DOKUMEN BUKTI HAK CIPTA

Rancangan Mesin Compiler-Interpreter untuk Sistem Augmented Reality yang gramatika bahasa interaksinya setara dengan generalized Finite State Automata (gFSA)

Pengusul Hak Cipta

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

Dr. Azhari, M.T

Dr. Suprapto, M.Kom

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2021

Rancangan Mesin Compiler-Interpreter untuk Sistem Augmented Reality yang gramatika bahasa interaksinya setara dengan generalized Finite State Automata (gFSA)

A. RINCIAN COMPILER/INTERPRETER

Bagian ini merinci rancangan compiler/interpreter antara sistem Augmented Reality dengan manusia yang diusulkan sebagai bagian utama dari pengusulan hak cipta. Deksripsi rancangan ini dijelaskan secara rinci pada bagian klaim sebagai berikut.

KLAIM RANCANGAN

Bagian mengemukakan jalannya rincian dan langkah-langkah pembuktian secara matematika bahwa rancangan yang dibuat dapat dibuktikan kebenaran algoritmanya atau langkah-langkah mesinnya. Untuk menunjukkan secara ketat bagaimana konstruksi dilakukan, konstruksi ini dimulai dengan sebuah konjektur, kemudian pada bagian berikutnya sebuah deskripsi lengkap yang membuktikan kebenran kojektur sekaligu menunjukkan secara eksplisit proses konstruksi mesin compiler-interpreter yang diklaim dalam dokumen hak cipta ini. Konjektur itu adalah sebagai berikut:

> ***Konjektur 4. (Konjektur gFSA-UTM-Interpreter-Compiler)*** *Untuk setiap interaksi pengguna-ARS, dapat dikonstruksikan sebuah mesin Turing universal (UTM) yang berkedudukan sebagai sebuah interpreter atau compiler bagi interaksi pengguna-ARS yang menggunakan bahasa formal reguler.*

Berikutnya hendak ditunjukkan bahwa konjektur gFSA-UTM-*Interpreter-Compiler* (konjektur 4) adalah benar dan dapat dikonstruksikan. Konjektur 4 ini diwakilkan oleh pernyataan proposisi 3 kemudian dikemukakan argumentasi dan pembuktian matematika bahwa proposisi 3 adalah benar. Konjektur 4 atau proposisi 3.

Pembuktian dan Analisis Konjektur 4

Bagian ini menunjukkan sejumlah argumentasi untuk memberikan serangkaian bukti dan konstruksi yang mendukung konjektur 4.

> ***Proposisi 3.*** *Untuk setiap interaksi pengguna-ARS dapat dikonstruksikan sebuah mesin Turing universal (UTM) yang dapat digunakan sebagai sebuah interpreter atau compiler bagi bahasa formal reguler yang dikemukakan oleh pengguna untuk berinteraksi dengan ARS.*

Usaha untuk menunjukkan bahwa di sana ada sebuah UTM yang dapat bertindak sebagai *interpreter* atau *compiler* dilakukan dalam tahap:

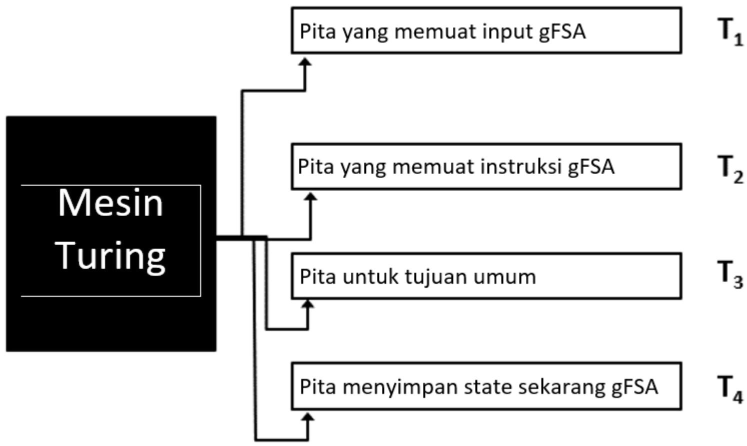
- Tahapan pertama menunjukkan secara intuitif tentang bagaimana membuat sebuah *interpreter* menggunakan mesin Turing yang dapat mensimulasikan seluruh gFSA yang mungkin. Ide ini dikonstruksikan secara langsung dan intuitif dalam arti bahwa sebuah mesin Turing dibuat dan dapat mensimulasikan seluruh gFSA yang mungkin tanpa terlebih dahulu melakukan enkoding atau membangun fungsi enkoding bijective terhadap gFSA ke dalam alphabet pita mesin Turing. Perkenalan intuitif ini semata-mata untuk memudahkan pandangan awal bahwa mesin Turing seperti itu dapat dibuat.
- Tahapan kedua adalah dikemukakan sebuah bukti formal bahwa mesin Turing seperti itu dapat dibuat dan dibuktikan dapat mensimulasikan seluruh gFSA yang mungkin dan dapat pula dibuktikan bahwa mesin Turing seperti itu adalah juga sebuah mesin Turing universal (UTM).

Perkenalan intuitif ini dimulai pada paragraf berikut.

Konstruksikan secara intuitif sebuah mesin Turing yang dapat menerima seluruh gFSA sebagai berikut:

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.



