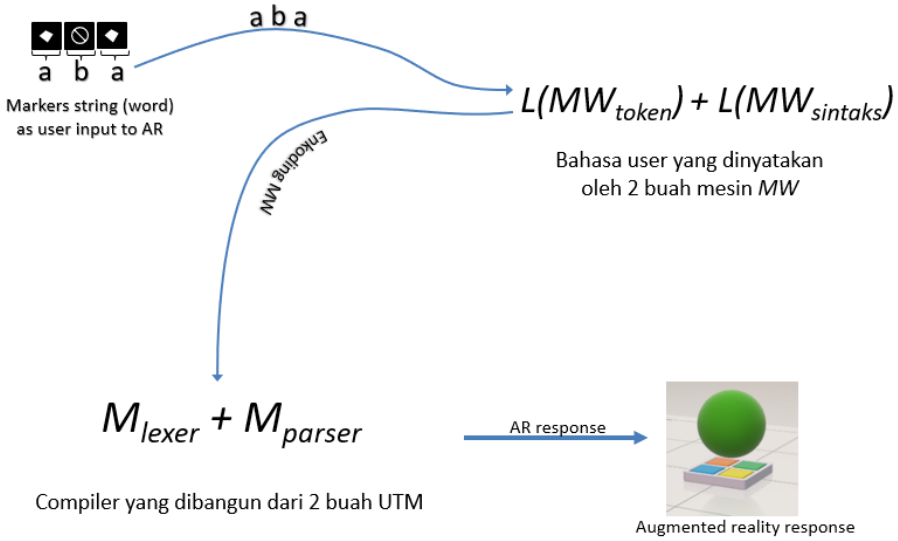
Akan tetapi, jika L_I memiliki gramatika, misal gramatika G maka di sana ada sebuah automata atau mesin Turing yang sepadan dengan L_I. Akan tetapi, telah ditunjukkan pada teorema 6.2 bahwa mesin Turing *MW* dapat dinyatakan sebagai mesin Turing standar sehingga mestilah L_I dapat dinyatakan oleh sebuah mesin *MW*. Sebut mesin *MW* itu sebagai MW_{token} sehingga $L_I=L(G)=L(MW_{token})$. Ini berarti sebuah mesin lexer menggunakan mesin *M* dapat dibuat untuk memeriksa token-token yang dihimpun dalam bahasa $L(MW_{token})$. Mesin *M* yang bertindak sebagai mesin lexer ini ditulis sebagai M_{lexer}.

Secara keseluruhan bahasa pengguna sebagai bahasa yang sintaksnya dinyatakan oleh sebuah automata adalah sebuah mesin *MW*. Bahasa ini disebut sebagai $MW_{sintaks}$ sehingga bahasa pengguna itu dapat ditulis sebagai $L(MW_{sintaks})$, selanjutnya sebuah mesin *M* lain, mesin ini disebuat sebagai mesin M_{parser} bertugas sebagai mesin Turing yang memeriksa automata (sintaks) dari bahasa tersebut.

Secara keseluruhan, sebuah *compiler* bagi bahasa pengguna untuk berinteraksi dengan sebuah ARS adalah sebuah susunan $M_{lexer} + M_{parser}$, yaitu dapat ditulis sebagai:

Compiler = $M_{lexer} + M_{parser}$ (6.2)

Gambar 6.8. memberikan ilustrasi bagaimana *compiler* tersebut menjadi penengah antara bahasa pengguna dengan sebuah ARS.



Gambar 6.8. Bahasa Pengguna yang Diterima oleh Mesin MW_{token} dan $MW_{sintaks}$ Ditengahi oleh Kompiler ($M_{lexer} + M_{parser}$)

