DOKUMEN BUKTI HAK CIPTA

Konstruksi Mesin Real Time Compiler/Interpreter Untuk Bahasa Yang Diberi Atribut Waktu (Real Time Language) untuk Interaksi Manusia-Mesin (Tidak Terbatas pada Mesin Augmented Reality)

Pengusul Hak Cipta

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

Dr. Azhari, M.T

Dr. Suprapto, M.Kom

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2021

Konstruksi Mesin Real Time Compiler/Interpreter Untuk Bahasa Yang Diberi Atribut Waktu (Real Time Language) untuk Interaksi Manusia-Mesin (Tidak Terbatas pada Mesin Augmented Reality)

A. RINCIAN KLAIM MESIN COMPILER-INTERPRETER

Bagian ini merinci tata cara konstruksi mesin compiler/interpreter untuk bahasa yang diberi atribut waktu (*real time language*). Sebagai pendahuluan, konstruksi ini didahului dengan rincian konstruksi automata *generalized Finite State Automata* yang diberi atribut waktu *(timed-gFSA)*, kemudian Rincian itu adalah sebagai berikut:

Konstruksi Timed-gFSA

Pada bagian ini akan dikemukakan sebuah teori tentang bagaimana mengkonstruksikan sebuah *timed-gFSA* kemudian bagaimana menggunakannya untuk membangun sebuah bahasa *timed-mL(timed-gFSA)*. Sebuah timed-gFSA diusulkan sebagai berikut:

> ***Definisi 8.14 (timed-gFSA)*** *Sebuah timed-gFSA adalah sebuah tupel* $<\Sigma, Q, Q_0, \text{timed-}L_{\Sigma i}, C, E, F>$ *dan:*
> Σ *= himpunan berhingga simbol input*
> Q *= himpunan berhingga state*
> $Q_0 \subseteq Q$ *himpunan state awal*
> $\text{Timed-}L_{\Sigma i} = \{ (w,t) \mid (w,t)=(a_1,t_1)(a_2,t_2)(a_3,t_3)\ldots(a_n,t_n)\ ,\ a_j \in \Sigma i \subseteq \Sigma \text{ dan } t_1 \leq t_1 \leq t_1 \leq \ldots\ t_{n-1} \leq t_n \leq t\}$
> C *= himpunan berhingga clock*
> $E = Q \times Q \times L_{\Sigma i} \times 2^C \times \Phi(C)$ *adalah himpunan transisi dan setiap elemen E dinyatakan oleh* $<q, q', w, \lambda, \delta>$, $\lambda \subseteq C$ *menyatakan sejumlah clock yang perlu direset dan* $\delta \subseteq \Phi(C)$ *menyatakan sejumlah clock constrain bagi sejumlah clock.*
> $F \subseteq Q$ *adalah himpunan accept state.*

Dengan demikian, pengertian tentang bagaimana cara mengekspresikan *clock* dalam konsep ini adalah sepakat dengan definisi yang dikemukakan oleh Rajeev (Alur-Dill, 1994). Definisi ini (ditulis dalam bahasa inggris sesuai aslinya) tentang cara mengekspresikan *clock* ke dalam automata adalah sebagai berikut:

> ***Definisi 8.15*** *Untuk sebuah himpunan X dari variabel clock, himpunan* $\Phi(X)$ *clock constraints* δ *yang didefinisikan secara induktif oleh:*
> $\delta := x \leq c \mid c \leq x \mid \neg\delta \mid \delta \wedge \delta$
> *dan x adalah sebuah clock dalam X dan c adalah sebuah constant dalam Q.*

Berikut ini adalah sebuah contoh *timed-gFSA* yang dikonstruksikan menggunakan definisi 8.14. contoh itu adalah sebagai berikut:

Contoh timed-gFSA:

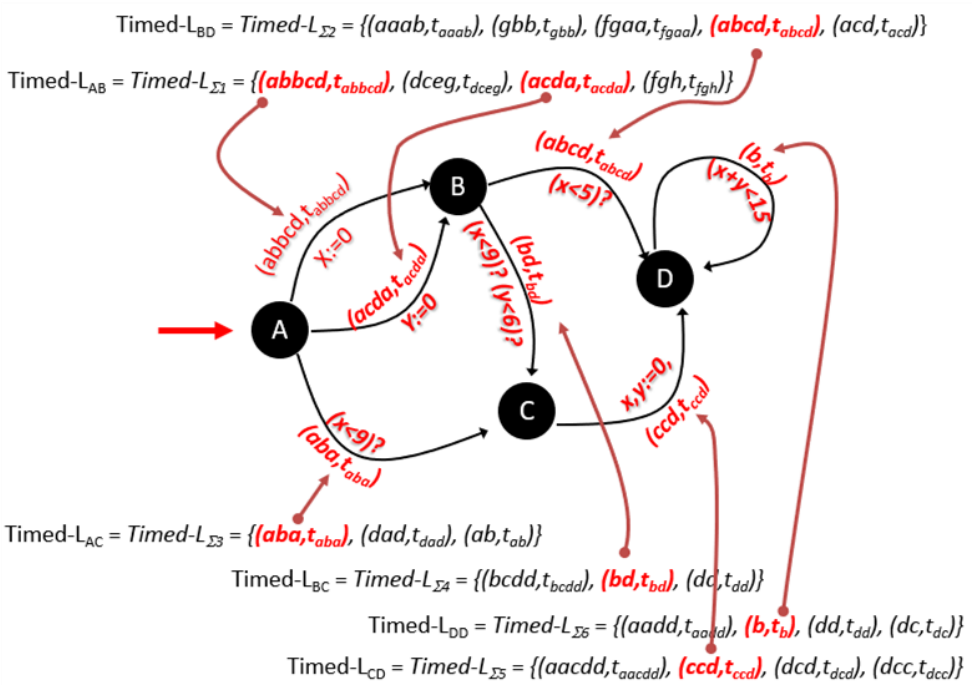
Sebuah timed-gFSA $<\Sigma, Q, Q_0, timed\text{-}L_{\Sigma i}, C, E, F>$ dan:

- $\Sigma = \{a, b, c, d, e, f, g, h\}$
- $Q = \{A, B, C, D\}$
- $Q_0 = \{A\} \subseteq Q$ himpunan *state* awal
- $Timed\text{-}L_{\Sigma 1} = \{(abbcd,t_{abbcd}), (dceg,t_{dceg}), (acda,t_{acda}), (fgh,t_{fgh})\}$
- $Timed\text{-}L_{\Sigma 2} = \{(aaab,t_{aaab}), (gbb,t_{gbb}), (fgaa,t_{fgaa}), (abcd,t_{abcd}), (acd,t_{acd})\}$
- $Timed\text{-}L_{\Sigma 3} = \{(aba,t_{aba}), (dad,t_{dad}), (ab,t_{ab})\}$
- $Timed\text{-}L_{\Sigma 4} = \{(bcdd,t_{bcdd}), (bd,t_{bd}), (dd,t_{dd})\}$
- $Timed\text{-}L_{\Sigma 5} = \{(aacdd,t_{aacdd}), (ccd,t_{ccd}), (dcd,t_{dcd}), (dcc,t_{dcc})\}$
- $Timed\text{-}L_{\Sigma 6} = \{(aadd,t_{aadd}), (b,t_b), (dd,t_{dd}), (dc,t_{dc})\}$
- $C = \{x,y\}$
- $E = Q \times Q \times timed\text{-}L_{\Sigma i} \times 2^C \times \Phi(C)$ adalah himpunan transisi dan setiap elemen E dinyatakan oleh $<q, q', (w,t), \lambda, \delta>$, $\lambda \subseteq C$ menyatakan sejumlah clock yang perlu direset dan $\delta \subseteq \Phi(C)$ menyatakan sejumlah *clock constrain* bagi sejumlah *clock*, yaitu:

  $E = \{<A, B, (abbcd,t_{abbcd}), (x:=0),\varnothing>, <A, B, (acda,t_{acda}), (y:=0),\varnothing>, <A, C,$ $(aba,t_{aba}), \varnothing, (x<9)>, <C, D, (ccd,t_{ccd}), (x,y:=0), \varnothing>, <B, C, (bd,t_{bd}), \varnothing, \{(x<9),$ $(y<6)\}>, <B, D, (abcd,t_{abcd}), \varnothing, (x<5)>, <D, D, (b,t_b), \varnothing, (x+y<15)>$ }.
- $F \subseteq Q$ adalah himpunan *accept state*.

Ilustrasi untuk konstruksi *timed-gFSA* ini adalah sebagaimana diberikan pada Gambar 8.2. Gambar ini mengilustrasikan sebuah bahasa *real time* yang bahasa tersebut diterima oleh sebuah *timed-gFSA*. bahasa ini ditulis sebagai *timed-mL(timed-gFSA)*.

Pada penerapnnya, pengguna yang berinteraksi dengan sebuah ARS dapat membangun bahasanya secara *real timed* dengan menggunakan sebuah *timed-gFSA*.



Gambar 8.2. Contoh Timed-gFSA yang Digunakan untuk Mengkonstruksikan Sebuah Timed-mL (Timed-gFSA)

Timed-gFSA yang dikonstruksikan menghasilkan suatu bahasa *timed-multilanguage* (*timed-mL*) yang merupakan konkatenasi dari beberapa *timed-L*. Konkatenasi itu dinyatakan sebagai berikut:

Timed-mL(timed-gFSA)=timed-L$_{AD}$

Timed-L$_{AD}$ = (timed-L$_{AB}$ o *timed-L$_{BD}$* o *timed-L$_{DD}$*... o *timed-L$_{DD}$)*$\cup$*(timed-L$_{AC}$* o *timed-L$_{CD}$* o *timed-L$_{DD}$*... o *timed-L$_{DD}$)* $\cup$ *(timed-L$_{AB}$* o *timed-L$_{BC}$* o *timed-L$_{CD}$* o *timed-L$_{DD}$*... o *timed-L$_{DD}$)* $\cup$ *(timed-L$_{AB}$* o *timed-L$_{BD}$)*$\cup$*(timed-L$_{AC}$* o *timed-L$_{CD}$)* $\cup$ *(timed-L$_{AB}$* o *timed-L$_{BC}$* o *timed-L$_{CD}$)*

Walaupun bahasa *timed-L* yang dikonkatenasi dapat berasal dari berbagai tipe bahasa Chomsky (tipe regular, bebas konteks, konteks sensitif, *unrestricted*), tetapi mereka dikonkatenasi dalam automata reguler *timed-gFSA* sehingga bahasa konkatenasi yang dihasilkannya adalah selalu merupakan bahasa reguler.

Berikut ini adalah sebuah contoh konstruksi timed-gFSA menggunakan definisi 8.14 yang lebih sederhana daripada contoh pertama di atas.

Konstruksi:

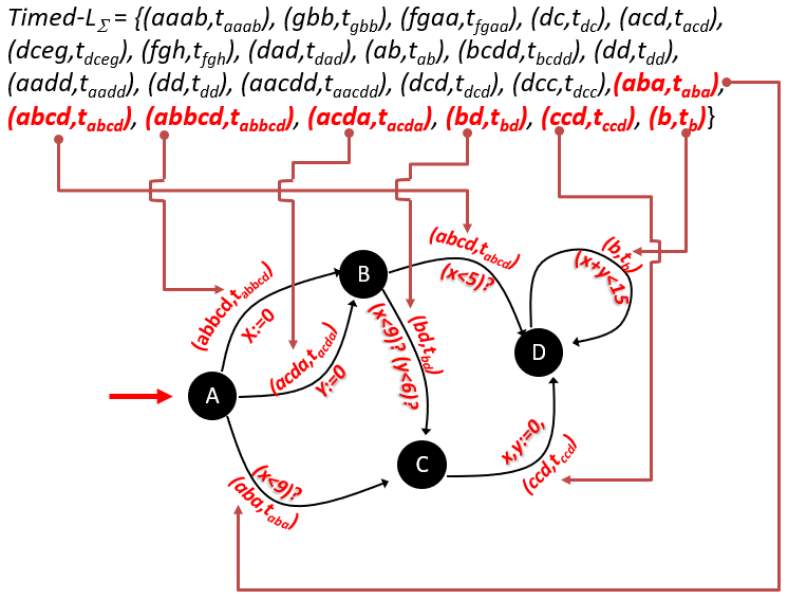
Sebuah timed-gFSA <Σ, Q, Q_0, *timed-L$_\Sigma$*, C, E, F> dan

- $\Sigma = \{a, b, c, d, e, f, g, h\}$

- $Q = \{A, B, C, D\}$
- $Q_0 = \{A\} \subseteq Q$ himpunan *state* awal
- *Timed-*L_Σ = $\{(abbcd,t_{abbcd}), (dceg,t_{dceg}), (acda,t_{acda}), (fgh,t_{fgh}), (aaab,t_{aaab}), (gbb,t_{gbb}), (fgaa,t_{fgaa}), (abcd,t_{abcd}), (acd,t_{acd}), (aba,t_{aba}), (dad,t_{dad}), (ab,t_{ab}), (bcdd,t_{bcdd}), (bd,t_{bd}), (dd,t_{dd}), (aacdd,t_{aacdd}), (ccd,t_{ccd}), (dcd,t_{dcd}), (dcc,t_{dcc}), (aadd,t_{aadd}), (b,t_b), (dd,t_{dd}), (dc,t_{dc})\}$
- $C = \{x,y\}$
- $E = Q \times Q \times$ *timed-*$L_\Sigma \times 2^C \times \Phi(C)$ adalah himpunan transisi dan setiap elemen E dinyatakan oleh $\langle q, q', (w,t), \lambda, \delta \rangle$, $\lambda \subseteq C$ menyatakan sejumlah clock yang perlu direset dan $\delta \subseteq \Phi(C)$ menyatakan sejumlah *clock constrain* bagi sejumlah *clock*, yaitu:

  $E = \{\langle A, B, (abbcd,t_{abbcd}), (x:=0), \varnothing \rangle, \langle A, B, (acda,t_{acda}), (y:=0), \varnothing \rangle, \langle A, C, (aba,t_{aba}), \varnothing, (x<9) \rangle, \langle C, D, (ccd,t_{ccd}), (x,y:=0), \varnothing \rangle, \langle B, C, (bd,t_{bd}), \varnothing, \{(x<9), (y<6)\} \rangle, \langle B, D, (abcd,t_{abcd}), \varnothing, (x<5) \rangle, \langle D, D, (b,t_b), \varnothing, (x+y<15) \rangle \}$.
- $F \subseteq Q$ adalah himpunan *accept state*.