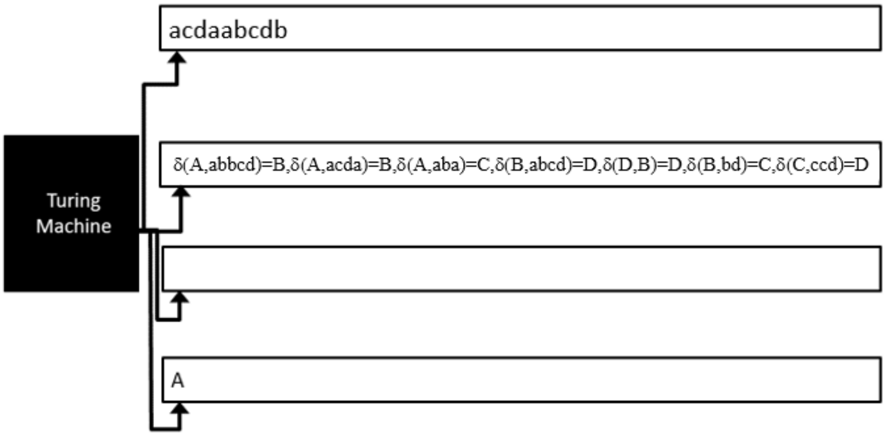
Gambar 6.1. Mesin Turing M untuk Mensimulasikan gFSA

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: $\{\delta(A, abbcd) = B, \delta(A, acda) = B, \delta(A, aba) = C, \delta(B, abcd) = D, \delta(D, B) = D, \delta(B, bd) = C, \delta(C, ccd) = D\}$.
- 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.

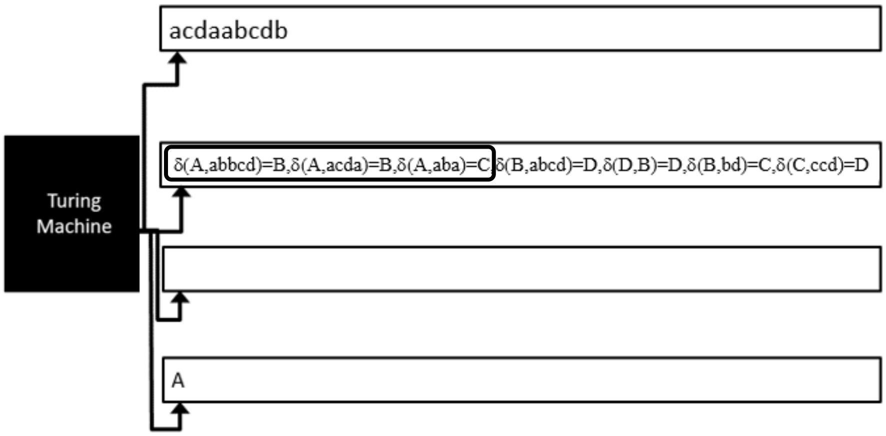
  Jika *state* pada argumen instruksi pertama adalah sama dengan *state* yang sedang terbaca pada (*current state*) step 2a dan karakter pertama pada *word* dalam argumen instruksi pertama adalah sama dengan karakter yang terbaca pada step 1 maka lanjut ke step 5.

- 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 $\{\delta(A, abbcd) = B, \delta(A, acda) = B, \delta(A, aba) = C\}$.

Ilustrasi untuk step 3 digambarkan oleh Gambar 6.3:



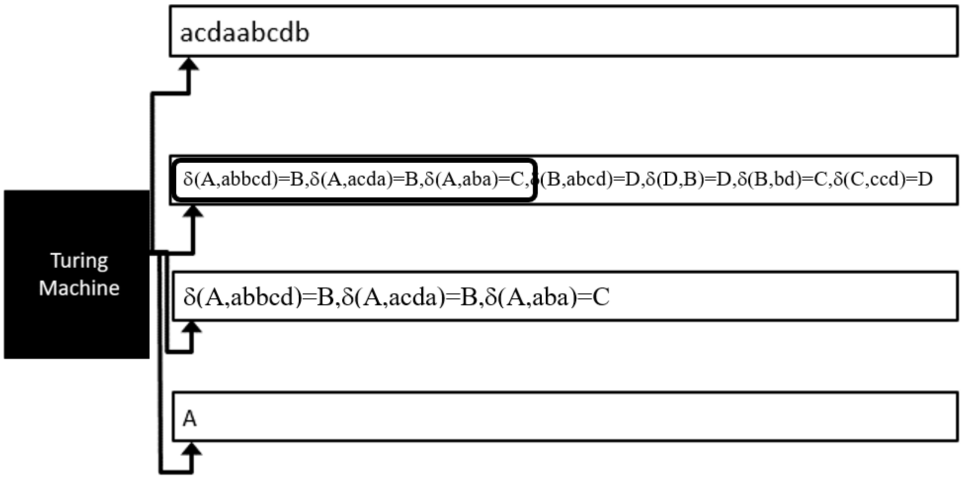
Gambar 6.3. Ilustrasi Step 3 Simulasi gFSA Mesin Turing M

- 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:

$\delta(A, abbcd) = B, \delta(A, acda) = B, \delta(A, aba) = C.$

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).