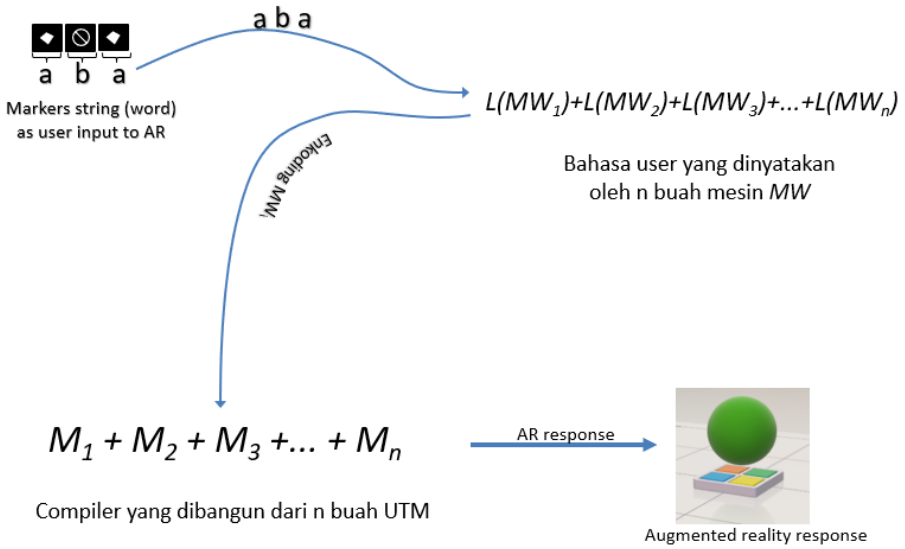
Akan tetapi, himpunan respon ARS dapat dianggap sebagai sebuah bahasa, walaupun dia hanya semata himpunan *word* (*word* yang mungkin mewakili ekspresi fungsi-fungsi *executable*) yang mungkin tanpa gramatika dan karena mesin Turing *M* adalah juga sebuah UTM (teorema 6.2.) maka mestilah sebuah algoritma untuk membangkitkan serangkaian respon ARS adalah dapat dinyatakan oleh sebuah mesin Turing *M* (*Church-Turing Thesis* bahwa setiap algoritma selalu dapat dinyatakan oleh sebuah mesin Turing, misal mesin Turing MW_A maka mestilah *M* sebagai UTM adalah dapat mensimulasikan MW_A. Sebut mesin *M* ini sebagai $M_{generator}$) sehingga sebuah *compiler* pada model persamaan (6.2) dapat diperluas menjadi:

$$Compiler = M_{lexer} + M_{parser} + M_{generator} \quad (6.3)$$

Secara umum, jika kompleksitas *compiler* bertambah, misal memerlukan n buah mesin Turing *M* maka sebuah arsitektur *compiler* secara umum dapat dinyatakan sebagai:

$Compiler = M_1 + M_2 + M_3 + ... + M_n$ dan M_i mesin Turing *M* ke-i. (6.4)

Generalisasi *compiler* ini diilustrasikan oleh Gambar 6.9, yaitu sebuah *compiler* antara pengguna dan ARS. Model ini sebenarnya sangat *general* karena dapat diperluas untuk sebarang sistem, tak terbatas kepada sistem *augmented reality*.



Gambar 6.9. Generalisasi Kompiler untuk Interaksi Pengguna-ARS

PURWARUPA BERUPA KODE PROGRAM

Bagian ini menyatakan purwarupa dalam bentuk kode program. Kode lengkap demonstrasi compiler dapat dilihat pada https://github.com/elangbijak4/compiler-automata. Kode program yang ditulis disini adalah kode utama yanga menyatakan purwarupa compiler atau interpreter bagi bahasa L(gFSA). Akan tetapi, telah menunjukkan fondasi untuk membuatnya menjadi *compiler* yang kompleks sebagai model *compiler* yang dideksripsikan di atas.

Kode program:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text.RegularExpressions;

namespace compiler
{
    class UTM
    {
        private ArrayList pita1 = new ArrayList();
        private ArrayList pita2 = new ArrayList();
        private ArrayList pita3 = new ArrayList();
        private ArrayList pita4 = new ArrayList();
```

```
private int head_pita1;
private int head_pita2;
private int head_pita3;
private int head_pita4;
private string state_UTM;
private string state_user;
private String state_awal;
private String state_akhir;

//Himpunan variabel untuk mengakses pita
private string bufferT1;
private string[] bufferT2;
private string[] bufferT3;
private int bufferT4;

//Himpunan variabel untuk mengakses instruksi individual
private string komponen_state_sekarang; //0
private string komponen_input; //1
private int komponen_ukuran_input; //2
private string komponen_state_berikut; //3
private string komponen_output; //4
private int komponen_geseran_head; //5


public string ConvertArrayListToString(ArrayList bufferarray)
{
    string buffer4 = "";
    foreach (char i in bufferarray)
    {
        buffer4 = buffer4 + i;
    }

    return buffer4;
}

public string ConvertArrayListToString2(ArrayList bufferarray)
{
    string buffer4 = "";
    for(int i=0; i < bufferarray.Count; i++)
    {
        buffer4 = buffer4 + bufferarray[i];
    }
```