Gambar 8.3. mengilustrasikan konstruksi *timed-ml(timed-gFSA)* yang didefinisikan sebelumnya.



Gambar 8.3. Sebuah Contoh Timed-mL (Timed-gFSA) yang Lebih Sederhana dari Contoh di Gambar 8.2

Timed-gFSA yang dikonstruksikan menghasilkan suatu bahasa *timed-multilanguage* yang merupakan konkatenasi dari *Timed-*L_Σ dengan dirinya sendiri. *Timed-multilanguage* tersebut dinyatakan sebagai berikut:

Timed-mL(timed-gFSA) = Timed-L

Dengan demikian, teori ini tentang bagaimana dia diterapkan pada sistem *augmented reality* adalah dikemukakan sebagai berikut:

Penerapan pada Sistem Augmented Reality

Untuk menerapkan kepada sistem *augmented reality*, pada himpunan penanda (*marker*) dapat didefinisikan Σ sebagai:

$$\Sigma = \{\blacklozenge, \oslash, \blacktriangle, \blacksquare, \blacktriangledown, \square, \boxtimes, \blacktriangledown\}$$

Pada Σ, setiap penanda adalah sebuah alphabet sehingga bahasa yang dibentuk oleh *Timed-*L_Σ adalah sebuah bahasa penanda *real time* untuk sistem *augmented reality*, selanjutnya dapat digunakan definisi Σ di atas untuk rekonstruksi contoh pertama dan contoh kedua di atas, sebagai penerapan kepada sistem *augmented reality*. Rekonstruksi contoh pertama dan contoh kedua menghasilkan sebuah bahasa interaksi yang *real time* antara sistem AR dengan pengguna.

Setelah usulan gagasan *timed-gFSA* ini, penjelasan berikutnya akan diusulkan gagasan sebuah *interpreter* atau *compiler* yang bisa menerima dan menerjemahkan (mensimulasikan) sebuah bahasa *timed-mL(timed-gFSA)* yaitu sebuah UTM untuk *timed-gFSA*. Pada penjelasan berikut juga akan dikemukakan gagasan *timed-gTM* (cara konstruksi gTM dan beberapa teorema tentangnya telah dikemukakan pada subpasal sebelumnya dalam bab ini), yaitu sebuah generalisasi mesin Turing mengikuti atau analogi cara generalisasi FSA menjadi gFSA kemudian menjadi *timed-gFSA* yang diharapkan bahwa *timed-gTM* dapat menghasilkan sebarang bahasa yang bukan saja terbatas pada bahasa regular tetapi dapat menghasilkan bahasa bebas konteks, konteks sensitif, bahkan *unrestricted* yang *real time*.

Konstruksi UTM untuk Timed-gFSA (Konstruksi compiler/interpreter)

Bagian ini memuat usulan gagasan yaitu sebuah *universal Turing machine* yang dapat mensimulasikan sebuah timed-gFSA, yaitu sebuah UTM yang dapat bertindak sebagai *compiler* atau *interpreter* antara pengguna dan sistem *augmented reality*.

Langkah pertama akan dikonstruksikan sebuah UTM untuk timed-gFSA pada contoh pada Gambar 8.2 dan 8.3. sebelumnya. Untuk penyederhanaan, pada konstruksi tidak dilakukan enkoding instruksi, terkecuali pada contoh timed-gFSA yang lebih rumit direncanakan berikutnya.

Timed-gFSA pada contoh pada Gambar 8.2 dan 8.3. dapat diterjemahkan sebagai semata himpunan instruksi yang diberi clock sebagai berikut:

1. *Timed-gFSA = {*

   $q_1(A, (abbcd,t_{abbcd}), (x:=0),\varnothing)=B,$

   $q_2(A, (acda,t_{acda}), (y:=0),\varnothing)=B,$

   $q_3(A, (aba,t_{aba}), \varnothing, (x<9))=C,$

   $q_4(C, (ccd,t_{ccd}), (x,y:=0), \varnothing)=D,$

   $q_5(B, (bd,t_{bd}), \varnothing, \{(x<9), (y<6)\})=C,$

   $q_6(B, (abcd,t_{abcd}), \varnothing, (x<5))=D,$

   $q_7(D, (b,t_b), \varnothing, (x+y<15))=D$

   }

2. Untuk memudahkan pembacaan instruksi pada pita UTM, susunan instruksi *timed-gFSA* pada point 1 di atas dirubah dengan menukar posisi *reset* dan *guard* sebagai berikut:

   Timed-gFSA = {

   $q_1(A, (abbcd,t_{abbcd}), \varnothing, (x:=0))=B,$

   $q_2(A, (acda,t_{acda}), \varnothing, (y:=0))=B,$

   $q_3(A, (aba,t_{aba}), (x<9), \varnothing)=C,$

   $q_4(C, (ccd,t_{ccd}), \varnothing, (x,y:=0))=D,$

   $q_5(B, (bd,t_{bd}), \{(x<9), (y<6)\}, \varnothing)=C,$

   $q_6(B, (abcd,t_{abcd}), (x<5), \varnothing)=D,$

   $q_7(D, (b,t_b), (x+y<15), \varnothing)=D$

   }

3. Konstruksi UTM:

   Assumsi mesin:

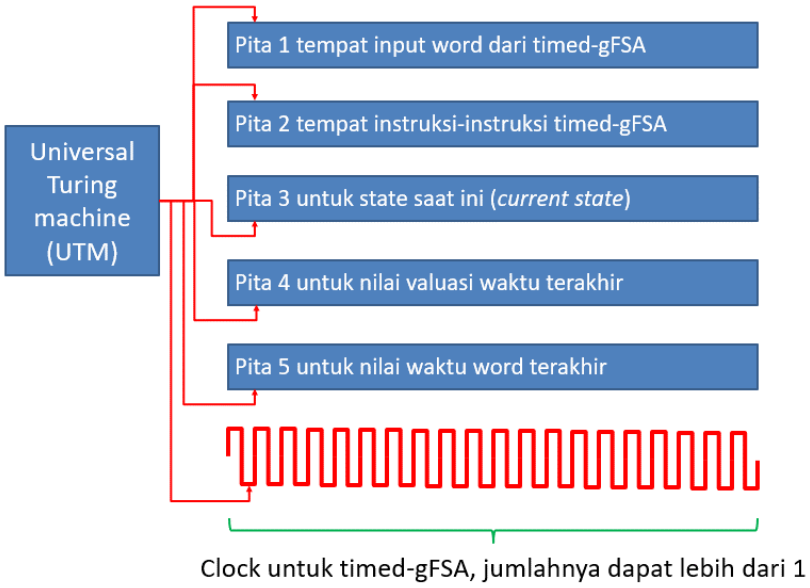
   a. Mesin memiliki pita input yang memisahkan setiap *word* dengan spasi atau kosong.
   b. Mesin memiliki pita yang menyimpan instruksi-instruksi *timed-gFSA* yang akan disimulasikan.
   c. Mesin memiliki pita-pita yang menyimpan nilai valuasi waktu untuk setiap *clock*.
   d. Mesin memiliki pita yang menyimpan *time* dari *word* yang telah terbaca paling akhir sebelumnya.
   e. Mesin memiliki pita yang menyimpan *current state*.

f. Mesin membuat sejumlah clock yang sesuai dengan clock pada instruksi yang hnedak disimulasikan.

g. Mesin hanya menerima *timed*-gFSA yang deterministik.

4. Fisik mesin UTM

Gambar 8.4. mengilustrasikan bentuk fisik dari mesin UTM yang dimaksud.



Gambar 8.4. Fisik mesin UTM untuk Timed-gFSA

5. Algoritma mesin UTM:

Inisialisasi mesin:

a. Step 1: UTM menulis semua input *word* untuk *timed-gFSA* pada pita 1. setiap *word* dipisah oleh spasi (atau sel blank).

b. Step 2: UTM menulis semua instruksi *timed-gFSA* pada pita 2.

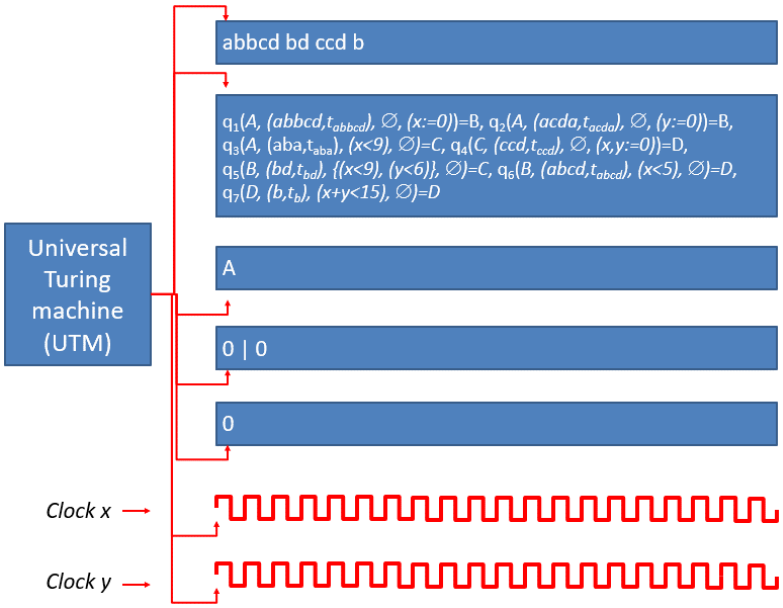
c. Step 3: UTM menulis start *state* pada pita 3.

d. Step 4: UTM menyiapkan n buah sel untuk n buah *clock*, secara berturut pada pita 4 dan mengisinya masing-masing dengan nilai 0.

e. Step 5: UTM menulis *time* atau nilai waktu awal dari *word* sebagai 0 pada pita 5.

f. Step 6: UTM mereset semua *clock* pada posisi 0 dan bersiap untuk berdetak.

Inisialisasi ini diilustrasikan pada Gambar 8.5. berikut yang digunakan contoh pada Gambar 8.2 dan 8.3. konstruk *timed-gFSA*. Contoh pada Gambar 8.2 dan 8.3. konstruksi *timed-gFSA* diharapkan dapat disimulasikan oleh UTM yang telah dibuat sebelumnya.



Gambar 8.5 Ilustrasi Inisialisasi UTM dengan 2 Clock (*x* dan *y*)

Simulasi mesin:

a. Step 0: UTM menjalankan *clock x* dan *clock y* (berdetak).

b. Step 1: UTM membaca input pada pita 1 secara 1 *word*, yaitu membaca seluruh karakter hingga spasi dan membaca pita 3. Jika *head* pada pita 1 membaca kosong dan *current state* pada pita 3 adalah *accept state* maka UTM HALT, tetapi jika *current state* bukan *accept state* maka UTM FAIL dan HALT. Jika tidak kosong maka lanjut ke step 2.

c. Step 2: UTM membaca pita 2, *searching* untuk mencari instruksi dengan *state-nya adalah current state* pada pita 3 dan string inputnya adalah *word* yang terbaca pada pita 1. Jika tidak *match* dengan satupun instruksi yang demikian maka UTM FAIL dan HALT. Jika *match* maka lanjut ke step 3.

d. Step 3: UTM lalu membaca *time t* pada *word* di instruksi lalu memeriksa apakah nilai *clock x* berjalan ditambah nilai *time* pada pita 5 dikurang nilai valuasi waktu *x* pada pita 4 adalah lebih kecil atau sama dengan *time t*? Jika tidak maka UTM FAIL dan HALT. Jika ya maka periksa lagi untuk waktu *clock y*, jika nilai *clock y* berjalan ditambah nilai *t* pada pita 5 dikurang nilai valuasi waktu *y* pada pita 4 adalah lebih kecil atau sama dengan nilai *t* dari *word*? Jika tidak maka UTM FAIL dan HALT. Jika ya maka lanjut step 4.

e. Step 4: UTM lalu membaca *guard* waktu pada instruksi, jika tidak kosong maka UTM memeriksa *clock*, jika waktu *clock x* atau *clock y* yang sedang berjalan tidak memenuhi *guard*

(salah satu atau kedua *clock* tidak memenuhi) maka UTM FAIL dan HALT, jika kosong atau memenuhi maka lanjut ke step 5.

f. Step 5: UTM lalu membaca *reset* instruksi, jika kosong maka lanjut ke step 6, jika tidak kosong maka UTM mereset *clock* yang dinyatakan untuk direset lalu menjalankan ulang *clock* tersebut kemudian lanjut ke step 6.

g. Step 6: UTM mengeksekusi instruksi dan hasil eksekusi ditulis pada pita 3 menimpa *current state* sebelumnya. *Head* pita 1 bergerak ke kanan satu langkah, UTM membaca *clock* lalu nilai *clock x* dan nilai *clock y* terakhir disimpan berturutan pada pita 4 menimpa valuasi nilai *clock* sebelumnya. *Head* pita 2 dan pita 4 dikembalikan ke posisi paling awal. UTM menulis nilai *t* dari *word* ke pita 5 menimpa nilai sebelumnya. Lanjut step 7.

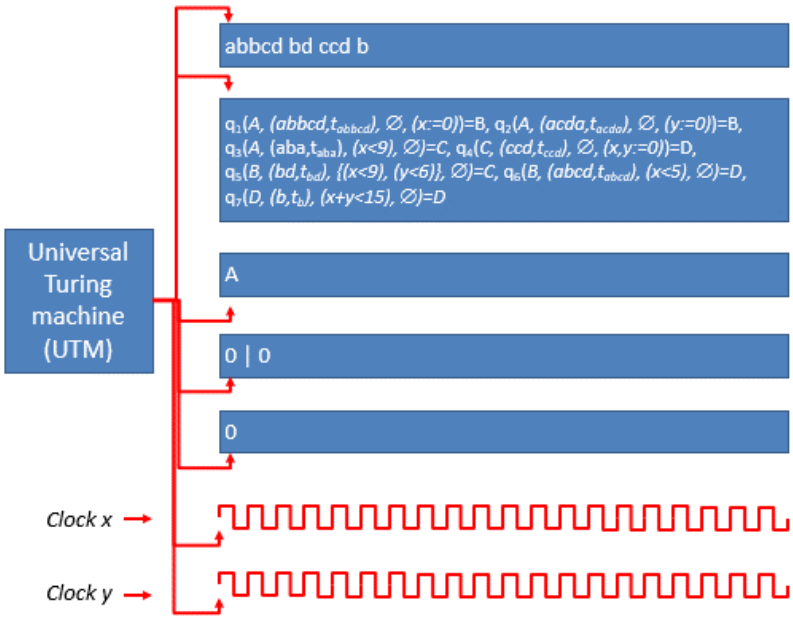
h. Step 7: Kembali ke step 1.

i. Selesai.