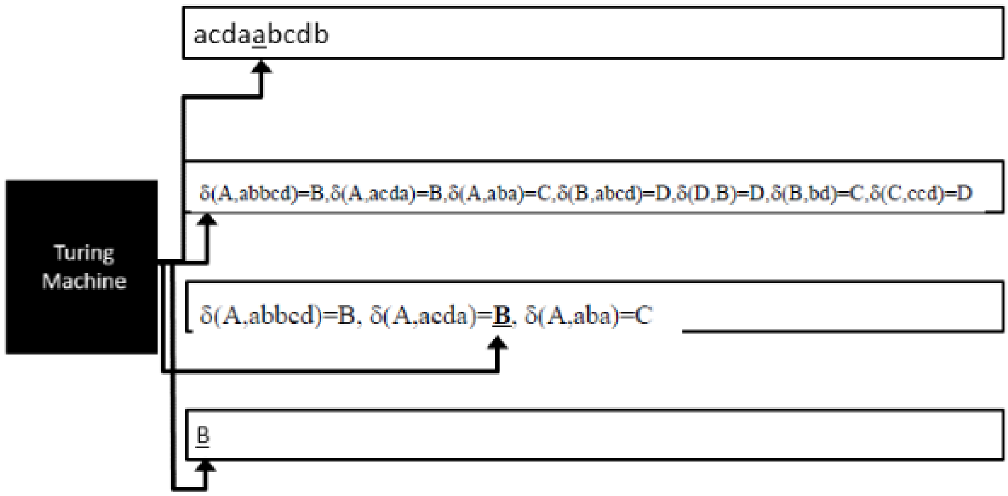
  Jika input *word* pada instruksi pertama tidak *match* dengan potongan *word* yang terbaca pada T_1 maka *head* berpindah ke instruksi kedua dan *head* pada pita T_1 direset posisinya pada karakter pertama potongan *word*. Input *word* pada instruksi kedua dibandingkan dengan potongan *word* yang terbaca 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 $\delta(A, acda) = B$ *maka M* mengeksekusi $\delta(A, acda) = B$ 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).

Permulaan Konstruksi Formal

Berikut ini akan dikemukakan sebuah konstruksi formal dan bukti matematika bahwa di sana ada sebuah mesin Turing yang dapat mensimulasikan sebarang gFSA. Pada bagian ini, mesin Turing *M* yang sebelumnya dikemukakan secara intuitif, mesin *M* tersebut selanjutnya akan dikonstruksikan secara formal dan dibuktikan bahwa dia dapat mensimulasikan seluruh gFSA. *M* juga akan ditunjukkan adalah sebuah mesin Turing universal (UTM) dengan mendemosntrasikan bahwa *M* juga dapat mensimulasikan sebuah mesin Turing standar. Jika *M* dapat mensimulasikan sebuah mesin Turing standar maka itu berarti secara otomatis *M* dapat mensimulasikan seluruh mesin Turing dan algoritma (berdasarkan pada *Church-Turing thesis*). Jika dia dapat ditunjukkan sebagai sebuah UTM maka otomatis juga berarti bahwa *M* dapat menjadi *interpreter* atau *compiler* bagi semua bahasa selain bahasa reguler (semua bahasa dalam seluruh tipe bahasa Chomsky) dengan cara menerjemahkan terlebih dulu gramatika bahasa-bahasa tersebut ke dalam representasi mesin Turing sehingga dapat disimulasikan oleh UTM.

Pekerjaan membangun mesin Turing universal telah menjadi sebuah petualangan intelektual bagi banyak orang di masa lalu. A.M. Turing sendiri telah meletakkan ide tentang mesin otomatis (Turing, 1938), seperti juga yang telah ditunjukkan oleh Stephen Dolan yang dia menunjukkan bahwa hanya dengan perintah mov sepanjang registers pada mesin prosesor x86 dapat dibuktikan bahwa itu dapat mensimulasikan mesin standar sehingga dia *Turing complete* (Dolan, 2013) sehingga dia dapat membangun sebuah mesin Turing universal, yang memiliki arti bahwa dia dapat mensimulasikan seluruh algoritma. Serupa juga dengan kerja Minsky (Minsky, 1967) yang telah berusaha membuktikan bahwa sebuah *counter machine* atau *register machine* dapat menjadi sebuah mesin Turing universal dengan jumlah instruksi seminimum mungkin.

Pada bagian ini akan juga dikemukakan sebuah bukti formal bahwa mesin Turing *M* sebelumnya adalah juga dapat mensimulasikan seluruh gFSA dan mesin Turing standar sehingga itu cukup untuk mengklaim *M* sebagai sebuah mesin Turing universal (UTM). Pada kerja ini, pertama kali gFSA digeneralisasikan menjadi sebuah mesin Turing juga tetapi mesin Turing yang mengeksekusi *word*, tidak hanya mengeksekusi sebuah simbol pada pita. Dari perumuman ini diharapkan dua hal, yaitu sebuah mesin Turing yang terbukti dapat mensimulasikan seluruh gFSA sehingga dapat benar-benar menjadi *interpreter* atau *compiler* bagi bahasa L(gFSA) dan sebuah UTM yang dapat mensimulasikan mesin Turing standar sehingga berimplikasi dapat menjadi *interpreter* atau *compiler* bagi semua bahasa dalam seluruh tipe bahasa Chomsky.

Untuk usaha ini, definisi gFSA pada definisi 5.1. diperluas menjadi seperti pada definisi 6.1 yang akan 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 $\varnothing$. 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 $\delta(A, abcd) = (B, cda, -2)$ 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 $\delta(A, aaac) = (B, aadcda, 2)$, 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 $f, f + 1, \ldots k$ 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: $\delta(q, s) = (q', s', d)$ dan $s, s' \in L_{encode}$, $0 \leq q \leq f$ dan $d \in \{-m, 0, n\}$, $0 \leq q' \leq k$ 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*, $t_1, t_2, t_3, \ldots$ dan $t_1, t_2, t_3, \ldots$ 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 $encoding(MW) = k, f, (q, s, |s|, q', s', d), \ldots, (q, s, |s|, q', s', d)$ dan seterusnya hingga seluruh instruksi *MW* tertuliskan, dan n, f, q, q', d adalah bilangan-bilangan bulat. $0 \leq f$, $q < f$, $q' \leq k$, $d \in \{-m, 0, n\}$ dan $s, s' \in L_{encode}$.

