SEBUAH REVIEW SINGKAT TERHADAP EMULASI CELLULAR AUTOMATA PADA MESIN TURING

Hernawan Sulistyanto[1], Reza Pulungan[2]

[1]Program Studi Teknik Infomatika, Fak. Komunikasi dan Informatika, Universitas Muhammadiyah Surakarta

[2]Program Studi Ilmu Komputer, FMIPA, Universitas Gadjah Mada, Yogyakarta

Sekip Utara, Bulaksumur, Yogyakarta

E-mail: Hernawan.Sulistyanto@ums.ac.id, pulungan @ugm.ac.id

ABSTRAK

Mesin-mesin Turing adalah suatu jenis abstraksi dari komputasi dan merupakan pemodelan yang sangat sederhana dari komputer. Mesin Turing sebagai model komputasi teoritis berfungsi sebagai model ideal untuk melakukan ilustrasi perhitungan/komputasi matematis. Meskipun model ideal ini diperkenalkan sebelum komputer nyata dibangun, model ini tetap diterima kalangan ilmu komputer sebagai model komputer yang sesuai untuk menentukan apakah suatu fungsi dapat diselesaikan oleh komputer atau tidak (menentukan computable function). Universalitas komputasi adalah kemampuan dari sebuah mesin atau program untuk menghitung iterasi dari mesin atau program lain. Oleh karena bukti yang ada dari universalitas komputasi hanya berkaitan dengan Mesin Turing yang asli, maka pembuktian universalitas komputasional bagi program-program yang lain dapat dilaksanakan melalui emulasi. Emulasi berarti bahwa serangkaian iterasi dalam suatu program akan menghasilkan suatu representasi yang setara (equivalent) dengan setiap langkah komputasi dari program yang ditiru. Pada artikel ini akan dipaparkan secara singkat pengemulasian sebuah Celullar Automata terhadap Mesin Turing. Celullar Automata adalah sebuah model komputasi terdesentralisasi yang menyediakan sebuah platform yang mangkus bagi pelaksanaan suatu komputasi yang lebih komplek.

Kata kunci : Mesin Turing, Celullar Automata, Emulasi

1. PENDAHULUAN

Jauh sebelum lahirnya program komputer, Alan Turing pada tahun 1936 telah menyampaikan idenya berupa model mesin abstrak sebagai alat mekanik untuk mengerjakan prosedur yang efektif. Model ini disebut Mesin Turing.

Cara kerja Mesin Turing ekivalen dengan cara kerja komputer digital saat ini. Mesin Turing juga ekivalen dengan problem komputasi matematika.

Sebagai *input* Mesin Turing adalah kata atau untai atas suatu alfabet T. Mesin Turing berhenti dengan keadaan menerima atau menolak untai. Kadang-kadang terjadi pula perulangan atau *looping* tak-hingga. Koleksi Mesin Turing, {M} dikatakan mempunyai *power* yang sama dengan koleksi Mesin Turing, {N} bila untuk setiap mesin M di (M) ada mesin N di {N} yang ekivalen, dan sebaliknya. Selanjutnya dapat dikatakan {M} lebih *powerfull* dari {N}, {M}>{N}, apabila untuk setiap mesin N di {N} ada mesin M di {M} yang ekivalen, tetapi sebaliknya tidak berlaku. Sebagai contoh terdapat tiga mesin yang mempunyai *power* sama, yaitu Mesin Turing – Mesin Post – Mesin Finite dengan dua *pushdown store* (*pushdown automata* dengan dua *puhdown store* atau *stack*). Artikel ini adalah salah satu upaya untuk mengamati performa universalitas dari mesin komputasi lain dan membandingkannya

dengan kemampuan Mesin Turing. Oleh karena bukti yang ada dari universalitas komputasi hanya berkaitan dengan Mesin Turing asli, maka program-program lain dibuktikan universalitasnya secara komputasional melalui suatu emulasi. Emulasi berarti bahwa serangkaian iterasi dalam satu program menghasilkan suatu representasi setara (*equivalent*) dengan setiap langkah komputasi/perhitungan program yang ditiru (Sipser, 2013). Paparan pengemulasian antara *Cellular Automata* dengan Mesin Turing dalam artikel ini didasarkan pada suatu tinjauan teoritis yang disampaikan oleh Wolfram.

Secara keseluruhan artikel ini ditulis dengan pengorganisasian, yaitu pengertian Mesin Turing dan *Cellular Automata* disajikan pada bagian 2 dan 3, selanjutnya pada bagian 4 menyajikan universalitas komputasi yang disusul dengan dengan emulasi dan universalitas *Celllular Automata* pada bagian 5. Bagian 6 menyajikan ambang universalitas dan akhirnya kesimpulan ada pada bagian 7.

2. MESIN TURING

Mesin Turing adalah model yang sangat sederhana dari komputer. Secara esensial, mesin Turing adalah sebuah *finite automaton* yang miliki sebuah *tape* tunggal dengan panjang tak-hingga yang dapat membaca dan menulis data. Mesin Turing menggunakan notasi seperti ID-ID pada PDA untuk menyatakan konfigurasi dari komputasinya. *Stack* pada PDA memiliki keterbatasan akses. Elemen yang dapat diakses hanya elemen yang ada pada *top stack*. Pada Mesin Turing, memori akan berupa suatu *tape* yang pada dasarnya merupakan *array* dari sel-sel penyimpanan.