```
            return buffer4;
        }

        public string ConvertArrayListToString_bykonektor(ArrayList bufferarray,
char konektor)
        {
            string buffer4 = "";
            for (int i = 0; i < bufferarray.Count; i++)
            {
                buffer4 = buffer4 + bufferarray[i] + konektor;
            }
            buffer4 = buffer4.TrimEnd(konektor);
            return buffer4;
        }

        public void Inisialisasi()
        {
            /* Tugas method ini:
             * (1).
             * Mempersiapkan seluruh pita dengan cara memanggil kelas register
bahasa
             * Sehingga user bisa memasukkan seluruh informasi tentang bahasanya.
             * Setelah user selesai memasukkan info bahasa, maka method ini
mengambil nilai2 pada data class register bahasa untuk dimasukkan ke
             * pita-pita UTM, yaitu:
             * Informasi state dan dipindahkan ke array state_user.
             * Informasi state awal lalu memindahkannya ke array string_awal dan
menyalin ke array pita4
             * Informasi state akhir (accept states) lalu memindahkannya ke array
state_akhir.
             * Informasi enkoding instruksi-instruksi mesin Turing lalu
memindahkannya ke array pita2
             * .
             * Setelah itu, Semua variabel head_pita1, head_pita2, head_pita3,
head_pita4 direset ke nilai 0.
             *
             * (2).
             * Menginisiasi state UTM dengan state 0 atau 1 untuk digunakan method
simulasi.
             * Dalam hal ini, variabel state_UTM direset ke 0.
             *
             * (3).
             * Menjalankan method input untuk menerima input dari user.
```

```
             * Setelah user memasukkan maka method ini menyalin semua input ke
array pita1.
             *
             * (4).
             * Method menginisiasi objek untuk class respon.
             *
             * (5).
             * Method dialihkan tugasnya ke method simulasi.
             */

            //Memanggil class bahasa untuk memnita user memasukkan bahasa yang
ingin digunakan:
            Language_profil profil;
            profil = new Language_profil();
            profil.Register_bahasa();

            //Inisialisasi pita
            state_user = profil.state;
            state_awal = profil.state_awal;
            state_akhir = profil.state_accept;
            pita1 = null;
            pita2 = profil.enkoding_ut_pita2;
            pita3 = null;
            pita4.Add(profil.automata[4]);

            //Reset semua head pita
            head_pita1 = 0;
            head_pita2 = 0;
            head_pita3 = 0;
            head_pita4 = 0;

            //Reset pointer state UTM
            state_UTM = "1";

            //Gunakan kelas inputan untuk menerima input user:
            Console.Clear();
            Console.WriteLine("Mesin telah disiapkan untuk menerima input anda.
Silahkan memasukkan input di bawah ini:");
            Inputclass input_user;
            input_user = new Inputclass();
            input_user.Set_input();
            input_user.Enkoding();
            pita1 = input_user.Enkodingkalimat;
```

```
            //Proses inisiasi kelas respon belum dapat dilakukan karena Kelasnya
belum ada.



            //Konsol sementara untuk melihat hasil kerja kelas Inisialisasi:
            Console.WriteLine("\nHasil inisiasliasi pita:");
            Console.WriteLine("state_user = {0}", state_user);
            Console.WriteLine("state_awal = {0}", state_awal);
            Console.WriteLine("state_akhir = {0}", state_akhir);
            if (pita1 == null) Console.WriteLine("pita1 = null"); else
Console.WriteLine("pita1 = {0}", ConvertArrayListToString(pita1));
            Console.WriteLine("pita2 = {0}", ConvertArrayListToString2(pita2));
            if (pita3 == null) Console.WriteLine("pita3 = null"); else
Console.WriteLine("pita3 = not null");
            Console.WriteLine("pita4[0] = {0}", pita4[0]);
            Console.WriteLine("state_user = {0}", state_user);

            Console.WriteLine("\nHasil reset head semua pita:");
            Console.WriteLine("head_pita1 = {0}", head_pita1);
            Console.WriteLine("head_pita2 = {0}", head_pita2);
            Console.WriteLine("head_pita3 = {0}", head_pita3);
            Console.WriteLine("head_pita4 = {0}", head_pita4);

            Console.WriteLine("\nHasil inisialisasi counter state UTM:");
            Console.WriteLine("state_UTM = {0}", state_UTM);