Berikut ini diberikan contoh kerja mesin UTM yang dibangun di atas. Pada contoh ini, mesin UTM bekerja untuk mensimulasikan *timed-gFSA* yang dibuat pada Gambar 8.2 dan 8.3.

1. Inisialisasi mesin:

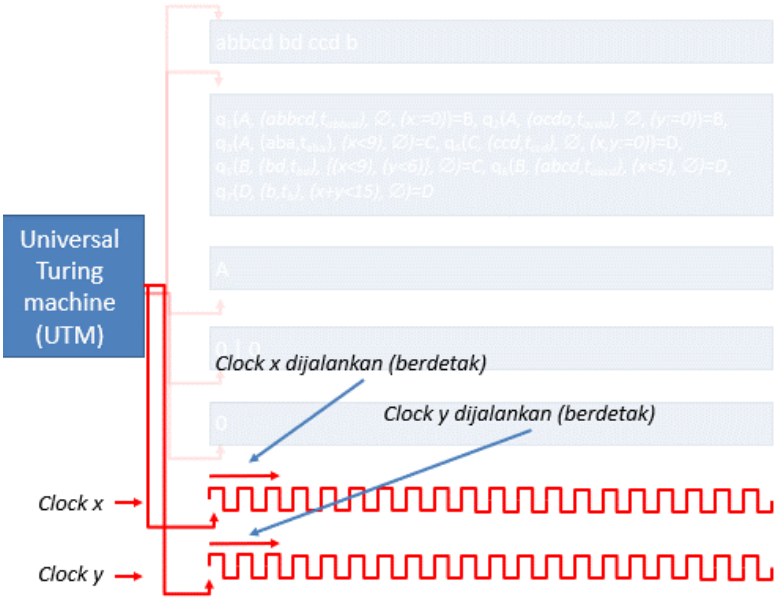
Gambar 8.6. mengilustrasikan step 0, 1, 2, 3, 4, 5 dan 6 pada langkah inisialisasi mesin UTM.



Gambar 8.6. Tahap Inisialisasi Mesin

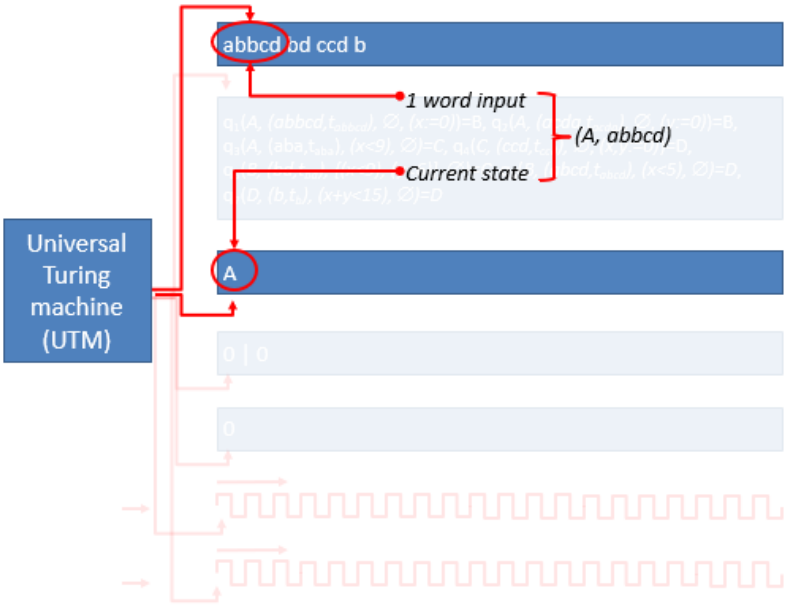
2. Simulasi mesin:

a. Step 0: UTM menjalankan *clock x* dan *clock y* (berdetak). Gambar 8.7. mengilustrasikan step 0.



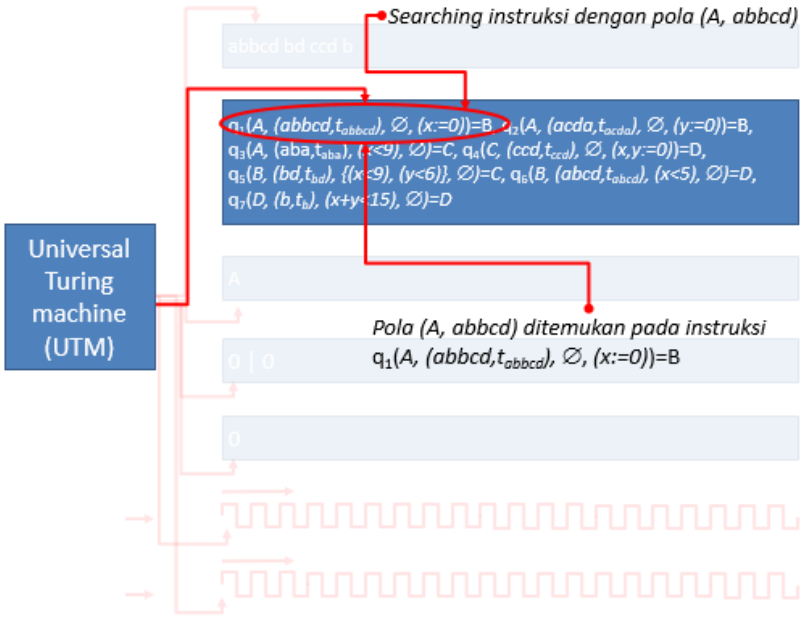
Gambar 8.7. Simulasi Step 0

b. Step 1: UTM membaca input pada pita 1 secara 1 *word*, yaitu membaca seluruh karakter hingga spasi...dan seterusnya. Gambar 8.8. mengilustrasikan step 1.



Gambar 8.8. Simulasi Step 1

c. Step 2: UTM membaca pita 2, *searching*...dan seterusnya. Gambar 8.9. mengilustrasikan step 2.

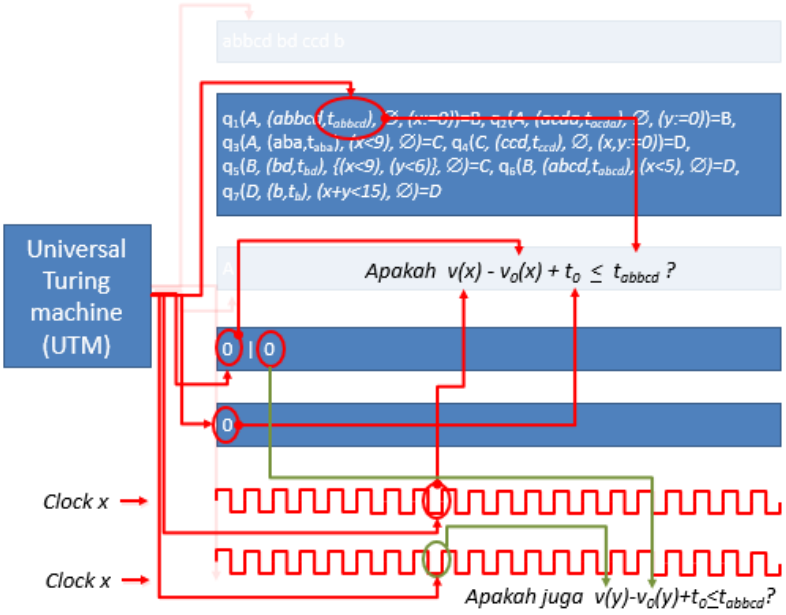


Gambar 8.9. Simulasi Step 2

d. Step 3: UTM lalu membaca *time t* pada *word* di instruksi.

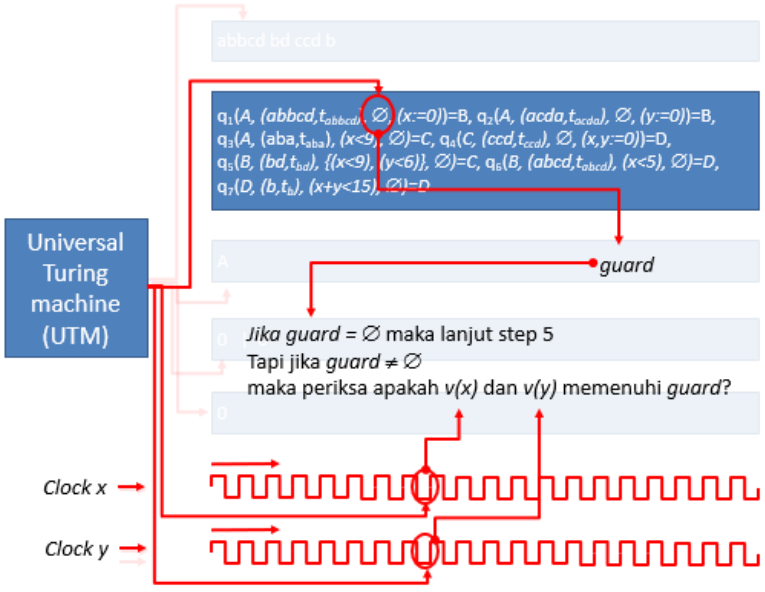
Pada step ini nilai *clock x* dan *y* sekarang (*current time*) ditulis *v(x)* dan *v(y)*. Nilai *clock x* dan *y* yang terbaca terakhir sebelumnya (*previous time*) dan tercatat pada pita 4 ditulis sebagai $v_o(x)$ dan $v_o(y)$. Hubungan antara semua nilai *clock* itu terhadap nilai waktu pada *timed-word* atau *timed-multiword* adalah selisih waktu eksekusi antar instruksi sebelumnya dengan instruksi berikut pada mesin UTM haruslah lebih kecil atau sama dengan selisih waktu antar *timed-word* yang dieksekusi $(v(x) - v_o(x)) \leq (t_{abbcd} - t_o)$ sebagai waktu yang normal untuk mengeksekusi instruksi, ini berarti mesin UTM harus memeriksa batasan berikut: $v(x) \leq v_o(x)+t_{abbcd} - t_o$ atau $v(x)- v_o(x)+t_o \leq t_{abbcd}$.

Proses pemeriksaan ini ditulis sebagai satu langkah dalam algoritma UTM, yaitu seperti tertulis pada step 3 algoritma UTM. Gambar 8.10. mengilustrasikan bagaimana step 3 melaksanakan pengecekan waktu pada *clock* mesin dan input *word* pada instruksi *timed-gFSA*.



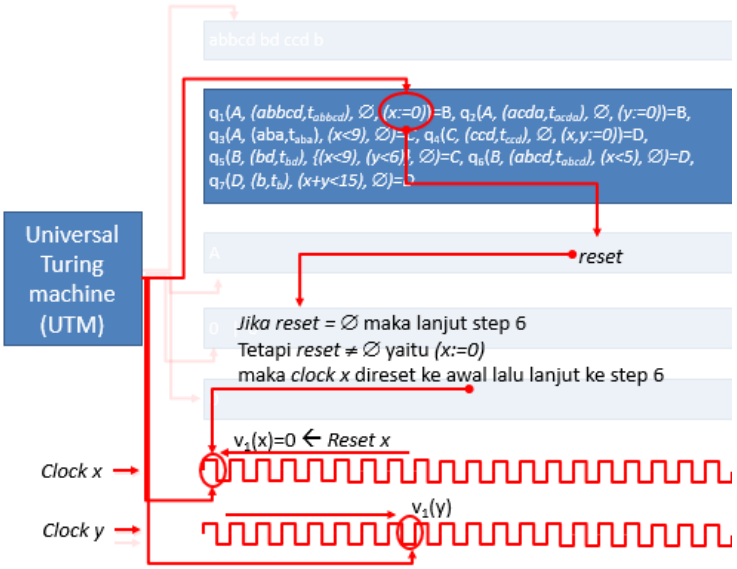
Gambar 8.10. Simulasi Step 3

e. Step 4: UTM lalu membaca *guard* waktu pada instruksi...dan seterusnya. Gambar 8.11. mengilustrasikan step 4.



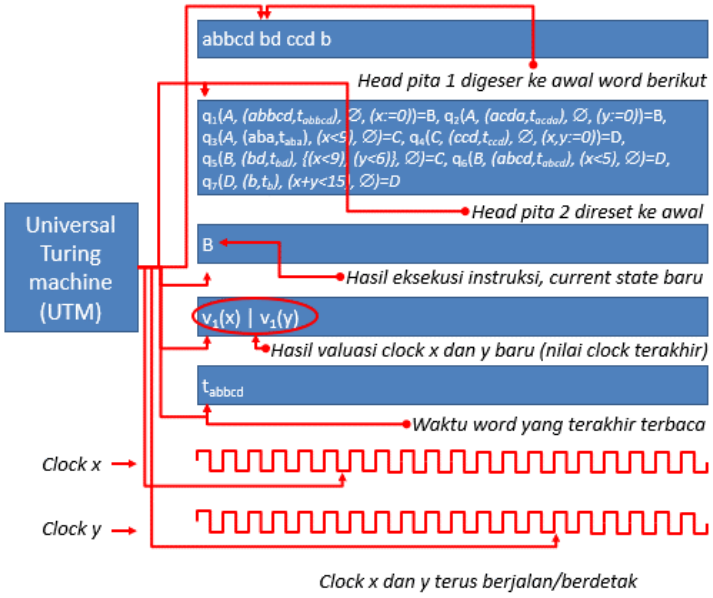
Gambar 8.11. Simulasi Step 4

f. Step 5: UTM lalu membaca *reset* instruksi...dan seterusnya. Gambar 8.12. mengilustrasikan step 5.