Konstruksi UTM untuk MW

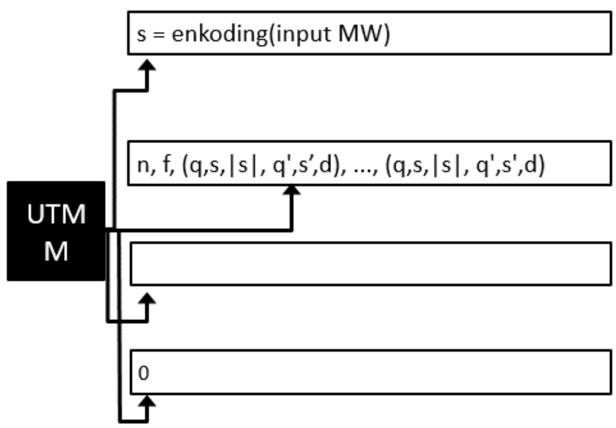
Pada bagian ini akan dikonstruksikan sebuah mesin Turing universal (UTM) yang sebenarnya merupakan versi mesin Turing *M* yang disuplai enkoding dari *MW* dan input *MW*. Pada versi ini terdapat modifikasi algoritma versi intuitif sebelumnya. Mesin Turing *M* versi ini adalah juga seperti Gambar 6.1, 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).

Bukti Formal Menggunakan Induksi Matematika

Pada bagian ini akan ditunjukkan bahwa *M* adalah sebuah mesin Turing universal yang diharapkan dapat dilakukan dengan menggunakan induksi matematika. Skenario dari pembuktian adalah sebagai berikut.

Pertama, Setiap fungsi transisi *MW* yaitu $\delta(q, s) = (q', s', d)$ ditulis ulang ke dalam bentuk instruksi 6-tuples $(q, s, |s|, q', s', d)$ *kemudian* tandai setiap instruksi sebagai *p* yang $p = (q, s, |s|, q', s', d)$ dan $|s|$ adalah panjang *word s*.

Untuk setiap input *word* yang diberikan kepada *MW*, *MW* menghasilkan sebuah *run* (barisan eksekusi instruksi) yang mengeksekusi input *word* hingga HALT atau FAIL. Andaikan setiap *run* dari *MW* diekspresikan sebagai $r_{MW} = p_0, p_1, p_2, p_3, ..., p_k$ dan $p_i = (q_i, s_i, |s_i|, q_i', s_i', d_i)$ dan untuk setiap input $w = w_1w_2w_3...w_h$ dan w_i adalah *word* ke-i, $i = 1,2,3, ...h$. *Run* untuk input w adalah ditulis sebagai $r_{MW}(w)$. Interpretasi $r_{MW}(w)$ adalah bahwa *MW* mengeksekusi input *w* dalam sebuah urutan transisi dalam bentuk barisan eksekusi instruksi r_{MW}.

Secara sederhana, *M* dikatakan mensimulasikan *MW* jika *M* juga dapat menghasilkan barisan eksekusi instruksi r_{MW}, yaitu bahwa *M* juga dapat memproduksi $r_{MW}(w)$ secara tepat sama dengan $r_{MW}(w)$ yang dihasilkan oleh *MW* sendiri. Secara umum, andaikan r_M adalah barisan instruksi yang dieksekusi oleh *M* maka *M* dikatakan dapat mensimulasikan *MW* jika hanya jika untuk setiap input *w* dari *MW* sehingga *MW* menghasilkan $r_{MW}(w)$ maka *M* dapat memproduksi $r_M(w)$ dan $r_{MW}(w)=r_M(w)$.

Lebih umum lagi, misalkan α adalah sebuah fungsi enkoding yang bijective, sedemikian sehingga misalkan p_i adalah instruksi ke-i dan $\alpha(p_i)=\alpha(q_i,w_i,|w_i|,q_i',w_i',d_i)=(\alpha(q_i),\alpha(w_i),\alpha(|w_i|),\alpha(q_i'),\alpha(w_i'),\alpha(d_i))$ dan r_{MW} adalah diekspresikan sebagai barisan instruksi yang dieksekusi, yaitu $r_{MW} = p_0, p_1, p_2, p_3, ..., p_k$ maka fungsi enkoding α untuk r_{MW} adalah $\alpha(r_{MW}) = \alpha(p_0), \alpha(p_1), \alpha(p_2), \alpha(p_3), ..., \alpha(p_k)$ dengan menggunakan fungsi enkoding α, dapat diformulasikan perumuman dari simulasi sebagai berikut:

> ***Definisi 6.3*** *Mesin Turing M dikatakan dapat mensimulasikan MW jika hanya jika untuk setiap input w untuk MW sehingga* $r_{MW}(w)$ *dan* α *adalah sebuah fungsi enkoding yang bijective maka M dapat memproduksi* $r_M(s)$ *dan* $\alpha(r_{MW})=r_M$ *dan* $s=\alpha(w)$.