            Console.WriteLine("\nInShaa Allah hasil selanjutnya adalah pekerjaan
method simulasi:");
            Console.ReadLine();
        }

        private string[] Pembaca_instruksi(ArrayList pita_instruksi, int nomor_sel)
        {
            string buffer_instruksi;
            string[] instruksi;

            buffer_instruksi = pita_instruksi[nomor_sel].ToString().TrimEnd(';');
            instruksi = buffer_instruksi.Split(',');



            return instruksi;
```

```
        }

        public ArrayList Penghilang_item_ganda(ArrayList arrayList)
        {
            ArrayList arrayList2 = new ArrayList();
            string SarrayList = ConvertArrayListToString_bykonektor(arrayList,
';');
            string[] SarrayList2 = SarrayList.Split(';');
            SarrayList2 = SarrayList2.Distinct().ToArray();
            for(int i=0; i< SarrayList2.Length; i++)
arrayList2.Add(SarrayList2[i]);

            return arrayList2;
        }

        public string ConvertStringArrayToString(string[] word)
        {
            string word_string = "";
            foreach (string i in word)
            {
                word_string = word_string + i;
            }
            return word_string;
        }

        public string ConvertIntArrayToString(int[] word)
        {
            string word_string = "";
            foreach (int i in word)
            {
                word_string = word_string + i;
            }
            return word_string;
        }

        public string ConvertStringArrayToString_bykonektor(string[] word, char
konektor)
        {
            string word_string = "";
            foreach (string i in word)
            {
                word_string = word_string + i + konektor;
            }
```

```
            word_string = word_string.TrimEnd(konektor);
            return word_string;
        }

        public void Cek_point_trace(string point, string beginend)
        {
            Console.WriteLine("\nIni di {0} Proses sekarang di state_UTM {1}\n",
beginend, point);
        }

        public void Simulasi()
        {
            /* Tugas method ini adalah:
             * (1).
             * Big picture method ini adalah berupa switch.
             * Nilai switch berubah menurut perubahan nilai pada variabel
state_UTM.
             * Setiap nilai state_UTM menggambarkan prosedur yang harus dikerjakan
kepada pita-pita.
             * Setiap blok dalam switch adalah blok kode.
             * Blok kode secara umum bisa merubah nilai state_UTM, sehingga dapat
menjalankan prosedur berikut.
             * Variabel State_UTM adalah analogi dengan program counter pada
prosesor komputer biasa.
             * .
             * Setiap blok kode dalam switch adalah algoritma tersendiri.
             * .
             * (2).
             * Setiap switch mencapai HALT pada state accept, maka rencananya dia
memanggil method di dalam class respon
             * untuk menerjemahkan hasil dari state accept tersebut.
             * class respon dijalankan objeknya diawal agar tidak harus selalu
menginisiasi objeknya, tetapi cukup sekali.
             * .
             * (3).
             * Selesai.
             */

            Boolean HALT = false;
            string state_accept_tercapai;
            ArrayList buffer_ut_disalin_keT3 = new ArrayList();

            //Baca terlebih dulu state_UTM
```

```
            do
            {
                Cek_point_trace(state_UTM, "begin do");

                switch (state_UTM)
                {
                    case "1":
                        Cek_point_trace(state_UTM, "begin");
                        //Cek apakah state pada pita4 adalah state accept atau
bukan?
                        bufferT4 = int.Parse(pita4[head_pita4].ToString());
                        if ((bufferT4 > (int)pita2[1]) & (head_pita1>pita1.Count-
1))
                        {
                            HALT = true;
                            state_accept_tercapai =
state_user[int.Parse(pita4[head_pita4].ToString())].ToString();
                            state_UTM = "HALT";
                            Console.WriteLine("State accept telah tercapai, yaitu
state; {0} ", bufferT4);
                            //LAPORAN:
                            Console.WriteLine("LAPORAN = Nilai bufferT4:{0}  dan
(int)pita2[1]:{1} dan state_UTM:{2} dan HALT:{3} dan state_accept_tercapai:{4}",
bufferT4, (int)pita2[1], state_UTM, HALT, state_accept_tercapai);
                        }
                        else state_UTM = "2a";

                        //LAPORAN:
                        Console.WriteLine("LAPORAN = Nilai bufferT4:{0}  dan
(int)pita2[1]:{1} dan state_UTM:{2}", bufferT4, (int)pita2[1], state_UTM);
                        Cek_point_trace(state_UTM,"end");
                        break;
                    case "2a":
                        Cek_point_trace(state_UTM, "begin");
                        if (head_pita1>pita1.Count-1)
                        {
                            state_UTM = "HALT";
                            Console.WriteLine("Head pita 1 telah menunjuk sel yang
melampaui panjang input");
                            //LAPORAN:
                            Console.WriteLine("LAPORAN = Nilai head_pita1:{0}  dan
pita1.Count-1:{1} dan state_UTM:{2}", head_pita1, pita1.Count - 1, state_UTM);
                        }
```