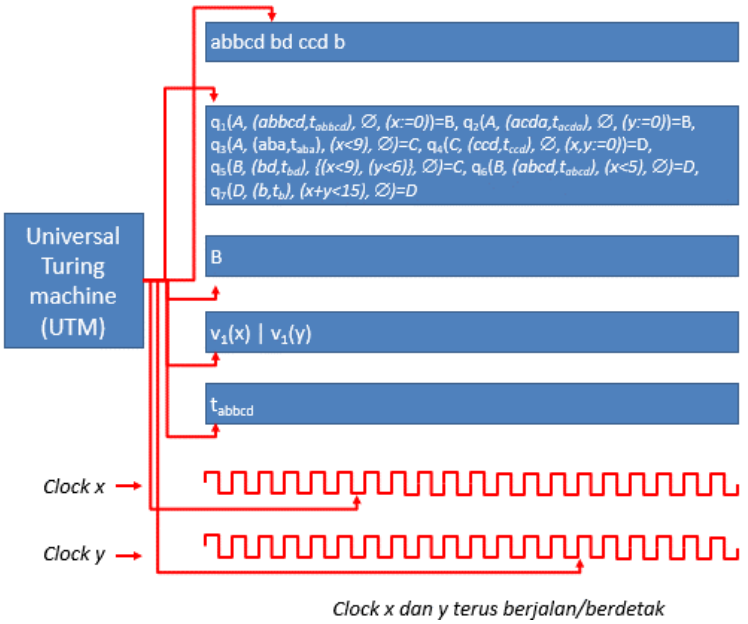
Gambar 8.12. Simulasi Step 5

g. Step 6: UTM mengeksekusi instruksi...dan seterusnya. Gambar 8.13. mengilustrasikan step 6.



Gambar 8.13. Simulasi Step 6

h. Step 7: Ulangi step 1 untuk konfigurasi mesin berikut ini. Gambar 8.14. mengilustrasikan step 7.



Gambar 8.14. Simulasi Step 7

Sampai pada tahap ini, sebuah UTM untuk *timed-gFSA* telah selesai di demonstrasikan. Berikut ini akan dikemukakan konstruksi UTM yang formal, yaitu konstruksi yang dilakukan enkoding terhadap mesin yang disimulasikan. Untuk mencakup keluasan dari kerja UTM, UTM direncanakan dikonstruksikan tidak hanya untuk *timed-gFSA* tetapi langsung dibuat untuk mensimulasikan *timed-gTM*, sebagai bentuk mesin yang lebih luas sehingga dengan sendirinya dapat dianggap juga dapat mensimulasikan *timed-gFSA*.

Konstruksi Timed-generalized Turing Machine (Timed-gTM)

Pada bagian ini akan dikemukakan sebuah konstruksi *timed-gTM*. Sebagai langkah berikut adalah pembuatan *timed-gTM*. Langkah ini diharapkan dapat menjadi fondasi untuk membangun bahasa-bahasa *real time* untuk interaksi pengguna-mesin (pengguna-ARS) yang tidak terbatas pada tipe bahasa reguler, tetapi kepada semua tipe bahasa lain dalam hireraki Chomsky (bebas konteks, konteks sensitif, rekursif dan *recursively enumerable*).

Berikut ini diusulkan sebuah definisi untuk sebuah *timed-gTM*.

Definisi 8.15 (timed-gTM) *Sebuah timed-gTM adalah sebuah 7-tuple* $<Q, \Sigma, \{\text{timed-}L_{\Sigma i} | i=1,2,3,..\}, q_0, C, \delta, F>$ *dan:*
Q *adalah himpunan berhingga state*
$F \subseteq Q$ *adalah himpunan state accept*
q_0 *adalah state awal*
$\Sigma i \subseteq A$ *dan* A *adalah alphabet*
$\Sigma = A \cup \{\emptyset\}$ *adalah himpunan simbol pita*
$L_{\Sigma i} = \{ w \mid w=a_1a_2a_3... a_n , a_j \in \Sigma i$ *dan* $j=1,2,3...n\}$

Timed-$L_{\Sigma i}$ = { (w,t) | (w,t)=(a_1,t_1)(a_2,t_2)(a_3,t_3) ...(a_n,t_n) , $a_j \in \Sigma_i$ dan $t_1 \leq t_1 \leq t_1 \leq \ldots$ $t_{n-1} \leq t_n \leq t$}
C = himpunan berhingga clock
δ: Q x timed-$L_{\Sigma i} \Rightarrow$ Q x $L_{\Sigma j}$ x $\mathbb{Z}$ x Φ(C) x Reset(2^C)
Φ(C) = himpunan quard atau constrain bagi clock.
Reset(2^C) himpunan reset bagi clock.
$\mathbb{Z}$ adalah himpunan bilangan bulat yang menyatakan perpindahan head di pita.
$F \subseteq Q$ adalah himpunan accept state.
Hanya memiliki 1 pita, head membaca pita dari kiri ke kanan dan head menulis dari kiri ke kanan.

Enkoding untuk *timed-gTM*:

Langkah berikutnya adalah mempersiapkan sebuah UTM secara formal bagi *timed-gTM* sehingga sebuah *interpreter* atau *compiler* untuk bahasa *timed-mL(timed-gTM)* dapat dibangun maka enkoding *timed-gTM* menjadi sebuah barisan kode yang dapat diumpankan kepada pita UTM adalah diperlukan. *Timed-gTM* dan inputnya di enkode ke alphabet *A* = {a, b, c, d, e, f, g, ..., y, z, A, B, C, D ..., Z, 1,2,3,4 ..., 9, + _) (*, &, %, $, £, @, <, >, =, ≤, ≥, (,),...}, yang menyatakan semua karakter pada keyboard digabung simbol {≤, ≥,}. Enkoding ini adalah deskripsi dari *timed-gTM* dan deskripsi input dari *timed-gTM* yang kesemuanya disuplai masuk kedalam UTM. Enkoding itu adalah sebagai berikut:

- Seluruh *state* dari *timed-gTM* dipetakan ke integer *0, 1, ... k*, Pemetaan dilakukan secara fungsi satu ke satu (*bijective*) dan *state* awal adalah $q_0 \equiv 0$, dan *state accept* adalah *f, f + 1, ... k* untuk $0 < f \leq k$ sehingga $Q = \{0,1,2, 3, ...k\}$, $q_0 = 0$ *dan* $F = \{f, f + 1, ...k\}$, $k \geq 0, f \leq k$.
- *Word* yang merupakan input dari *timed-gTM* di enkoding ke dalam $A \cup \{\varnothing\}$ secara *bijective* sehingga diperoleh sebuah bahasa yang dienkoding yaitu $L_{encode\Sigma i}$ = encode($L_{\Sigma i}$) = { b | b= $b_1b_2b_3 \ldots b_n$, b_j = encode(a_j), $a_j \in \Sigma i$, $b_j \in A \cup \{\varnothing\}$}
- *timed-$L_{encode\Sigma i}$ = encode(timed-$L_{\Sigma i}$) = { (b,t) | (b,t)= ($b_1b_2b_3 \ldots b_n$,t) = (b_1,t_1)(b_2,t_2)(b_3,t_3) ...(b_n,t_n)}* *dan* (b_j, t_j) = (encode(a_j), t_j) , (a_j,t_j) $\in$ *timed-$L_{\Sigma i}$* , $b_j \in A \cup \{\varnothing\}$.
- *Clock* tidak dienkoding atau tetap ke dalam dirinya sendiri, misal *clock x* dan *y, enkoding(x) = x* dan *enkoding(y) = y demikian* juga semua relasi aritmetika dalam *guard (constrain) dan reset* dienkoding ke dirinya sendiri.

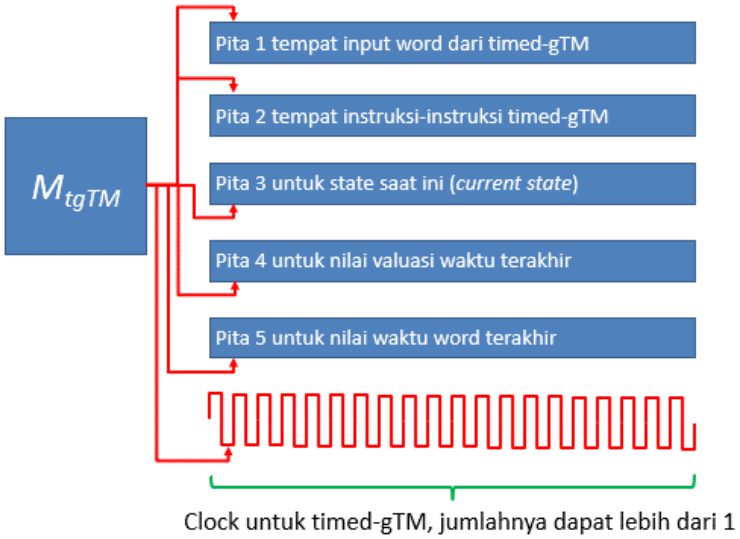
- Fungsi transisi yang dinyatakan sebagai *δ-function: δ(q, (s,t)) = (q', s', d, guard, reset)* dan *(s,t) ∈timed-$L_{encode\Sigma i}$ dan s'∈$L_{\Sigma j}$, 0≤ q ≤f , d ∈Z, 0 ≤q'≤k, guard* adalah *constrain clock* dan reset adalah reset *clock.*
- Langkah berikut fungsi transisi *δ* di dalam mesin UTM dinyatakan dalam 7-tuples *(q, (s, t), q', s', d, guard, reset).*
- Deskripsi *timed-gTM* ke dalam pita UTM adalah ditulis dalam urutan sebagai berikut: *k, f,* δ_1, δ_2, δ_3, ...dan *k* adalah jumlah *state*, *f* adalah *state accept* pertama dan δ_1 adalah *(q,(s,t),q',s',d,guard,reset)*. Ini memiliki arti bahwa enkoding(*timed-gTM*) = *k, f,* $(q_1,(s_1,t_1),q_1',s_1',d_1,guard_1,reset_1)$, ..., $(q_n, (s_n, t_n), q_n', s_n', d_n, guard_n, reset_n)$ dan *n, f, q, q', d* adalah bilangan bulat dan berlaku: *0≤f, q <f, q '≤k dan d ∈{-m, 0, n}, s, s' ∈*$L_{encode\Sigma i}$
- Nilai valuasi waktu setiap *clock* tidak dienkoding atau tetap ke dirinya sendiri.
- Nilai waktu setiap *word* input juga dienkoding dirinya sendiri.

Langkah berikutnya akan dikonstruksikan sebuah UTM untuk *timed-gTM* yang telah dienkoding.

Konstruksi UTM untuk Timed Generalized Turing Machine (timed-gTM)

Mesin Turing yang digunakan untuk menjadi sebuah mesin Turing universal (UTM) bagi *timed-gTM* adalah sebuah mesin Turing yang memiliki 5 buah pita dan beberapa *clock* yang banyaknya menuruti jumlah *clock* yang dimiliki oleh *timed-gTM*. Sebut mesin ini sebagai mesin M_{tgTM}. Deskripsi mesin adalah sebagai berikut:

1. Bentuk fisik secara umum mesin M_{tgTM}: Gambar 8.15. mengilustrasikan bentuk fisik mesin M_{tgTM}.



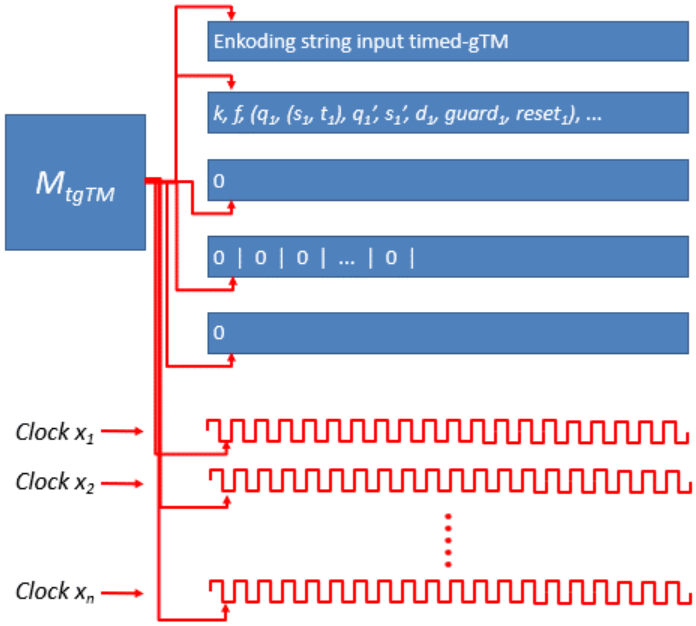
Gambar 8.15. Bentuk Mesin M_{tgTM}

2. Algoritma mesin M_{tgTM}:

Inisialisasi mesin M_{tgTM}:

- Step 1: M_{tgTM} menulis 0 pada sel pertama pita 3. (0 adalah *state* start)
- Step 2: M_{tgTM} menulis seluruh hasil enkoding *word* input untuk *timed-gTM* pada pita 1, mulai pada sel pertama, setiap *word* dipisah oleh spasi atau ∅.
- Step 3: M_{tgTM} menulis seluruh enkoding *timed-gTM* pada pita 2, mulai pada sel pertama.
- Step 4: M_{tgTM} menulis semua nilai valuasi waktu pertama untuk setiap *clock* beturut-turut sel ke sel pada pita 4.
- Step 5: M_{tgTM} menulis nilai 0 pada sel pertama pita 5.
- Step 6: M_{tgTM} membuat sejumlah *clock* yang dibutuhkan sesuai dengan jumlah *clock* pada *timed-gTM*.
- Step 7: M_{tgTM} memposisikan seluruh *head* pada seluruh pita di sel pertama.
- Step 8: M_{tgTM} memposisikan mereset seluruh *clock* pada posisi pertama (nilai waktu direset ke 0)

Ilustrasi inisialisasi mesin M_{tgTM} ini adalah sebagaimana Gambar 8.16.