Dengan menggunakan definisi 6.3, secara umum mesin Turing *M* dapat dibuktikan sebagai sebuah mesin Turing universal. Argumen ini dapat dinyatakan sebagai berikut:

> ***Teorema 6.1*** *M adalah sebuah mesin Turing yang dapat mensimulasikan MW.*

Bukti:

Sebelumnya ketika mesin Turing *M* dikonstruksikan, pada penjelasan sebelumnya telah ditunjukkan sebuah fungsi enkoding yang memetakan setiap simbol *state* ke sebuah bilangan dalam $C \cup \{\emptyset\}$ dan untuk setiap simbol pita *MW* ke $C \cup \{\emptyset\}$. Sebuat fungsi enkoding ini sebagai α dan tentunya *bijektive* (dapat ditetapkan atau dibuat *bijective* jika bukan menggunakan contoh sebelumnya karena fungsi enkoding ini bersifat bebas untuk ditetapkan). Ini memiliki arti bahwa setiap saat input $w = w_1w_2w_3...w_h$ dimasukkan sebagai input ke *MW* maka *MW* menghasilkan sebuah *run* $r_{MW}(w)$ yang mengeksekusi *w* sampai HALT atau FAIL dan karena fungsi enkoding α yang bersifat bijective, dapat dibangun $\alpha(r_{MW})= \alpha(p_0,p_1,p_2,p_3,...,p_k) = \alpha(p_0), \alpha(p_1), \alpha(p_2), \alpha(p_3), ...,\alpha(p_k)$ and $\alpha(p_i) = \alpha(q_i,w_i,|w_i|,q_i',w_i',d_i) = (\alpha(q_i), \alpha(w_i), \alpha(|w_i|), \alpha(q_i'), \alpha(w_i'), \alpha(d_i))$, selanjutnya, akan dibuktikan bahwa mesin *M* juga membuat $r_M(s)$ dan $\alpha(r_{MW})=r_M(s)$ dan $s=\alpha(w)$ (i.e, yaitu bahwa *M* mensimulasikan *MW*). Ini berarti, harus ditunjukkan bahwa untuk $r_{MW} = p_0, p_1, p_2, p_3, ..., p_k$ diikuti oleh $\alpha(p_i)=p_i'$ dan $r_M = p_0', p_1', p_2', p_3', ..., p_k'$. i=1,2,3,...,k.

Untuk bukti ini, $p_i = (q_i, w_i, |w_i|, q_i', w_i', d_i)$ adalah di enkoding dalam ekspresi enkoding $p_i'=\alpha(p_i)$ konsisten dengan konstruksi *M* sebelumnya. Enkoding itu adalah sebagai berikut:

$$p_i'= \alpha(p_i)= \alpha(q_i,w_i,|w_i|, q_i',w_i',d_i)=(\alpha(q_i), \alpha(w_i), \alpha(|w_i|), \alpha(q_i'), \alpha(w_i'), \alpha(d_i)) \quad (6.3)$$

Setiap *state* dari *MW* dienkoding ke bilangan bulat dan $s = \alpha(w)$ agar bisa sesuai dengan enkoding *MW* sebelumnya. *s* adalah sebuah *word* yang dikonstruksikan dari $C \cup \{\emptyset\}$ sehingga persamaan enkoding (6.3) dapat ditulis sebagai:

$$p_i'= \alpha(p_i)= \alpha(q_i,w_i,|w_i|, q_i',w_i',d_i)=$$
$$(\alpha(q_i), \alpha(w_i), \alpha(|w_i|), \alpha(q_i'), \alpha(w_i'), \alpha(d_i))=(m_i, s_i, |s_i|, m_i',s_i', d_i) \quad (6.1)$$

dan m_i, m_i', d_i bilangan-bilangan bulat dan s_i, $s_i' \in L_{encode}$

Untuk *k* = 0:

Untuk iterasi $k = 0$ dari algoritma M adalah tahapan langkah-langkah inisialisasi berikut:

Inisialisasi:

- Step 1: M menulis 0 pada sel pertama pita T_4. ($0 = \alpha(q_0)$ adalah enkoding untuk *state* start).
- Step 2: M menulis enkoding *word* input s dari MW pada T_1, penulisan dimulai pada sel pertama ($s = \alpha(w)$ pada T_1, $s=s_1s_2s_3...s_h=\alpha(w_1)\alpha(w_2)\alpha(w_3)...\alpha(w_h) =\alpha(w)$)
- Step 3: M mengosongkan T_3.
- Step 4: M menulis semua encoding(MW) pada T_2, penulisan dimulai pada sel pertama. (M menulis $p_i'= \alpha(p_i)$ untuk seluruh instruksi p_i milik MW).
- Step 5: M mereset seluruh posisi *head* pada pita T_1, T_2, T_3 dan T_4 kembali pada posisi sel pertama masing-masing pita. (*Word* s_0 sebagai yang pertama ditunjuk oleh *head* pada T_1 dan 0 ditunjuk pada T_4 sesuai dengan susunan template instruksi ($0, s_0,?,?,?,?$). Akan tetapi, pada T_3 di sana terdapat instruksi-instruksi ($0, s_0, |s_0|, m_0', s_0', d_0$) yang sesuai dengan ($0,s_0,?,?,?,?$) sebagai konsekuensi step 4).

Berdasarkan pada step 5 ini, untuk $p_0 = (q_0, w_0, |w_0|, q_0', w_0', d_0)$ dan q_0, q_0' adalah *state-state* pada MW dan w_0, w_0' adalah input dan output *word* pada MW. d_0 adalah pergerakan *head* dalam pita MW kemudian diperoleh $p_0' = (0, s_0, |s_0|, m_0', s_0', d_0)$ pada step 5, yaitu:

$\alpha(p_0)= \alpha(q_0,w_0,|w_0|,q_0',w_0',d_0)= (\alpha(q_0), \alpha(w_0), |\alpha(w_0)|, \alpha(q_0'), \alpha(w_0'), \alpha(d_0)) = (0,s_0,|s_0|,m_0',s_0',d_0)=p_0'$ konsisten dengan persamaan (6.1), yaitu $\alpha(p_0)=p_0'$

Ini berarti telah ditunjukkan bahwa untuk $r_{MW} = p_0$ maka $r_M= \alpha(r_{MW})=\alpha(p_0)=p_0'$ dan $s_0= \alpha(w_0)$ konsisten dengan definisi 6.3.