```
                        else
                        {
                            bufferT1 = pita1[head_pita1].ToString();
                            state_UTM = "2b";

                            //LAPORAN:
                            Console.WriteLine("LAPORAN = Nilai bufferT1:{0}
state_UTM:{1}", bufferT1, state_UTM);
                        }
                        Cek_point_trace(state_UTM, "end");
                        break;
                    case "2b":
                        Cek_point_trace(state_UTM, "begin");
                        if (pita3 != null)
                        {
                            bufferT3 = Pembaca_instruksi(pita3, head_pita3);
                            komponen_state_sekarang = bufferT3[0];
                            komponen_input = bufferT3[1];
                            bufferT4 = int.Parse(pita4[head_pita4].ToString());

                            //LAPORAN:
                            Console.WriteLine("LAPORAN (pita3 != null) = Nilai
bufferT3:{0}  dan komponen_state_sekarang:{1} dan state_UTM:{2} dan
komponen_input:{3} dan bufferT4:{4}", bufferT3, komponen_state_sekarang, state_UTM,
komponen_input, bufferT4);
                            //Di sini ada pertanyaan, bagaimana jika tidak?
                            Console.WriteLine("\nCEK MANAKALA PITA3 TIDAK NULL
KARENA PROSES BERULANG");
                            Console.WriteLine("bufferT4.ToString()={0}",
bufferT4.ToString());
                            Console.WriteLine("komponen_state_sekarang = {0}",
komponen_state_sekarang);

Console.WriteLine("pita1[head_pita1].ToString().ToArray()[0] = {0}",
pita1[head_pita1].ToString().ToArray()[0]);
                            Console.WriteLine("komponen_input[0] = {0} \n",
komponen_input[0]);

                            if ((bufferT4.ToString() == komponen_state_sekarang) &
(pita1[head_pita1].ToString().ToArray()[0] == komponen_input[0])) state_UTM = "5";
else
                            {
```

```
                                state_UTM = "3";
                                pita3 = null;
                                pita3 = new ArrayList();
                                buffer_ut_disalin_keT3 = null;
                                buffer_ut_disalin_keT3 = new ArrayList();

                                head_pita3 = 0;
                                //LAPORAN:
                                Console.WriteLine("LAPORAN (bufferT4.ToString() ==
komponen_state_sekarang) & = Nilai pita3:{0}  dan head_pita3:{1} dan
state_UTM:{2}", pita3, head_pita3, state_UTM);

                            }
                        }
                        else
                        {
                            state_UTM = "3";
                            buffer_ut_disalin_keT3 = null;
                            buffer_ut_disalin_keT3 = new ArrayList();

                            //LAPORAN
                            Console.WriteLine("LAPORAN else = state_UTM:{0}",
state_UTM);
                        }

                        Cek_point_trace(state_UTM, "end");
                        break;
                    case "3":
                        Cek_point_trace(state_UTM, "begin");

                        //Membaca semua instruksi di pita2 untuk diperiksa cocok
atau tidak atau HALT
                        for (int i=2; i < pita2.Count; i++)
                        {
                            head_pita2 = i;
                            bufferT2 = Pembaca_instruksi(pita2, head_pita2);
                            komponen_state_sekarang = bufferT2[0];
                            komponen_input = bufferT2[1];
                            bufferT4 = int.Parse(pita4[head_pita4].ToString());


                            //CEK PEMBAGIAN STRING DALAM bufferT2:
```

```
                            int p = 0;
                            foreach(string u in bufferT2)
                            {
                                Console.WriteLine("Cek nilai item yaitu
bufferT2[{0}] = {1}",p,u); p++;
                            }
                            //LAPORAN
                            Console.WriteLine("LAPORAN awal for (int i=2; i <
pita2.Count - 2; i++):");
                            Console.WriteLine("LAPORAN head_pita2:{0} dan
bufferT2:{1} dan komponen_state_sekarang:{2} dan komponen_input:{3} dan
bufferT4:{4} ", head_pita2,ConvertStringArrayToString(bufferT2),
komponen_state_sekarang, komponen_input, bufferT4);


                            //CEK KONDISI LOGIK:
                            Console.WriteLine("\nbufferT4.ToString() = {0}",
bufferT4.ToString());
                            Console.WriteLine("komponen_state_sekarang = {0}",
komponen_state_sekarang);