Gambar 8.16. Inisialisasi M_{tgTM} untuk Timed-gTM

Simulasi *timed-gTM* oleh mesin M_{tgTM}:

- Step 0: Semua *clock* dijalankan, yaitu untuk setiap *clock* x_i mulai berdetak.
- Step 1: M_{tgTM} membaca *current state* q pada sel pertama pita 3. Jika $q \geq f$ maka HALT. Instruksi mencapai *state accept*. Jika $q<f$, lanjut ke step 2.
- Step 2: M_{tgTM} membaca sel-sel pada pita 1 (membaca 1 *word*, *head* berhenti membaca jika mencapai sel spasi atau ∅).
- Step 3: M_{tgTM} mencari instruksi pada pita 2 yang memiliki *state* awal adalah q sesuai pada pita 3 dan memiliki *word* input adalah sesuai yang terbaca pada step 2. Jika tidak ada instruksi yang demikian maka FAIL dan HALT. Jika diketemukan (misal (q, (s, t), q', s', d, *guard*, *reset*)) maka lanjut ke step 4.
- Step 4: M_{tgTM} lalu membaca *time* t pada *word* di instruksi lalu memeriksa apakah untuk setiap nilai *clock* x_i *yang* berjalan dipenuhi hitungan waktu berikut:

$$v(x_i) - v_0(x_i) + t_0 \leq t \qquad ...(1)$$

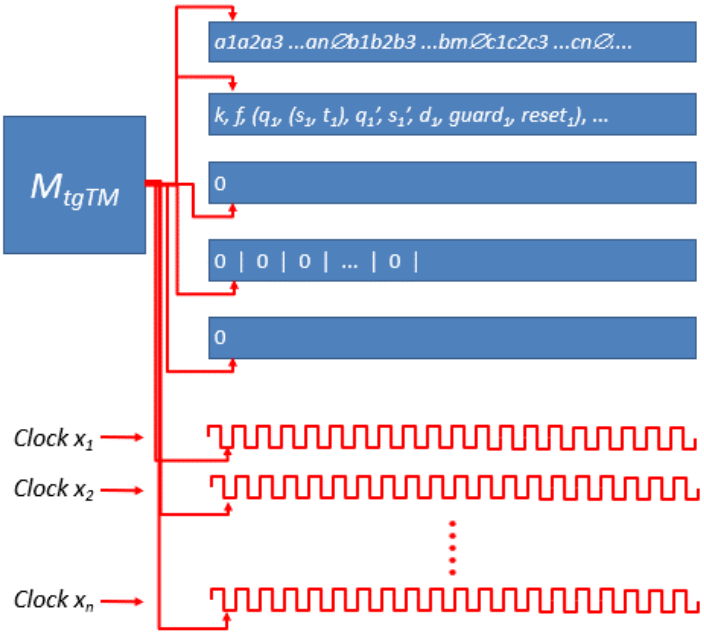
Arti dari persamaan (1) ialah bahwa $v(x_i)$ adalah nilai *clock* x_i saat ini (*current time* x_i) yang diambil dari *clock* x_i yang sedang berdetak dan $v_0(x_i)$ adalah nilai *clock* x_i yang tercatat pada pita 4 untuk masing-masing x_i. Jika salah satu dari *clock* x_i tidak memenuhi persamaan (1) maka FAIL dan HALT, tetapi Jika sebaliknya, seluruh *clock* x_i memenuhi maka lanjut ke step 5.

- Step 5: M_{tgTM} menulis *state* berikut yaitu *q'* pada pita 3 di sel yang menimpa *current state* sebelumnya.
- Step 6: M_{tgTM} menulis *s'* menimpa *s* sebelumnya, mulai sel pertama dan *s* mulai pada pita 1.
- Step 7: Setelah selesai menulis *s'*, M_{tgTM} lalu menggeser *head* pada pita 1 sejauh *d* sel.
- Step 8: M_{tgTM} lalu membaca *guard* waktu pada instruksi, jika tidak kosong maka M_{tgTM} memeriksa setiap *clock*, jika setiap waktu *clock* x_i yang sedang berjalan tidak memenuhi *guard* (minimal salah satu *clock* tidak memenuhi) maka FAIL dan HALT, jika *guard* kosong atau tidak kosong tetapi semua *clock* x_i memenuhi *guard* maka lanjut ke step 9.
- Step 9: M_{tgTM} lalu membaca *reset* instruksi, jika kosong maka lansung ke step 10, jika tidak kosong maka M_{tgTM} mereset *clock* x_i yang bersesuaian atau dinyatakan untuk direset lalu menjalankan ulang *clock* tersebut mulai dari titik reset.
- Step 10: M_{tgTM} selesai mengeksekusi instruksi. M_{tgTM} lalu membaca nilai *clock* x_i terakhir yang sedang berjalan lalu nilainya disimpan berturutan pada pita 4 sebagai $v_0(x_i)$ menimpa valuasi nilai *clock* sebelumnya. Nilai waktu *t* pada *word (s,t)* yang terbaca sebelumnya ditulis pada pita 5 menimpa waktu t_0 sebelumnya. *Head* pita 2, 3, 4 dan pita 5. dikembalikan ke posisi paling awal.
- Step 11: Kembali ke step 1.
- Selesai.

3. Demonstrasi cara kerja M_{tgTM} untuk *timed-gTM*:

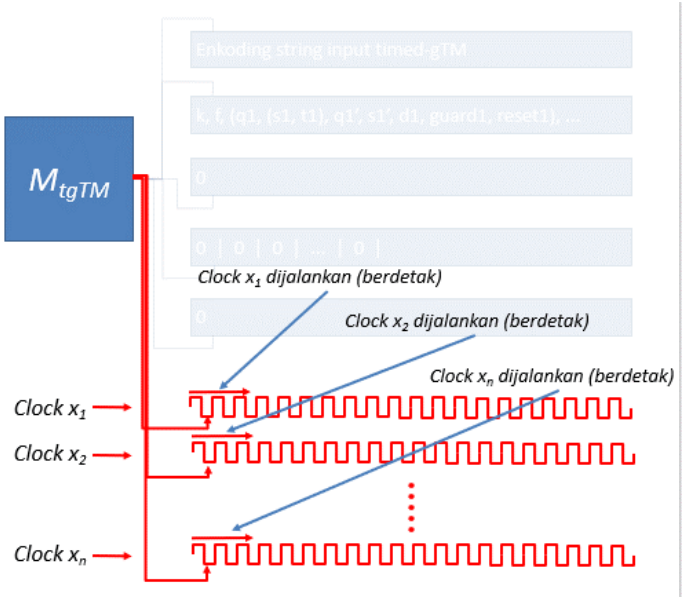
Langkah berikut ini akan dikemukakan demonstrasi kerja dari M_{tgTM} ketika mensimulasikan *timed-gTM*. Untuk keperluan demonstrasi ini, diambil sebuah ekspresi input *timed-multiword* kepada *timed-gTM* secara umum, yaitu $a_1a_2a_3 \ldots a_r \varnothing b_1b_2b_3 \ldots b_m \varnothing c_1c_2c_3 \ldots c_h \varnothing \ldots$ dan bentuk umum instruksi yang telah dienkoding, yaitu $k, f, (q_1, (s_1, t_1), q_1', s_1', d_1, guard_1, reset_1)$, ... serta di sana ada n buah *clock*. Demonstrasi itu adalah sebagai berikut:

a. Inisialisasi mesin: Step 1,2,3,4,5,6,7,8. Gambar 8.17. mengilustrasikan inisialisasi mesin.



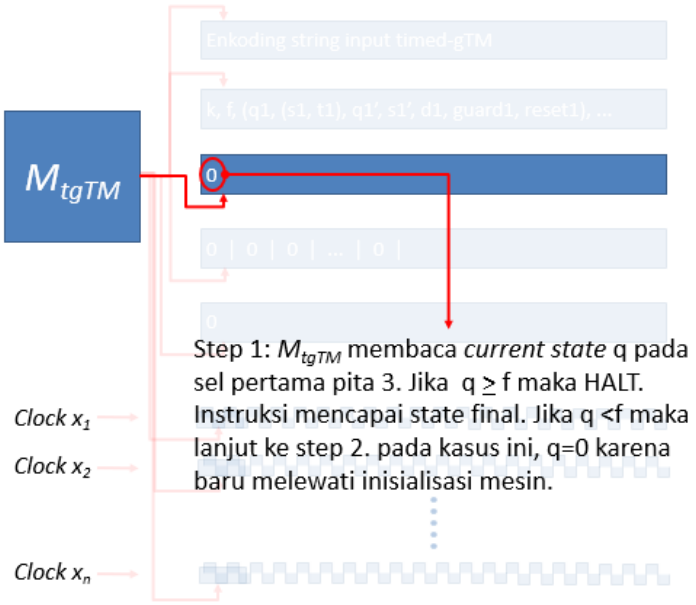
Gambar 8.17. Tahap Inisialisasi Mesin

b. Simulasi mesin: Step 0. Gambar 8.18. mengilustrasikan step 0.



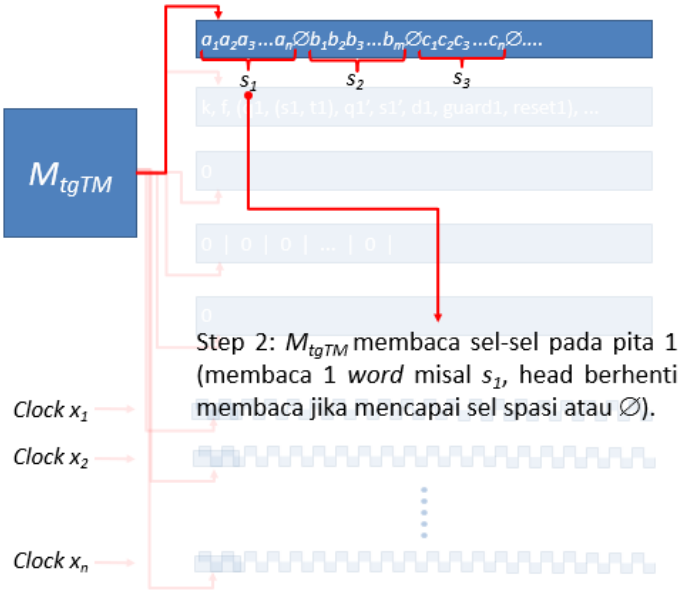
Gambar 8.18. Simulasi Step 0

c. Simulasi mesin: Step 1, Gambar 8.19. mengilustrasikan step 1.



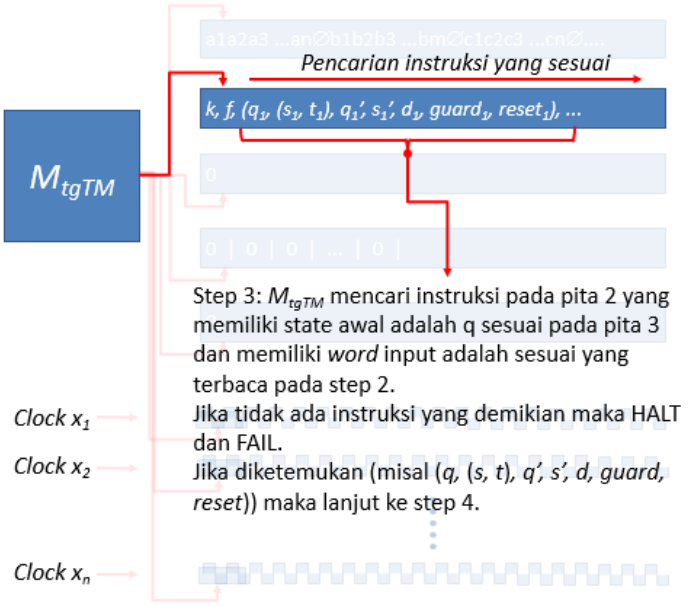
Gambar 8.19. Simulasi Step 1

d. Simulasi mesin: Step 2, Gambar 8.20. mengilustrasikan step 2.



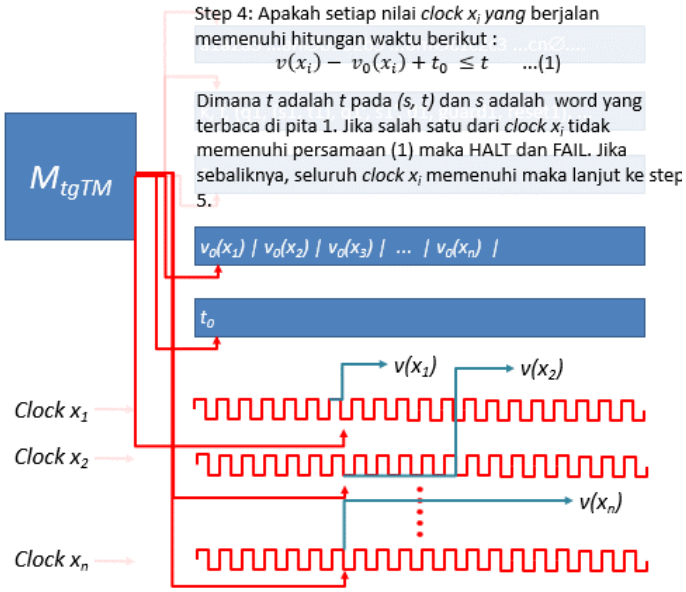
Gambar 8.20. Simulasi Step 2

e. Simulasi mesin: Step 3, Gambar 8.21. mengilustrasikan step 3.



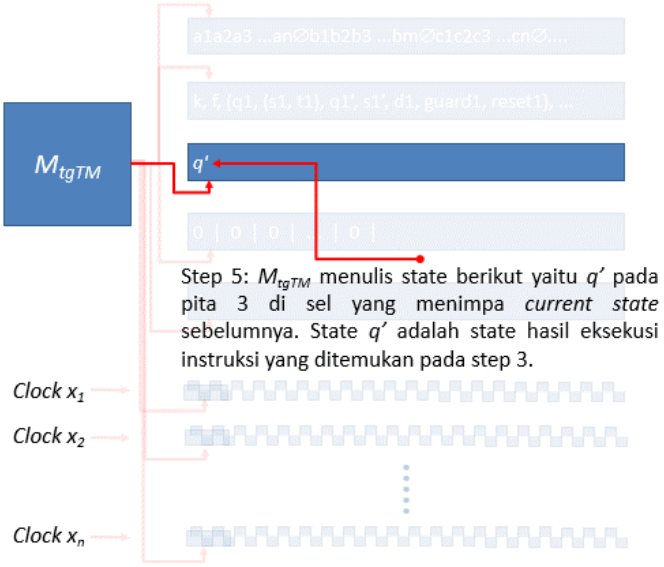
Gambar 8.21. Simulasi Step 3

f. Simulasi mesin: Step 4, Gambar 8.22. mengilustrasikan step 4.