Untuk *k* = n:

Andaikan bahwa pada iterasi k = n algoritma M menerapkan $\alpha(p_n)=p_n'$ dan $s_n=\alpha(w_n)$ adalah benar, yaitu bahwa:

$\alpha(p_n)= \alpha(q_n,w_n,|w_n|,q_n',w_n',d_n)= (\alpha(q_n), \alpha(w_n), |\alpha(w_n)|, \alpha(q_n'), \alpha(w_n'), \alpha(d_n)) = (m_n,s_n,|s_n|,m_n',s_n',d_n)=p_n'$, yaitu bahwa mesin dapat memproduksi p_n' yang ekivalen secara one-to-one dengan p_n, yaitu $p_n'=\alpha(p_n)$ dan α adalah bijective sehingga $r_M=\alpha(r_{MW})=\alpha(p_0,p_1,p_2,...,p_n)=\alpha(p_0),\alpha(p_1),\alpha(p_2),...,\alpha(p_n)=p_0',p_1',p_2',...,p_n'$ dan $s=\alpha(w)$ adalah konsisten dengan definisi 6.3.

Untuk *k* = n + 1:

Selanjutnya, akan ditunjukkan bahwa pada iterasi k=n+1, adalah juga benar bahwa $\alpha(p_{n+1})=p_{n+1}'$ and $s_{n+1}=\alpha(w_{n+1})$.

Dengan menggunakan algoritma *M*, simulasi *MW* pada k=n + 1:

- Step 1: *M* membaca sel yang telah ditunjuk oleh *head* pada T_4. (Karena pada iterasi ke-n benar, yaitu $\alpha(p_n) = p_n'$ dan $s_n = \alpha(w_n)$ maka pada k = n + 1, *head* pita T_4 menunjuk *state* yang aktif (*current state*) adalah m_n').

  Jika $m_n' \underline{\geq} f$ maka HALT. Instruksi mencapai *state accept*.

  (Ini terjadi karena sifat *bijective* dari α maka invers m_n' adalah sebuah *state accept* di *MW*, yang berarti bahwa *MW* juga HALT)

  Jika $m_n' < f$ maka lanjut ke step 2a.
- Step 2a: *M* membaca sel yang ditunjuk oleh *head* pada pita T_1. (pada iterasi k = n + 1, T_1 menunjuk karakter pertama *word* s_{n+1} *karena* $s=s_1s_2s_3...s_h= \alpha(w_1)\alpha(w_1)\alpha(w_1)...\alpha(w_h)= \alpha(w)$ maka $s_{n+1}= \alpha(w_{n+1})$).
- Step 2b: Jika T_3 tidak kosong, bandingkan *state* yang terbaca pada step 1 dengan *state* pada argumen instruksi pertama dan karakter pertama yang terbaca pada step 2a dengan karakter pertama input *word* pada argumen instruksi pertama di pita T_3 (yaitu, *current state* adalah m_n' di pita T_4 dan *head* pada T_1 menunjuk karakter pertama s_{n+1}. Jika tupel (m_n', s_{n+1}, ?, ?, ?, ?) tidak *match* dengan instruksi pertama maka lanjut ke step 3, jika tupel (m_n', s_{n+1}, ?, ?, ?, ?) *match* dengan instruksi pertama maka lanjut ke step 5)
- Step 3: *M* mencari semua instruksi pada pita T_2 yang karakter pertama input *word* nya adalah sama dengan yang sedang ditunjuk oleh *head* pada pita T_1 dan *state* pada argumen instruksi adalah sama dengan *current state* yang sedag ditunjuk oleh *head* pada T_4.

  Jika tak terdapat instruksi yang *match* maka FAIL dan HALT (yaitu jika tidak terdapat instruksi yang *match* dengan tupel (m_n', s_{n+1}, ?, ?, ?, ?) maka FAIL dan HALT. Karena semua enkoding instruksi *MW* tertulis di pita T_2 dan fungsi enkoding α bersifat bijective maka ini berarti di sana tidak terdapat juga instruksi pada *MW* yang dapat mengeksekusi bentuk inverse dari α yaitu ($\alpha^{-1}(m_n')$, $\alpha^{-1}(s_{n+1})$,?,?,?,?), ini berarti bahwa *MW* adalah juga FAIL dan HALT).

  Jika ditemukan 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 pada pita T_3, Penulisan secara berurut dimulai dari panjang *word* terbesar input *word* instruksi dengan membaca |*s*| pada *(q, s,* |*s*|, *q', s', d)*

kemudian *head* pada pita T_2 kembali ke posisi sel pertama T_2 dan *head* pada T_3 kembali ke posisi sel pertama T_3.

- Step 5: *M* mulai membaca instruksi pada urutan pertama T_3 dan membandingkan input *word* pada T_1. Jika instruksi pada T_3 terbaca habis dan tidak ada instruksi yang *match* maka FAIL dan HALT (yaitu jika tidak ada instruksi yang *match* dengan tupel (m_n', s_{n+1}, ?, ?, ?, ?) maka FAIL dan HALT. Karena semua enkoding instruksi *MW* tertulis di pita T_2 dan sifat *bijective* fungsi enkoding α maka ini berarti bahwa di sana juga tidak ada instruksi pada *MW* yang dapat mengeksekusi invers dari enkoding instruksi, yaitu tak ada ($\alpha^{-1}(m_n')$, $\alpha^{-1}(s_{n+1})$,?,?,?,?), ini berarti *MW* juga FAIL dan HALT).