Console.WriteLine("pita1[head_pita1].ToString().ToArray()[0] = {0}",
pita1[head_pita1].ToString().ToArray()[0]);
                            Console.WriteLine("komponen_input[0] = {0} \n",
komponen_input[0]);
                            if ((bufferT4.ToString() == komponen_state_sekarang) &
(pita1[head_pita1].ToString().ToArray()[0] == komponen_input[0]))
                            {

buffer_ut_disalin_keT3.Add(ConvertStringArrayToString_bykonektor(bufferT2,','));
                            }

                        }


                        if (buffer_ut_disalin_keT3.Count == 0)
                        {
                            state_UTM = "HALT";
                            Console.WriteLine("Pencarian instruksi di pita2 tidak
menemukan instruksi yang cocok dengan state sekarang (current state) atau cocok
dengan input");
                        }
                        else
```

```
                        {
                            state_UTM = "4";
                        }
                        //LAPORAN
                        Console.WriteLine("\nLAPORAN nilai buffer_ut_disalin_keT3 =
{0} ", ConvertArrayListToString2(buffer_ut_disalin_keT3));
                        Console.WriteLine("LAPORAN nilai state_UTM = {0} \n",
state_UTM);

                        Cek_point_trace(state_UTM, "end");
                        break;
                    case "4":
                        Cek_point_trace(state_UTM, "begin");
                        pita3 = null;
                        pita3 = new ArrayList();
                        head_pita3 = 0;
                        //ArrayList buffer_ut_disalin_keT3_terurut = new
ArrayList();
                        string[] buffer_sementara = new string[6]; //CEK
DISINI,MUNGKIN BUKAN 7 TETAPI 6
                        int[] urut_ukuran = new int[buffer_ut_disalin_keT3.Count];

                        //LAPORAN
                        Console.WriteLine("LAPORAN awal state 4 = pita3:{0} dan
head_pita3:{1} dan state_UTM:{2} dan buffer_sementara.Length:{3} dan
urut_ukuran.Length:{4} ", ConvertArrayListToString2(pita3), head_pita3,
state_UTM,ConvertStringArrayToString(buffer_sementara), urut_ukuran.Length);
                        //Proses mengurutkan semua instruksi yang ditemukan dan
telah disimpan pada buffer_ut_disalin_keT3
                        {
                            for (int i = 0; i < buffer_ut_disalin_keT3.Count; i++)
                            {
                                buffer_sementara =
Pembaca_instruksi(buffer_ut_disalin_keT3, i);
                                Console.WriteLine("nilai buffer_sementara = {0}",
ConvertStringArrayToString(buffer_sementara));

                                int.TryParse(buffer_sementara[2], out
komponen_ukuran_input); //terjadi lewat error index disini
                                urut_ukuran[i] = komponen_ukuran_input;
                                //LAPORAN
                                Console.WriteLine("LAPORAN di for (int i = 0; i <
buf..... = buffer_sementara:{0} dan buffer_sementara[2]:{1} dan
```

```
komponen_ukuran_input:{2} ", ConvertStringArrayToString(buffer_sementara),
int.Parse(buffer_sementara[2]), komponen_ukuran_input);
                            }

                            //LAPORAN
                            Console.WriteLine("LAPORAN hasil urut ukuran =
urut_ukuran_sebelum_urut:{0} dan state_UTM:{1}",
ConvertIntArrayToString(urut_ukuran), state_UTM);
                            urut_ukuran = urut_ukuran.OrderByDescending(j =>
j).ToArray();
                            //LAPORAN
                            Console.WriteLine("LAPORAN hasil urut ukuran =
urut_ukuran_setelah_urut:{0} dan state_UTM:{1}",
ConvertIntArrayToString(urut_ukuran), state_UTM);

                            ArrayList pita3_sementara = new ArrayList();
                            for (int k = 0; k < urut_ukuran.Length; k++)
                            {
                                for (int i = 0; i < buffer_ut_disalin_keT3.Count;
i++)
                                {
                                    buffer_sementara =
Pembaca_instruksi(buffer_ut_disalin_keT3, i);

                                    //CEK NILAI BUFFER SEMENTARA
                                    int e = 0;
                                    foreach(string y in buffer_sementara)
                                    {
                                        Console.WriteLine("nilai
buffer_sementara[{0}] = {1}",e,y);
                                        e++;
                                    }


                                    int.TryParse(buffer_sementara[2], out
komponen_ukuran_input);
                                    if (urut_ukuran[k] == komponen_ukuran_input)
                                    {

pita3_sementara.Add(ConvertStringArrayToString_bykonektor(buffer_sementara,','));
                                    }
                                }
                            }
```

```
                            pita3 = Penghilang_item_ganda(pita3_sementara);
//APAKAH INI COCOK?