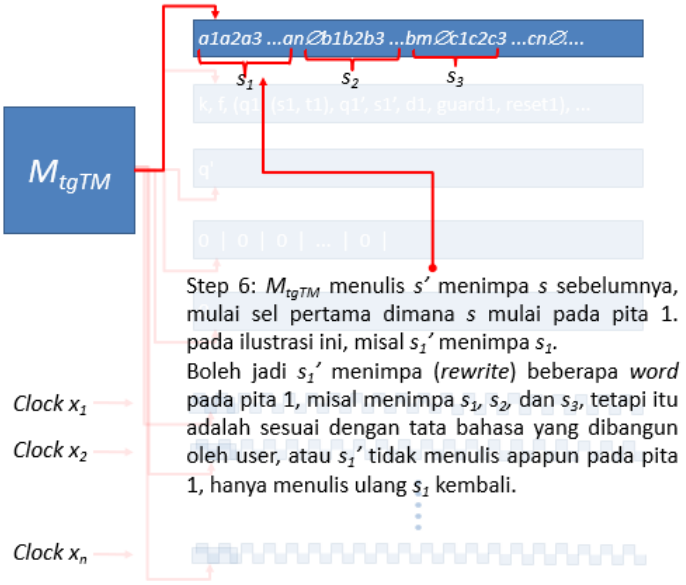
Gambar 8.22. Simulasi Step 4

g. Simulasi mesin: Step 5, Gambar 8.23. mengilustrasikan step 5.



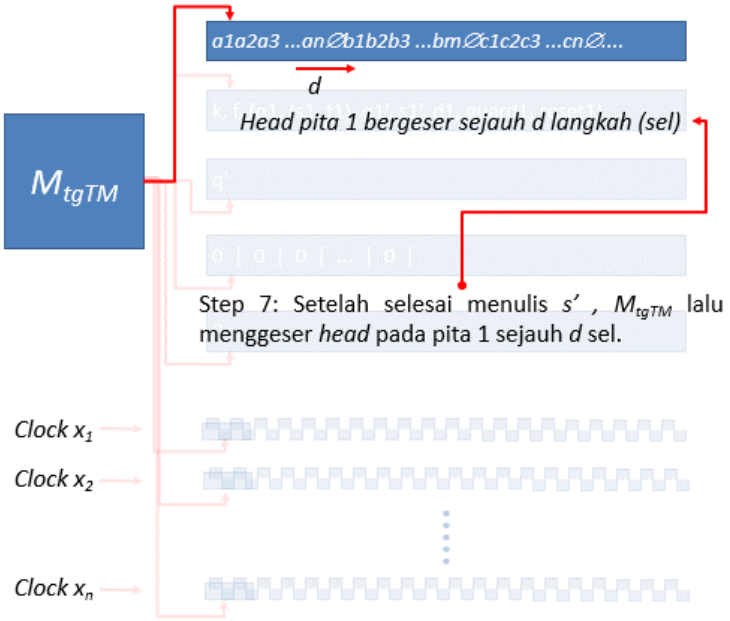
Gambar 8.23. Simulasi Step 5

h. Simulasi mesin: Step 6, Gambar 8.24. mengilustrasikan step 6.



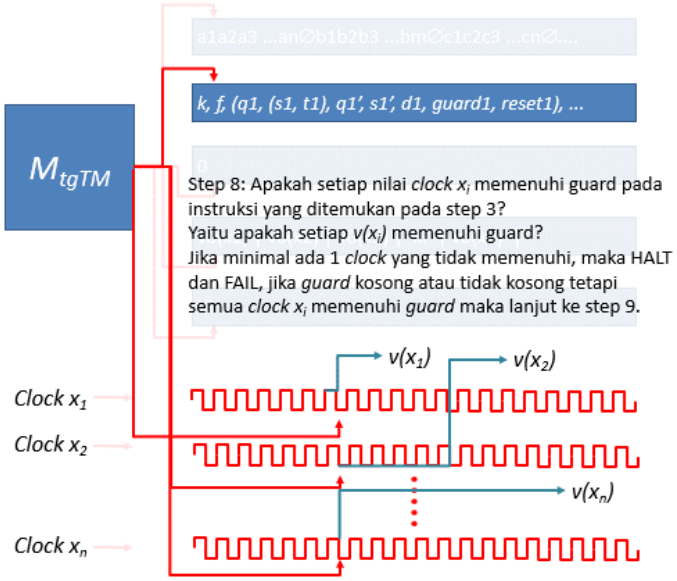
Gambar 8.24. Simulasi Step 6

i. Simulasi mesin: Step 7, Gambar 8.25. mengilustrasikan step 7.



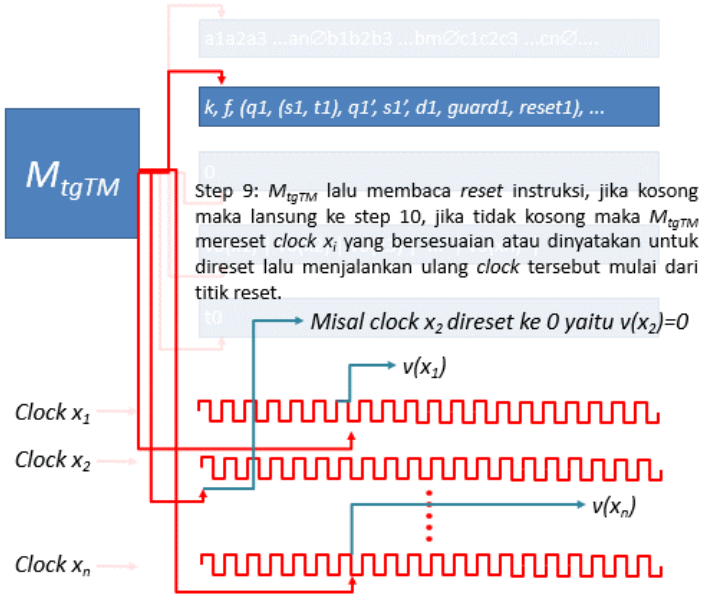
Gambar 8.25. Simulasi Step 7

j. Simulasi mesin: Step 8, Gambar 8.26. mengilustrasikan step 8.



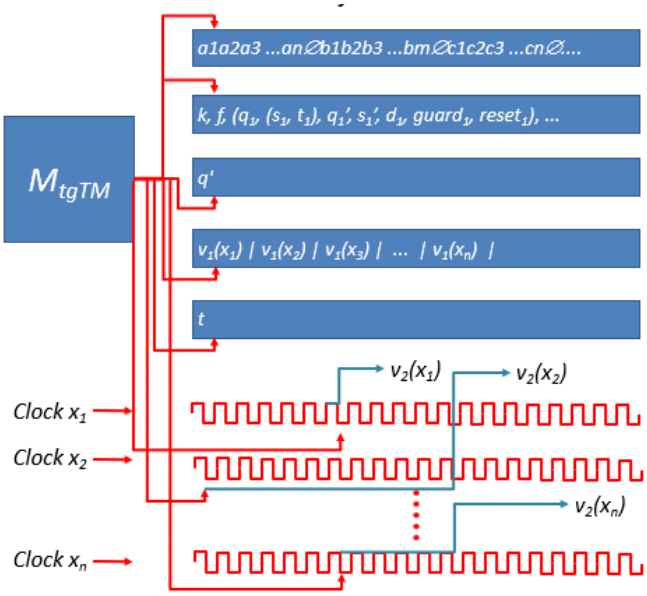
Gambar 8.26. Simulasi Step 8

k. Simulasi mesin: Step 9, Gambar 8.27. mengilustrasikan step 9.



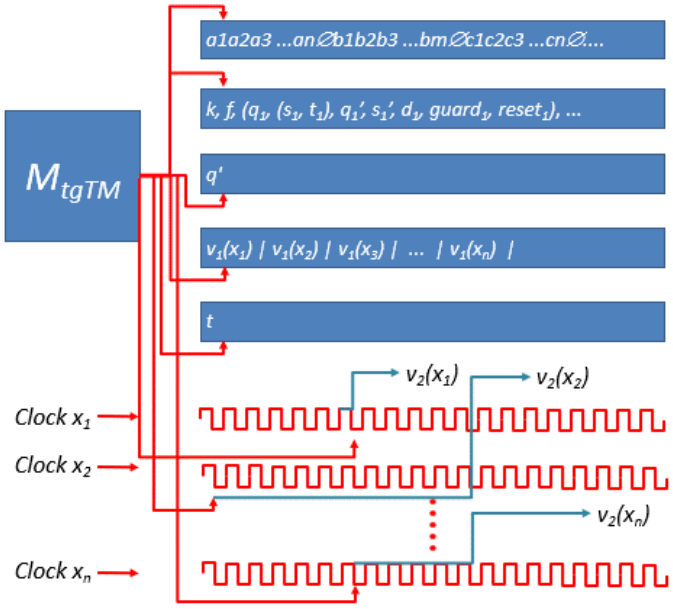
Gambar 8.27. Simulasi Step 9

l. Simulasi mesin: Step 10, Gambar 8.28. mengilustrasikan step 10.



Gambar 8.28. Simulasi Step 10

m. Simulasi mesin: Step 11, Gambar 8.29. mengilustrasikan step 11.



Gambar 8.29. Simulasi Step 11

n. Selesai.

Dengan demikian, langkah berikut akan dikemukakan sebuah bukti matematika bahwa mesin Turing M_{tgTM} adalah benar-benar dapat mensimulasikan *timed-gTM* tanpa harus dikemukakan bukti kasus dengan menulis algoritma M_{tgTM} sebagai program komputer yang bisa dieksekusi. Bukti ini diharapkan sebagai bukti universal (bukan per kasus) untuk teori yang dikemukakan pada pasal ini.

Pembuktian ini menggunakan definisi 6.3, yaitu dengan menunjukkan bahwa M_{tgTM} memenuhi definisi tersebut untuk mensimulasikan *timed-gTM*.

Teorema 8.5 *M_{tgTM} adalah sebuah mesin Turing yang dapat mensimulasikan timed-gTM.*

Bukti:

Untuk menunjukkan bahwa M_{tgTM} dapat mensimulasikan *timed-gTM*, menurut definisi 6.3, yaitu 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 $\alpha(r_{MW})=r_M$ and $s=\alpha(w)$ maka di sana harus dibangun beberapa fungsi enkoding.

- Buat fungsi enkoding α_w yang memetakan setiap input *word* dari *timed-gTM* ke input *word* pada pita 1 dan α_w *bijective* (α_w bersifat *injective* dan *surjective,* yaitu satu ke satu dan setiap simbol input *word* dari *timed-gTM* terpetakan ke simbol pita 1 dari M_{tgTM} tanpa sisa). *Word* yang merupakan input dari *timed-gTM* dienkoding ke dalam $A \cup \{\varnothing\}$ secara *bijective* sehingga diperoleh sebuah bahasa yang dienkoding yaitu $\alpha_w(L_{\Sigma i}) = \{b \mid b= b_1b_2b_3 \ldots b_n,\ b_j = \alpha_w(a_j),\ a_j \in \Sigma i ,\ b_j \in A \cup \{\varnothing\}\}$, selanjutnya untuk $\alpha_w(\text{timed-}L_{\Sigma i})= \{ (b,t) \mid (b,t)= (b_1b_2b_3 \ldots b_n,t) = (b_1,t_1)(b_2,t_2)(b_3,t_3) \ldots (b_n,t_n),\ (b_j, t_j) = (\alpha_w(a_j), t_j) ,\ (a_j,t_j) \in \text{timed-}L_{\Sigma i} ,\ b_j \in A \cup \{\varnothing\}$.
- Buat fungsi enkoding α_s yang *bijective* memetakan setiap *state* dari *timed-gTM* ke barisan bilangan bulat *0, 1, ... k* dan $q_0 \equiv 0$ dan *state accept* adalah *f, f + 1, ... k* untuk $0 < f \leq k$ sehingga himpunan *state* setelah enkoding adalah $\alpha_s(Q) = \{0,1,2, 3, \ldots k\},\ q_0 = 0$ *dan* $\alpha_s(F) = \{f, f + 1, \ldots k\},\ k \geq 0,\ f \leq k$.
- Buat fungsi enkoding α_c yang *bijective*, memetakan setiap nilai *clock* di *timed-gTM* menjadi nilai *clock* di M_{tgTM}. Akan tetapi, dalam hal *clock* untuk mempermudah pembuktian, Setiap *clock* pada M_{tgTM} dianggap sama saja ekspresi dan nilainya dengan *clock* pada *timed-gTM* sehingga α_c dapat didefinisikan sebagai $\forall x \in C,\ \alpha_c(x)=x$ *clock* di M_{tgTM} atau $\alpha_c(C)=C$, misal *clock x* dan *y* pada *timed-gTM* dienkoding menjadi $\alpha_c(x) = x$ dan $\alpha_c(y) = y$ sebagai *clock* pada

M_{tgTM}. Secara umum, semua simbol yang digunakan untuk membuat ekspresi *guard* dan *reset* pada *timed-gTM* dienkoding ke dirinya sendiri, yaitu $\alpha_c(guard)=guard$ dan $\alpha_c(reset)=reset$.

- Buat fungsi enkoding α_d yang memetakan setiap perpindahan *head* sejauh *d* pada pita *timed-gTM* dan α_d *bijective*.
- Langkah berikut buat fungsi enkoding α yang memetakan setiap instruksi pada *timed-gTM* menjadi instruksi di M_{tgTM}. α didefinisikan sebagai:

  $\alpha(q,(w,t),q',s',d,guard,reset)=(\alpha_s(q),\alpha_w(w,t),\alpha_s(q'),\alpha_w(w'),\alpha_d(d),\alpha_c(guard),\alpha_c(reset))$

  Akan tetapi, untuk

  $\alpha(q_1,(w_1,t_1),q_1',w_1',d_1,guard_1,reset_1)=\alpha(q_2,(w_2,t_2),q_2',w_2',d_2,guard_2,reset_2)$

maka $(\alpha_s(q_1),\alpha_w(w_1,t_1),\alpha_s(q_1'),\alpha_w(w_1'),\alpha_d(d_1),\alpha_c(guard_1),\alpha_c(reset_1)) = (\alpha_s(q_2), \alpha_w(w_2,t_2), \alpha_s(q_2'), \alpha_w(w_2'), \alpha_d(d_2), \alpha_c(guard_2), \alpha_c(reset_2))$

sehingga $\alpha_s(q_1)=\alpha_s(q_2)$, $\alpha_w(w_1,t_1)=\alpha_w(w_2,t_2)$, $\alpha_s(q_1')=\alpha_s(q_2')$, $\alpha_w(w_1')=\alpha_w(w_2')$, $\alpha_d(d_1)=\alpha_d(d_2)$, $\alpha_c(guard_1)=\alpha_c(guard_2)$, $\alpha_c(reset_1)=\alpha_c(reset_2)$. *Akan tetapi*, α_s, α_w, α_c, α_d, kesemuanya adalah fungsi yang *bijective* sehingga mereka juga *injective* sehingga berakibat $q_1=q_2$, $(w_1,t_1)= (w_2,t_2)$, $q_1'=q_2'$, $w_1'=w_2'$, $d_1=d_2$, $guard_1=guard_2$, $reset_1= reset_2$. Ini berarti $(q_1,(w_1,t_1),q_1',w_1',d_1,guard_1,reset_1) = (q_2, (w_2,t_2), q_2', w_2', d_2, guard_2, reset_2)$. Karena itu, α adalah fungsi yang *injective*.