  Jika sebuah instruksi ditemukan *match* dengan (m_n', s_{n+1}, ?, ?, ?, ?), yaitu $(m_n',s_{n+1},|s_{n+1}|,m_{n+1}', s_{n+1}', d_{n+1})$, tidak peduli bahwa *MW* adalah deterministic atau nondeterministic maka invers dari $(m_n',s_{n+1},|s_{n+1}|,m_{n+1}', s_{n+1}', d_{n+1})$ haruslah juga *exist* tersebab oleh sifat *bijective* fungsi enkoding α dan karena telah ditentukan bahwa semua enkoding instruksi *MW* tertulis di pita T_3, tak ketinggalan satupun, yaitu:

  $(\alpha^{-1}(m_n'), \alpha^{-1}(s_{n+1}),|\alpha^{-1}(s_{n+1})|, \alpha^{-1}(m_{n+1}'), \alpha^{-1}(s_{n+1}'), \alpha^{-1}(d_{n+1})) = (q_{n+1},w_{n+1},|w_{n+1}|,q_{n+1}',w_{n+1}',d_{n+1})=p_{n+1}$

  Step 5 membuktikan bahwa untuk iterasi ke k = n + 1, diperoleh:

  $r_M=\alpha(r_{MW})=\alpha(p_0,p_1,p_2,...,p_n,p_{n+1})=\alpha(p_0),\alpha(p_1),\alpha(p_2),...,\alpha(p_n),\alpha(p_{n+1})=p_0',p_1',p_2',...,p_n', p_{n+1}'$ and $s=\alpha(w)$ konsisten dengan definisi 6.3.

  Catatan:

  $m_n' = m_{n+1}$ karena m_n' mestilah state awal dari p_{n+1} sebagai sebelumya di adalah state akhir dari p_n sebagai bagian dari keterkaitan eksekusi instruksi. Keterkaitan eksekusi instruksi yaitu bahwa barisan instruksi $p_0,p_1,p_2,...,p_n$ sambung menyambung antar state, yaitu state akhir p_i menjadi state awal p_{i+1}.

- Step 6: Ulangi step 1 sampai 5.

Secara umum berdasarkan algoritma mesin Turing *M*, run r_M dapat mensimulasikan run r_{MW} yaitu bahwa $r_M = \alpha(r_{MW})$ menurut definisi 6.3.

Teorema 6.1 Telah dibuktikan.

Selanjutnya, secara trivial dapat dibuktikan teorema berikut:

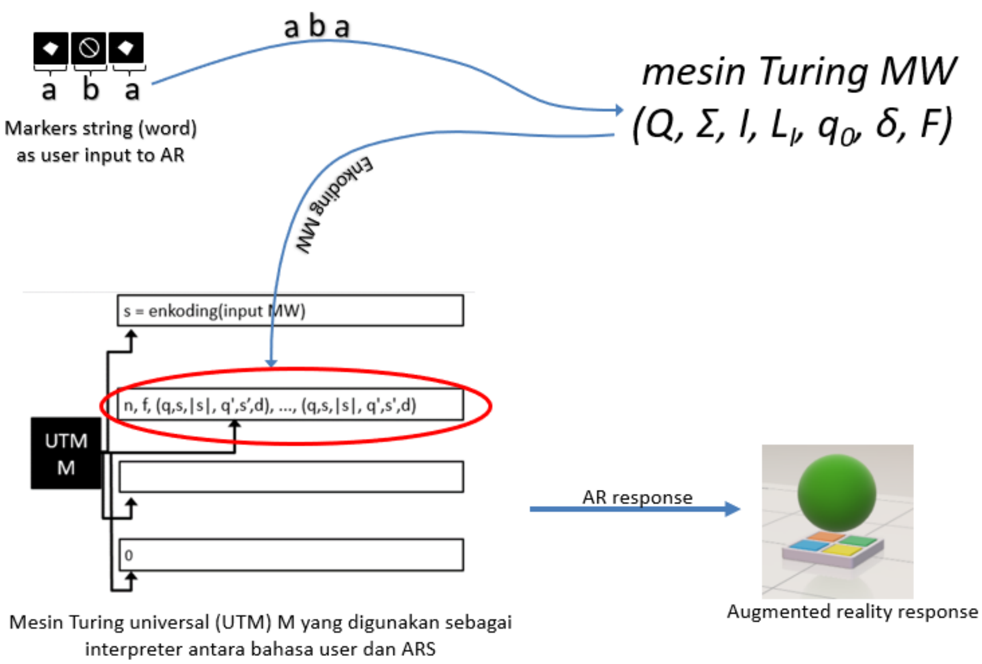
Teorema 6.2 *Mesin Turing M yang dapat mensimulasikan MW adalah sebuah mesin Turing universal (UTM).*

Bukti:

Ambil contoh $w=w_1w_2w_3...w_h$ adalah input *word* dari *MW* sedemikian sehingga $|w_i|=1$ untuk setiap i = 1,2,3, ..., h dan d = {-1,0,1} adalah nilai perpindahan *head* pita maka *MW* adalah sebuah mesin Turing standar. Akan tetapi, dengan menggunakan teorema 6.1. maka teorema 6.2. terbukti.

Implementasi Teorema 6.1 dan 6.2 (Model Interpreter)

Berdasarkan teorema 6.1 dan teorema 6.2., 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)

Implementasi Teorema 6.1 dan 6.2 (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.