                            //LAPORAN
                            Console.WriteLine("LAPORAN operasi
Penghilang_item_ganda = pita3:{0} dan
state_UTM:{1}",ConvertArrayListToString2(pita3), state_UTM);
                            Console.WriteLine("LAPORAN operasi
Penghilang_item_ganda = pita3_sementara:{0} dan state_UTM:{1}",
ConvertArrayListToString2(pita3_sementara), state_UTM);
                        }

                        head_pita2 = 0;
                        head_pita3 = 0;
                        state_UTM = "5";

                        //LAPORAN
                        Console.WriteLine("LAPORAN setelah state 4 selesai =
head_pita2:{0} dan head_pita3:{1} dan state_UTM:{2}", head_pita2, head_pita3,
state_UTM);

                        Cek_point_trace(state_UTM, "end");
                        break;
                    case "5":
                        Cek_point_trace(state_UTM, "begin");
                        string[] buffer_sementara2 = new string[6];
                        bufferT4 = int.Parse(pita4[head_pita4].ToString());
                        Boolean apakah_halt = true;

                        //LAPORAN:
                        Console.WriteLine("LAPORAN di awal state 5 selesai:
head_pita4:{0}, bufferT4:{1}, apakah_halt:{2}, buffer_sementara2.Length:{3}",
head_pita4, bufferT4, apakah_halt, buffer_sementara2.Length);

                        //Proses eksekusi instruksi di pita3, mulai head_pita3=0
(sudah ditentukan di state 4) yaitu mulai dari sel pertama pita3.
                        for (int i=head_pita3; i < pita3.Count; i++)
                        {
                            //Baca instruksi di sel ke-i, hanya perlu baca
current_state nya dan input
                            buffer_sementara2 = Pembaca_instruksi(pita3, i);
```

```
                            //UJI BUFFER_SEMENTARA:
                            Console.WriteLine("\nNilai buffer_sementara {0}",
ConvertStringArrayToString(buffer_sementara2));
                            int e = 0;
                            foreach (string y in buffer_sementara2)
                            {
                                Console.WriteLine("nilai buffer_sementara2[{0}] =
{1}", e, y);
                                e++;
                            }


                            komponen_state_sekarang = buffer_sementara2[0];
                            komponen_input = buffer_sementara2[1];
                            komponen_state_berikut = buffer_sementara2[3];
                            komponen_output = buffer_sementara2[4];
                            int.TryParse(buffer_sementara2[5], out
komponen_geseran_head);
                            string[] tampung_baca_pita1_sementara = new
string[komponen_input.Length];
                            //CEK INSTRUKSI YANG DIBACA DI SEL KE-i DI PITA3:
                            Console.WriteLine("\nCEK INSTRUKSI YANG DIBACA DI SEL
KE-i YAITU KE-{0} DI PITA3:",i);
                            Console.WriteLine("buffer_sementara = {0}",
ConvertStringArrayToString(buffer_sementara2));
                            Console.WriteLine("komponen_state_sekarang = {0}",
komponen_state_sekarang);
                            Console.WriteLine("komponen_input = {0}",
komponen_input);
                            Console.WriteLine("komponen_state_berikut = {0}",
komponen_state_berikut);
                            Console.WriteLine("komponen_output = {0}",
komponen_output);
                            Console.WriteLine("komponen_geseran_head = {0}",
komponen_geseran_head);
                            Console.WriteLine("tampung_baca_pita1_sementara.Length
= {0} \n", tampung_baca_pita1_sementara.Length);

                            //Baca pita1 sebanyak ukuran komponen input, tapi ingat
pointer head_pita1 jangan dirubah atau digeser karena ini cuma digunakan sementara.
                            //CEK SEBELUM ERROR:
                            Console.WriteLine("\nCEK SEBELUM ERROR:", i);
                            Console.WriteLine("head_pita1 = {0}", head_pita1);
```

```
                            Console.WriteLine("komponen_input = {0}",
komponen_input);
                            Console.WriteLine("komponen_input.Length + head_pita1 =
{0}\n", head_pita1+ komponen_input.Length);

                            for (int h= head_pita1;h< komponen_input.Length +
head_pita1; h++)
                            {
                                Console.WriteLine("Nilai h = {0}", h);
                                Console.WriteLine("Nilai h-head_pita1= {0}", h-
head_pita1);
                            }