Akan tetapi, dalam konstruksi M_{tgTM}, pita 2 hanya diperuntukkan oleh semua hasil enkoding dari instruksi *timed-gTM* sehingga secara otomatis, untuk semua instruksi di pita 2 M_{tgTM}, selalu merupakan hasil enkoding dari instruksi di *timed-gTM*. Secara formal dinyatakan bahwa $\forall p'$ instruksi yang tertulis di pita 2 M_{tgTM} maka $\exists p$ instruksi di *timed-gTM*, $\ni p'= \alpha(p)$. Ini berarti α juga bersifat *surjective*.

Karena α *injective* dan *surjective* maka α adalah fungsi enkoding yang *bijective*, selanjutnya dengan menggunakan α yang *bijective* ditunjukkan bahwa mesin Turing M_{tgTM} dapat mensimulasikan seluruh *timed-gTM*, yaitu bahwa untuk suatu input (w,t) yang diberikan kepada *timed-gTM* maka di sana ada barisan instruksi yang dieksekusi berurutan oleh *timed-gTM*, yaitu $r_{timed\text{-}gTM} = p_0, p_1, p_2, p_3, ..., p_k$, ditunjukkan bahwa M_{tgTM} juga mengeksekusi barisan instruksi pada pita 2, yaitu $r_{MtgTM} = p_0', p_1', p_2', p_3', ..., p_g'$ sedemikian sehingga $\alpha(r_{timed\text{-}gTM}) = r_{MtgTM}$ dan $\alpha_w(w,t)=(s,t)$.

Untuk $k = 0$:

Untuk iterasi $k = 0$ dari algoritma M_{tgTM} akan ditunjukkan bahwa $\alpha(r_{timed\text{-}gTM}) = \alpha(p_0) = p_0' = r_{MtgTM}$ dan $\alpha_w(w,t)=(s,t)$.

Tulis $(w,t)=(w_0,t_0)(w_1,t_1)(w_2,t_2)...$

Untuk $(w,t)=(w_0,t_0)$ maka di sana ada instruksi *timed-gTM* yang dieksekusi untuknya yang *timed-gTM* FAIL atau HALT. Sebut instruksi yang pertama dieksekusi itu sebagai $p_0=(q,(w,t_0),q_0',w_0',d_0,guard_0,reset_0)$.

Ketika M_{tgTM} mensimulasikan *timed-gTM*, α_w mengenkoding $(w,t)=(w_0,t_0)$ ke pita 1 M_{tgTM} sebagai $\alpha_w(w,t) = \alpha_w(w_0,t_0) = (s_0,t_0) = (s,t)$ kemudian M_{tgTM} mengeksekusi perintah p_0' yang diambil pada pita 2. Misalkan $p_0' \neq \alpha(p_0)$, tetapi α *bijective* sehingga dia *surjective*. Karena α *surjective* maka pastilah di sana ada p_i instruksi *timed-gTM* sedemikian sehingga:

$p_0'=\alpha(p_i)= (\alpha_s(q_i),\alpha_w(w_i,t_i),\alpha_s(q_i'),\alpha_w(w_i'),\alpha_d(d_i),\alpha_c(guard_i),\alpha_c(reset_i))$

$= (\alpha_s(q_i),(s_0,t_0),\alpha_s(q_i'),\alpha_w(w_i'),\alpha_d(d_i),\alpha_c(guard_i),\alpha_c(reset_i))$

Akan tetapi, α_w, α_s, α_d, dan α_c juga *bijective* sehingga diperoleh:

$p_i = (q_i,(w_i,t_i),q_i',w_i',d_i,guard_i,reset_i)$ (8.1)

Akan tetapi, pada tahapan inisialisasi M_{tgTM}, pada pita 1, *head* bersiap membaca (s_0,t_0). Sedang (s_0,t_0) adalah hasil enkoding $\alpha_w(w_0,t_0)$, yaitu $(s_0,t_0)=\alpha_w(w_0,t_0)$. Sedang pita 3 berisi enkoding dari *state* start, yaitu $0=\alpha_s(q_0)$, yaitu:

$(s_0,t_0)=\alpha_w(w_0,t_0)$ dan $0=\alpha_s(q_0)$ (8.2)

Ini berarti, ketika simulasi dimulai, yaitu saat k = 0, M_{tgTM} menggunakan instruksi p_0'. Akan tetapi, sudah ditunjukkan pada penurunan persamaan (8.1), yaitu bila $p_0'= \alpha(p_i)$. Ini berarti $\alpha(p_i)$ adalah yang pertama dieksekusi oleh M_{tgTM}. Karena $\alpha(p_i)$ yang pertama dieksekusi, tetapi $\alpha(p_i) = (\alpha_s(q_i), \alpha_w(w_i,t_i), \alpha_s(q_i'), \alpha_w(w_i'), \alpha_d(d_i), \alpha_c(guard_i), \alpha_c(reset_i))$, berdasarkan algoritma M_{tgTM} (lihat step 3, bagian simulasi) maka haruslah berlaku bahwa:

$\alpha_w(w_i,t_i)=(s_0,t_0)$ dan $\alpha_s(q_i)=0$ (8.3)

Karena (8.2) dan (8.3) maka:

$\alpha_s(q_0)=0=\alpha_s(q_i)$ dan $\alpha_w(w_0,t_0)= (s_0,t_0)= \alpha_w(w_i,t_i)$ (8.4)

Karena α_s dan α_w bersifat *bijective* dan karena (8.4) maka:

$q_0=q_i$ dan $(w_0,t_0)=(w_i,t_i)$ (8.5)

Karena (8.1) dan (8.5) maka:

$p_i = (q_0,(w_0,t_0),q_i',w_i',d_i,guard_i,reset_i)$

Akan tetapi, *timed-gTM* bukanlah mesin atau automata nondeterministik sehingga pastilah di sana hanya ada 1 transisi yang berpindah dengan *state* awal adalah q_0 dan *word* input adalah (w_0,t_0). Ini berarti hanya ada 1 instruksi yang memiliki pola $(q_0,(w_0,t_0),$?, ?, ?, ?, ? $)$.

Akan tetapi, p_i, p_0 memenuhi pola instruksi $(q_0,(w_0,t_0),$?, ?, ?, ?, ? $)$ karena *timed-gTM* adalah bukan nondeterministik maka diperoleh:

$$p_i=p_0 \qquad (8.6)$$

Karena (8.6) dan sifat *bijective* α maka:

$$\alpha(r_{timed\text{-}gTM})=\alpha(p_0)=\alpha(p_i)=p_0'=r_{MtgTM} \qquad (8.7)$$

Berdasarkan definisi 6.3 dan $\alpha_w(w_0,t_0)=(s_0,t_0)$ maka M_{tgTM} telah dibuktikan dapat mensimulasikan *timed-gTM* untuk $k = 0$.

Untuk $k = n$:

Misalkan untuk langkah ke-n eksekusi instruksi, yaitu bahwa *timed-gTM* mengeksekusi instruksi sampai pada $r_{timed\text{-}gTM} = p_0, p_1, p_2, p_3, ..., p_n$ dan tetap berlaku bahwa:

$$\alpha(r_{timed\text{-}gTM})=\alpha(p_0),\alpha(p_1),\alpha(p_2),\alpha(p_3),...,\alpha(p_n)=p_0',p_1',p_2',p_3',..., p_n'=r_{MtgTM} \qquad (8.8)$$

$$\alpha_w(w,t)= \alpha_w((w_0,t_0)(w_1,t_1)(w_2,t_2)... (w_n,t_n))= (s_0,t_0)(s_1,t_1)(s_2,t_2)...(s_n,t_n) \qquad (8.9)$$

Untuk $k = n+1$:

Timed-gTM sendiri untuk langkah ke-n, menghasilkan *state* q_{n+1} sebagai *state* berikut dan bersiap membaca *word* input yang ke-n+1, yaitu (w_{n+1}, t_{n+1}). Akan tetapi, berdasarkan (8.8) maka $\alpha(p_n)=p_n'$, yaitu bahwa:

$$\alpha(p_n)= (\alpha_s(q_n),\alpha_w(w_n,t_n),\alpha_s(q_n'),\alpha_w(w_n'),\alpha_d(d_n),\alpha_c(guard_n),\alpha_c(reset_n))= (m_n, (s_n,t_n), m_n', s_n', d_n, guard_n,reset_n)= p_n'. \qquad (8.10)$$

dan $p_n = (q_n,(w_n,t_n),q_n',w_n',d_n, guard_n, reset_n)$

Dengan menulis q_n' sebagai q_{n+1} ($q_n'=q_{n+1}$), Ini berarti instruksi berikut pada *timed-gTM* adalah instruksi p_{n+1} yaitu

$$p_{n+1} = (q_{n+1}, (w_{n+1},t_{n+1}), q_{n+1}', w_{n+1}', d_{n+1}, guard_{n+1}, reset_{n+1}) \qquad (8.11)$$

Dengan menulis m_n' sebagai m_{n+1} ($m_n' = m_{n+1}$) maka instruksi berikut yang dieksekusi oleh M_{tgTM} adalah instruksi dengan pola:

$$p_{n+1}' = (m_{n+1}, (s_{n+1},t_{n+1}), m_{n+1}', s_{n+1}', ?, ?, ?) \qquad (8.12)$$

Selanjutnya, akan ditunjukkan bahwa $\alpha(p_{n+1}) = p_{n+1}'$.

Berdasarkan (8.10) maka konfigurasi mesin M_{tgTM} sesaat sebelum memulai langkah ke-n+1 adalah:

(1) Pita 1, telah selesai membaca *word* input yang ke-n, siap membaca *word* input yang ke-n+1, yaitu (s_{n+1}, t_{n+1}), tetapi α_w bersifat *bijective* maka mestilah bahwa $\alpha_w^{-1}(s_{n+1}, t_{n+1}) = (w_{n+1}, t_{n+1})$, yaitu bahwa $(s_{n+1}, t_{n+1}) = \alpha_w(w_{n+1}, t_{n+1})$.

(2) *Head* Pita 2, direset ke sel pertama.

(3) Sel pertama pita 3, berisi *state* pada langkah ke-n+1 berikutnya, yaitu m_n' sebagai m_{n+1} ($m_n' = m_{n+1}$). Akan tetapi, dari (8.10) dan bahwa q_n' sebagai q_{n+1} (yaitu $q_n'=q_{n+1}$) maka berlaku $\alpha_s(q_{n+1})=m_{n+1}$

(4) Pita 4 berisi nilai-nilai *clock* sebelumnya (di sana ada *g* buah *clock*, sebarang), nilai saat di langkah ke-n, yaitu: $v_n(x_1), v_n(x_2), v_n(x_3),..., v_n(x_g)$.

(5) Pita 5 berisi waktu dari *word* input ke-n yang baru saja selesai dibaca dan menjadi bagian dari eksekusi instruksi ke-n, yaitu t_n.

(6) Semua *g* buah *clock* berjalan atau berdetak lagi melampaui nilai detak ke-n sebelumnya, yaitu melampaui $v_n(x_1), v_n(x_2), v_n(x_3),..., v_n(x_g)$ berjalan sebagai $v_{n+1}(x_1), v_{n+1}(x_2), v_{n+1}(x_3),..., v_{n+1}(x_g)$.

Dengan konfigurasi mesin di atas, selanjutnya mesin M_{tgTM} mencari instruksi yang ke-n+1, yaitu p_{n+1}'. Pencarian (lihat step 3, bagian simulasi oleh M_{tgTM}) dilakukan dengan mencari instruksi yang cocok dengan pola $(m_{n+1}, (s_{n+1},t_{n+1}), ?, ?, ?, ?, ?)$ seperti (8.12), jika tidak ketemu maka FAIL dan HALT. Jika tidak ketemu karena α bersifat *bijective* maka pastilah di sana tidak ada juga invers dari α, yaitu tidak ada instruksi *timed-gTM* yang bisa mengeksekusi pada langkah ke-n+1 sehingga *timed-gTM* juga FAIL dan HALT. Ini berarti simulasi selesai. Pengandaian untuk k=n dapat diundur menjadi pengandaian untuk k=n-1 dan yang dibuktikan selanjutnya adalah untuk k=n demikian seterusnya jika masih FAIL dan HALT, hingga mencapai langkah k=0, tetapi untuk langkah k=0 telah dibuktikan sebelumnya sehingga pembuktian selesai. Andaikan tidak FAIL atau HALT maka pembuktian itu diwakilkan pada pembuktian untuk k=n+1 berikut ini.

Langkah berikutnya andaikan instruksi dengan pola $(m_{n+1}, (s_{n+1},t_{n+1}), ?, ?, ?, ?, ?)$ ditemukan, yaitu p_{n+1}' ditemukan dan siap untuk dieksekusi, tetapi berdasarkan konfigurasi mesin M_{tgTM} sesaat sebelum mengeksekusi p_{n+1}' yaitu point (1) dan (3) di atas, diperoleh:

$$(s_{n+1}, t_{n+1}) = \alpha_w(w_{n+1}, t_{n+1}) \text{ dan } \alpha_s(q_{n+1})=m_{n+1} \qquad (8.13)$$

Berdasarkan algoritma simulasi M_{tgTM} untuk *timed-gTM* (lihat pada step 3 dari algoritma bagian simulasi) dan konfigurasi mesin sesaat setelah mengeksekusi p_n', yaitu konfigurasi (1) sampai (6) di atas, mesin M_{tgTM} mencari instruksi yang ke-n+1 yang memenuhi pola $(m_{n+1}, (s_{n+1},t_{n+1}), ?, ?, ?, ?, ?)$, tulis itu sebagai $p_{n+1}' = (m_{n+1}, (s_{n+1},t_{n+1}), ?, ?, ?, ?, ?)$. berdasarkan (8.13) maka $p_{n+1}' = (m_{n+1}, (s_{n+1},t_{n+1}), ?, ?, ?, ?, ?) = (\alpha_s(q_{n+1}), \alpha_w(w_{n+1}, t_{n+1}), ?, ?, ?, ?, ?)$, tetapi $\alpha(p_{n+1})$ juga memenuhi $(\alpha_s(q_{n+1}), \alpha_w(w_{n+1}, t_{n+1}),?,?,?,?,?)$, yaitu dari bentuk ekspresinya $\alpha(p_{n+1}) = (\alpha_s(q_{n+1}), \alpha_w(w_{n+1},t_{n+1}), \alpha_s(q_{n+1}'), \alpha_w(w_{n+1}'), \alpha_d(d_{n+1}), \alpha_c(guard_{n+1}), \alpha_c(reset_{n+1}))$.