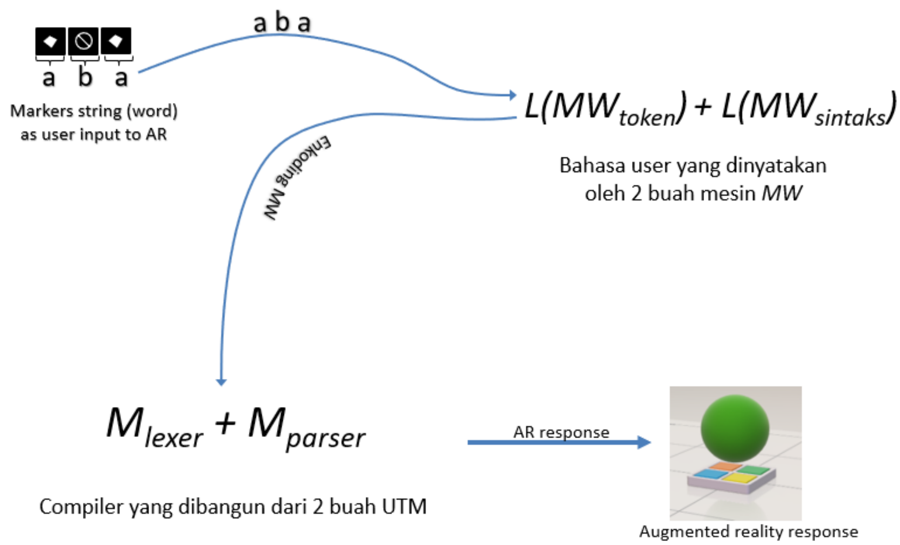
Akan tetapi, jika L_I memiliki gramatika, misal gramatika G maka di sana ada sebuah automata atau mesin Turing yang sepadan dengan L_I. Akan tetapi, telah ditunjukkan pada teorema 6.2 bahwa mesin Turing *MW* dapat dinyatakan sebagai mesin Turing standar sehingga mestilah L_I dapat dinyatakan oleh sebuah 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 $\mathrm{L(MW_{sintaks})}$, 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:

$$\textit{Compiler} = \mathrm{M_{lexer} + M_{parser}} \qquad (6.2)$$

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})

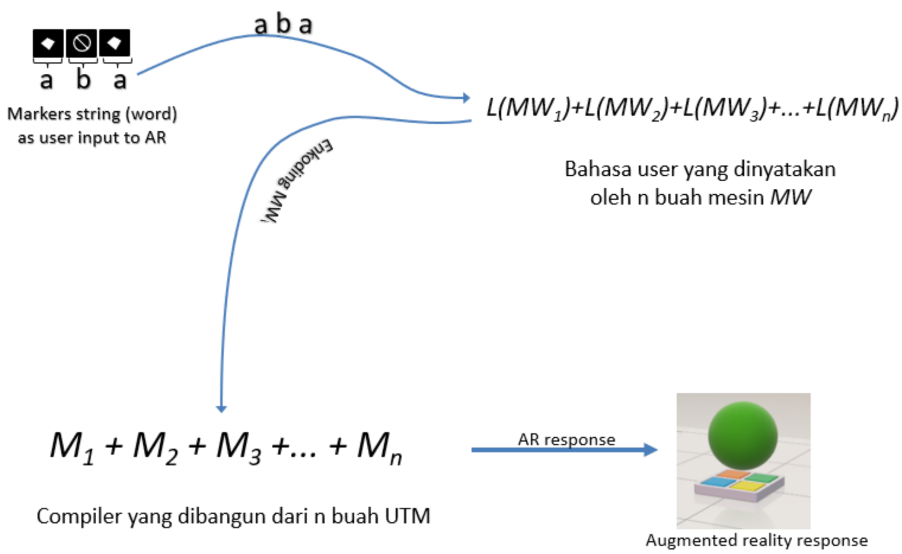
Akan tetapi, himpunan respon ARS dapat dianggap sebagai sebuah bahasa, walaupun dia hanya semata himpunan *word* (*word* yang mungkin mewakili ekspresi fungsi-fungsi *executable*) yang mungkin tanpa gramatika dan karena mesin Turing *M* adalah juga sebuah UTM (teorema 6.2.) maka mestilah sebuah algoritma 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:

$$Compiler = M_1 + M_2 + M_3 + ... + M_n \text{ dan } M_i \text{ mesin Turing } M \text{ ke-i.} \quad (6.4)$$

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

Kesimpulan Proposisi 3

Dari uraian argumentasi proposisi 3 di atas, uraian ini memberikan bahwa sebuah mesin Turing universal yaitu mesin *M* dapat dibuat sebuah *interpreter* dan atau sebuah *compiler* bagi bahasa-bahasa pengguna untuk ARS. Ini memberikan argumentasi dan pembuktian deduktif bagi konjektur 4 pada bab landasan teori sebelumnya.

Proposisi 3 ini juga memberikan bahwa bahasa pengguna terhadap ARS dapat dibangun mengunakan bahasa apapun dalam berbagai tipe bahasa Chomsky, baik itu bahasa reguler, bebas konteks, konteks sensitif atau rekursif, *unrestricted*. Ini karena mesin Turing *MW* yang merepresentasikan bahasa pengguna, yaitu L(MW) adalah dapat dinyatakan dalam mesin Turing standar dan karena dia mesin Turing standar maka bahasa apapun dalam berbagai tipe bahasa Chomsky adalah dapat direprepentasikan oleh mesin *MW*. Akan tetapi, juga bahwa mesin *M* adalah sebuah UTM (teorema 6.2) maka mesin *M* dapat menjadi *interpreter* atau *compiler* bagi bahasa apapun dari pengguna.

PENUTUP

Dari semua deskripsi di atas, telah dikemukakan dengan lengkap gagasan rancangan 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

Minsky, M. L. (1967) ‘Computation: Finite and Infinite Machines’, *ACM Classic Books Series*, p. 0. Available at: http://portal.acm.org/citation.cfm?id=1095587.

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.