                            for (int h = head_pita1; h < komponen_input.Length +
head_pita1; h++)
                            {
                                if (h<pita1.Count) tampung_baca_pita1_sementara[h -
head_pita1] = pita1[h].ToString(); else
                                {
                                    tampung_baca_pita1_sementara[h - head_pita1] =
";"; //Perhatikan di masa depan pengisian ini, apakah bukan diisi " " saja yang
bagus?
                                }
                            }

                            //CEK INPUT YANG DIBACA DI SEL KE-i DI PITA1:
                            Console.WriteLine("\nCEK INPUT YANG DIBACA DI SEL KE-h
YAITU KE-{0} DI PITA1:", head_pita1);
                            Console.WriteLine("tampung_baca_pita1_sementara = {0}",
ConvertStringArrayToString(tampung_baca_pita1_sementara));

                            if (komponen_input ==
ConvertStringArrayToString(tampung_baca_pita1_sementara))
                            {
                                //Berpindah ke state berikut:
                                pita4[0] = komponen_state_berikut;
                                //CEK PERPINDAHAN STATE BERIKUT DI PITA4:
                                Console.WriteLine("\nCEK PERPINDAHAN STATE BERIKUT
DI PITA4:");
                                Console.WriteLine("pita4[0] = {0}", pita4[0]);

                                //menulis output di pita1:
                                //CEK PENULISAN OUTPUT DI PITA1:
```

```
                                Console.WriteLine("\nCEK PENULISAN OUTPUT DI
PITA1:");
                                for (int h = head_pita1; h < komponen_output.Length
+ head_pita1; h++)
                                {
                                    if (h < pita1.Count) pita1[h] =
komponen_output[h - head_pita1]; else pita1.Add(komponen_output[h - head_pita1]);
                                    Console.WriteLine("pita1[{0}] = {1}",
h,pita1[h]);
                                }




                                //Menggeser head_pita1 sejauh d?
                                //CEK PERGESERAN d DI PITA1:
                                Console.WriteLine("\nCEK PERGESERAN d DI PITA1:");
                                Console.WriteLine("head_pita1 sebelumnya: {0}",
head_pita1);
                                head_pita1 =head_pita1+ komponen_geseran_head;
                                Console.WriteLine("head_pita1 setelah digeser
sejauh d={0} adalah: {1}", komponen_geseran_head, head_pita1);

                                apakah_halt = false; //Jika ketemu instruksinya dan
                                //CEK PERGESERAN d DI PITA1:
                                Console.WriteLine("\nCEK NILAI apakah_halt YAITU
:{0}", apakah_halt);
                                Console.WriteLine("KALAU FALSE MAKA BERARTI
INSTRUKSI KETEMU DAN TERJADI TRANSISI STATE");
                                break;
                            }
                        }

                        if (apakah_halt == true)
                        {
                            state_UTM = "HALT";
                            Console.WriteLine("Pencarian instruksi di pita3 tidak
menemukan input yang cocok dengan instruksi");
                        }
                            else state_UTM = "6";
                        Cek_point_trace(state_UTM, "end");
                        break;
                    case "6":
```

```
                        Cek_point_trace(state_UTM, "begin");
                        state_UTM = "1";
                        Cek_point_trace(state_UTM, "end");
                        break;
                    case "HALT":
                        Cek_point_trace(state_UTM, "begin");
                        Console.WriteLine("Mesin telah HALT, mesin halt di posisi
state {0}", pita4[0]);
                        HALT = true;
                        Cek_point_trace(state_UTM, "end");
                        break;
                    default:
                        Cek_point_trace(state_UTM, "begin");
                        Console.WriteLine("Maaf state mesin tidak dikenal, cek
perubahan nilai pointer state_UTM di dalam setiap case.");
                        Cek_point_trace(state_UTM, "end");
                        break;
                }

                //if (HALT) Console.WriteLine("Mesin telah HALT, state accept
tercapai. State accept adalah {0}", (int)pita2[1]);
                Cek_point_trace(state_UTM, "end do");
            } while (!HALT);
            Cek_point_trace(state_UTM, "diluar do");

        }

    }
}
```

PENUTUP

Dari semua deskripsi di atas, telah dikemukakan dengan lengkap *pseudocode* dan kode program compiler dan interpreter untuk interaksi antara sistem Augmented Reality dengan manusia menggunakan bahasa interaksi yang gramatikanya sepadan dengan *generalized Finite State Automata,* yaitu L(gFSA).

DAFTAR PUSTAKA

Turing, A. M. (1938) ‘On computable numbers, with an application to the entscheidungsproblem. a correction’, *Proceedings of the London Mathematical Society,* s2-43(1), pp. 544–546.

doi: 10.1112/plms/s2-43.6.544.