Akan tetapi, *timed-gTM* adalah mesin yang bukan mesin atau automata nondeterministik dan fungsi enkoding α juga *bijective* maka pastilah hanya ada 1 instruksi yang memenuhi pola $(m_{n+1}, (s_{n+1},t_{n+1}), ?, ?, ?, ?, ?)$, yaitu hanya ada 1 transisi automata yang word inputnya (s_{n+1},t_{n+1}) dan *state* awalnya adalah m_{n+1}. Ini berarti:

$$p_{n+1}'=(m_{n+1},(s_{n+1},t_{n+1}),?,?,?,?,?)=(\alpha_s(q_{n+1}),\alpha_w(w_{n+1},t_{n+1}),?,?,?,?,?)=\alpha(p_{n+1}) \quad (8.14)$$

Dengan demikian, berdasarkan (8.14) dan (8.13) maka diperoleh:

$$\alpha(r_{timed\text{-}gTM})=\alpha(p_0),\alpha(p_1),\ldots,\alpha(p_n),\alpha(p_{n+1})=p_0',p_1',,\ldots,p_n',p_{n+1}'=r_{MtgTM} \quad (8.15)$$

$$\alpha_w(w,t)=\alpha_w((w_0,t_0)(w_1,t_1)\ldots(w_n,t_n)(w_{n+1},t_{n+1}))=(s_0,t_0)(s_1,t_1)\ldots(s_n,t_n)(s_{n+1},t_{n+1}) \quad (8.16)$$

Dengan demikian, untuk langkah ke-n+1, M_{tgTM} terbukti dapat mensimulasikan *timed-gTM*.

Secara keseluruhan telah ditunjukkan bahwa teorema 8.5 dapat dibuktikan.

Dengan demikian, langkah berikut dengan mudah dapat ditunjukkan bahwa M_{tgTM} adalah sebuah mesin Turing universal, sebagai berikut:

Teorema 8.6 *Mesin Turing M_{tgTM} yang dapat mensimulasikan timed-gTM adalah sebuah mesin Turing universal (UTM).*

Bukti:

Misalkan sebuah MS adalah mesin Turing standar. Enkoding MS dengan suatu fungsi α yang memetakan MS menjadi MS yang *real time* yaitu *timed-MS*, yaitu setiap *word* input MS, misal *w* dipetakan menjadi (w,t) dan *t* sebuah waktu sebarang tetapi memenuhi postulat waktu kemonotonan. α tidak mengubah *w*, tetapi hanya menambahkan *t* dan menjadikannya 2-tupel (w,t). Buat α sedemikian sebagai fungsi enkoding yang *bijective* demikian juga bahwa enkoding setiap instruksi MS secara *bijective* yaitu (q_i, w_i, d_i) menjadi $(q_i, w_i, d_i, guard_i, reset_i)$ dan $guard_i$ dan $reset_i$ adalah sebarang ekspresi konstrain waktu.

Akan tetapi, *timed-MS* dapat dikonsruksikan dari *timed-gTM*, tersebab definisi *timed-gTM* yang umum dapat dibuat sedemkian sehingga memiliki panjang *word* yang menyusun *multiword* adalah memiliki panjang *word* sepanjang 1 karakter saja.

Karena MS ekivalen dengan *timed-MS* (fungsi enkoding yang mengenkoding MS ke *timed-MS* adalah *bijective*), sedang *timed-MS* adalah sebuah *timed-gTM* dan berdasarkan teorema 8.5 maka M_{tgTM} adalah mesin Turing yang dapat mensimulasikan sebuah mesin Turing standar, MS. Ini berarti M_{tgTM} adalah sebuah mesin Turing universal (UTM). Teorema 8.6. telah dibuktikan.

Selanjutnya, akan ditunjukkan bahwa mesin Turing M_{tgTM} dalam mesnimulasikan *timed-gTM* adalah *decidable* atau *solvable* terhadap permasalahan *halting problem*.

Teorema 8.7 *Misalkan timed-gTM dikonstruksikan dengan jumlah instruksi yang berhingga, panjang input yang selalu berhingga serta simbol pita yang berhingga dan jumlah clock yang berhingga maka mesin Turing M_{tgTM} yang mensimulasikannya adalah solvable untuk halting problem.*

Bukti:

Untuk menunjukan bahwa M_{tgTM} adalah solvable untuk simulasi *timed-gTM* yang dimaksud, harus diperiksa apakah panjang setiap pita M_{tgTM} adalah behingga, yaitu apakah jumlah instruksi yang digunakannya untuk mensimulasikan *timed-gTM* adalah berhingga dan apakah jumlah simbol tiap-tiap pita juga adalah berhingga lalu kemudian ditunjukkan bahwa untuk setiap konfigurasi M_{tgTM} dapat diverifikasi (dicapai) tak peduli dia HALT atau NEVER HALT (*looping*).

Tinjau pita 1:

Dikarenakan telah ditetapkan bahwa panjang input *timed-gTM* adalah berhingga, sedang pita 1 semata berisi enkoding input *timed-gTM* dan fungsi enkoding bersifat bijectiv (satu ke satu) maka panjang input pada pita 1 juga adalah selalu berhingga.

Akan tetapi, juga bahwa, jumlah simbol default pada pita 1 adalah sama dengan kardinal himpunan simbol pada keyboard digabung $\{\leq,\geq\}$ yang nilainya berhingga, misal s dan panjang input adalah n maka banyaknya konfigurasi pita yang mungkin pada pita 1 dengan membatasi panjang pita adalah sama dengan n (atau secara umum linier terhadap n) adalah s^n kombinasi.

Karena panjang pita adalah n maka *head* pita dapat bergerak leluasa pada n posisi yang berbeda pada pita sehingga konfigurasi pita jika dimasukkan posisi *head* menjadi $n.s^n$.

Tinjau pita 2:

Timed-gTM ditetapkan memiliki jumlah instruksi yang berhingga karena enkoding instruksi *timed-gTM* bersifat *bijective* (satu ke satu) maka jumlah enkoding instruksi pada pita 2 juga berhingga sehingga panjang input pada pita 2 juga selalu berhingga. Misalkan panjang input pada pita 2 adalah m dan panjang pita 2 dianggap sama dengan m (atau secara umum linier terhadap m), sedang banyaknya simbol yang bisa digunakan pada pita 2 adalah juga s (seperti pita 1) maka jumlah konfigurasi simbol yang mungkin pada pita adalah s^m.

Dengan mengikutkan posisi *head* pita 2 yang bebas berada pada m posisi maka jumlah konfigurasi simbol dan posisi *head* pada pita 2 adalah $m.s^m$.

Tinjau pita 3:

Panjang pita 3 dapat dilihat selalu sama dengan 1 (atau linier terhadap 1) karena dia hanya selalu berisi *current state* dari instruksi *timed-gTM*.

Akan tetapi, jumlah instruksi *timed-gTM* adalah berhingga sehingga jumlah *state* juga adalah berhingga. Misalkan jumlah *state* adalah *q* maka kombinasi simbol pada pita 3 yang panjangnya 1 (atau linier terhadap 1) adalah $1.q^1$ yaitu *q* saja.

Tinjau pita 4:

Pita 4 berisi *c* buah nilai valuasi waktu atau nilai *clock* untuk *c* buah *clock* yang dimiliki *timed-gTM*. Karena jumlah *clock* pada *timed-gTM* adalah berhingga maka panjang pita 4 dapat dilihat berhingga yaitu sepanjang *c* (atau secara umum linier terhadap *c*)

Akan tetapi, apakah jumlah nilai waktu yang mengisi pita itu adalah juga berhingga? Untuk menghitung ini dapat ditinjau sebagai berikut:

Setiap nilai *clock* $v_i(c)$ memiliki hubungan terhadap nilai waktu *word* yang menjadi input *timed-gTM*, yaitu (w_{i-1},t_{i-1}) *dan* (w_i,t_i) dan

$$v_i(c)=v_{i-1}(c)+t_i-t_{i-1} \qquad (8.17)$$

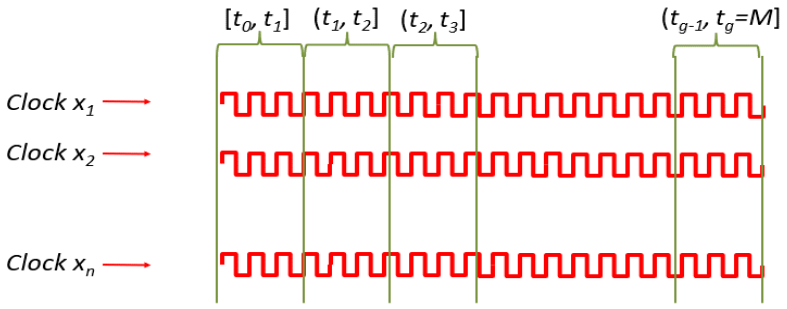
Akan tetapi, panjang input *timed-gTM* adalah terbatas sehingga di sana ada sebuah bilangan *N* dan untuk seluruh nilai i=0,1,2,3... maka $t_i \leq N$, untuk setiap t_i nilai waktu dari *word* input dari *timed-gTM* (yaitu (w_i,t_i)).

Karena persamaan (8.17) maka hubungan antara t_i adalah linier terhadap $v_i(c)$. Ini berarti, $v_i(c)=v_{i-1}(c)+t_i-t_{i-1} \leq v_{i-1}(c)+N-t_{i-1} \leq M$ *dan N,M* bulat.

Ini berarti ada sebuah bilangan bulat *M* sehingga $v_i(c) \leq M$ untuk semua nilai valuasi waktu *clock* dari setiap *clock timed-gTM*.

Jika di sana ada *g* buah *word* input pada pita 1 M_{tgTM} sehingga ada *g* satuan waktu untuk seluruh input (yaitu (w_i,t_i) dan $i=0,1,2,3,...,g-1$), buat *g* buah region waktu pada tiap-tiap *clock* pada M_{tgTM} dengan panjang setiap partisi adalah dalam susunan:

Partisi *clock* = $[t_0, t_1]$ $(t_1, t_2](t_2, t_3]...(t_{g-1}, t_g=M]$



Gambar 8.30. Partisi Seluruh Clock pada M_{tgTM}

Dengan menggunakan partisi pada Gambar 8.30, dapat dilihat bahwa setiap $v_i(c)$ berada dalam salah satu dari g region waktu yang mungkin dengan demikian dapat dilihat kombinasi valuasi waktu pada pita 4 sebagai kombinasi region waktu semata.

Karena di sana ada berhingga yaitu g region waktu maka terdapat g kombinasi region waktu pada pita 4. Karena ada c buah *clock* maka terdapat $c.g^c$ kombinasi waktu pada pita 4.

Tinjau pita 5:

Pita 5 hanya menyimpan nilai waktu *word* input yang telah selesai terbaca tepat sebelum *current time* dari *word* yang sedang terbaca, yaitu misal yang sedang terbaca dan dieksekusi oleh M_{tgTM} adalah (w_i, t_i) maka waktu yang tersimpan pada pita 5 adalah t_{i-1} dari (w_{i-1}, t_{i-1}) sebelumnya.

Ini berarti panjang pita 5 dapat dilihat sebagai 1 saja (atau secara umum linier terhadap 1). Akan tetapi, panjang *word* input *timed-gTM* adalah terbatas maka jumlah unit waktu yang terdapat pada *word* input *timed-gTM* adalah juga terbatas, yaitu misal input *word timed-gTM* memiliki jumlah *word* adalah g maka dapat ditulis:

$(w_0, t_0)\ (w_1, t_1)\ (w_2, t_2)\ldots\ (w_{g-1}, t_{g-1})$

Ini berarti di sana ada g satuan waktu yang mungkin mengisi pita 5. Waktu-waktu itu adalah $t_0, t_1, t_2 \ldots t_{g-1}$.

Dengan demikian, jumlah konfigurasi simbol pada pita adalah:

$1.g^1$ yaitu g saja.

Tinjau keseluruhan M_{tgTM}:

Jumlah instruksi yang dimiliki oleh M_{tgTM} adalah terbatas sebagaimana telah dikonstruksikan pada sebelum ini, tetapi jika tidak melihat pada algoritma M_{tgTM} yang telah dikonstruksikan, hanya melihatnya secara umum. Jika M_{tgTM} mensimulasikan jumlah instruksi yang terbatas (berhingga) dari *timed-gTM* maka di sana ada jumlah instruksi M_{tgTM} yang berhingga untuk mensimulasikan *timed-gTM*.

Misalkan jumlah instruksi M_{tgTM} itu adalah h *maka* secara keseluruhan jumlah konfigurasi yang mungkin untuk M_{tgTM} yang mensimulasikan *timed-gTM* dimaksud dengan k adalah jumlah keseluruhan konfigurasi M_{tgTM} yang mungkin, adalah:

$$k \leq h.(1.g^1).(c.g^c).(1.q^1).(m.s^m).(n.s^n) \qquad (8.18)$$

Ini berarti k terbatas dan karena k bilangan bulat maka k berhingga. Karena k berhingga maka setiap kemungkinan tercapainya sebuah konfigurasi *(reachbility)* dapat diverifikasi dalam

langkah-langkah yang berhingga. Ini berarti setiap konfigurasi dapat diputuskan apakah dia HALT atau dia LOOP FOREVER untuk mencapainya.

Dengan demikian M_{tgTM} untuk *timed-gTM* dimaksud adalah *solvable* untuk *halting problem*.

Teorema 8.7. selesai dibuktikan.

PENUTUP

Dengan selesainya pembuktian di atas, maka konstruksi mesin yang diklaim dalam hak cipta ini dianggap selesai.

DAFTAR PUSTAKA

Alur, R. and Dill, D. L. (1994) 'A theory of *timed automata*', *Theoretical Computer Science*, 126(2), pp. 183–235. doi: 10.1016/0304-3975(94)90010-8.