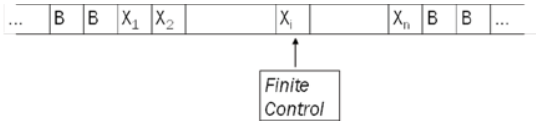
2.1 Spesifikasi Mesin Turing

Stack yang terdapat pada PDA memiliki keterbatasan kemampuan akses oleh karena hanya diperkenankan mengakses data yang terdapat pada *top stack*. Guna memindahkan akses pada bagian yang lebih rendah dari *top stack*, harus memindahkan bagian di atasnya. Permasalahan yang ada sekarang bukanlan keterbatasan memori tetapi bagaimana memori tersebut diorganisasikan.

Mesin Turing lebih bersifat umum dan memiliki kemampuan lebih tinggi dari pada *finite state automata* maupun *push down automata* dari segi tindakan dan komponennya. Pada mesin Turing, memori akan berupa suatu pita yang pada dasarnya berupa *array* atau deretan sel-sel penyimpanan. Pita tersebut tidak mempunyai sel pertama dan sel terakhir. Setiap sel mampu menyimpan sebuah simbol tunggal. Pita dapat memuat informasi dalam jumlah tak terbatas, dan dapat dijelajahi/diakses pada bagian manapun dan urutan manapun dari pita. Terdapat sebuah *head* yang menunjukkan posisi yang di akses pada pita. *Head* dapat bergerak kekanan dan kekiri untuk membaca *input* pada pita dan sekaligus juga bisa melakukan penulisan pada pita atau mengubah isi pita. Mesin Turing bisa dianalogikan seperti komputer sederhana, dengan sejumlah *state* sebagai memori, pita sebagai secondary *storage*, dan fungsi transisi sebagai program.

2.2. Penyajian Mesin Turing

Visualisasi dari sebuah mesin Turing diberikan oleh gambar berikut:



Gambar 1. Visualisasi Mesin Turing

Mesin terdiri dari sebuah *finite control*, yang dapat berada dalam sebuah himpunan berhingga dari *state*. Terdapat sebuah *tape* yang dibagi ke dalam kotak-kotak atau deretan sel-sel. Setiap sel dapat menampung sebuah dari sejumlah berhingga simbol. Pada awalnya, *input* yang merupakan *string* dari simbol dengan panjang berhingga dipilih dari *input alphabet*, ditempatkan pada *tape*. Sel-sel *tape* yang lain, merupakan perluasan secara *infinite* ke kiri dan ke kanan, pada awalnya menampung simbol khusus yang dinamakan *blank*. *Blank* bukan sebuah *input symbol*, dan mungkin terdapat simbol *tape* yang lain disamping *input symbol* dan *blank*. Terdapat sebuah *tape head* yang selalu ditempatkan pada salah satu dari sel-sel *tape*. Mesin Turing dikatakan memindai sel tersebut. Pada awalnya, *tape head* berada pada sel paling kiri yang menampung *input*.

2.3. Notasi Formal MT

Mesin Turing dijelaskan dengan 7-*tuple*, yaitu:

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ (1)

Komponen-komponennya adalah:

- Q: Himpunan berhingga dari *state* dari *finite control*.
- Σ: himpunan berhingga dari simbol-simbol *input*.
- Γ: Himpunan dari *tape symbol*. Σ merupakan subset dari Γ.
- δ: Fungsi transisi. Argumen δ(q, X) adalah sebuah *state* q dan sebuah *tape symbol* X. Nilai dari δ(q, X), jika nilai tersebut didefinisikan, adalah triple (p, Y, D), dimana:
  - p adalah *next state* dalam Q
  - Y adalah simbol, dalam Γ, ditulis dalam sel yang sedang di-*scan*, menggantikan simbol apapun yang ada dalam sel tersebut.
  - D adalah arah, berupa L atau R, berturut-turut menyatakan *left* atau *right*, dan menyatakan arah dimana *head* bergerak.
- q_0: start *state*, sebuah anggota dari Q, dimana pada saat awal *finite control* ditemukan.
- B: simbol *blank*. Simbol ini ada dalam Γ tapi tidak dalam Σ, yaitu B bukan sebuah simbol *input*.
- F: himpunan dari final *state*, subset dari Q.

2.4. Mekanisme Kerja Mesin Turing

Cara kerja Mesin Turing dapat dideskripsikan sebagai berikut:

- Mula – mula untai ditempatkan dibagian paling kiri dari *tape*.
- Sisa dibagian kanan diisi simbol *blank*
- *Tape head* menunjuk pada sel paling kiri
- Program bermula pada *state START*
- Kalau tercapai *state Halt*, komputasi dihentikan, untai diterima mesin turing.

- Jika tak ada jalan untuk melanjutkan proses, maka untai tersebut ditolak mesin.

Sebuah pergerakan Mesin Turing adalah sebuah fungsi *state* dari *finite control* dan *tape symbol* yang dipindai. Dalam satu pergerakan, Mesin Turing akan:

- Merubah *state*. *Next state* dapat sama dengan *current state*.
- Menulis sebuah *tape symbol* dalam sel yang dipindai. *Tape* symbol ini mengganti simbol apapun yang ada dalam sel tersebut. Secara opsional, simbol yang dituliskan dapat sama dengan simbol yang sekarang ada dalam *tape*.
- Memindahkan *tape head* ke kiri atau ke kanan.

Pergerakan Mesin Turing direpresentasikan dengan R = *right*/kanan dan L = *left*/kiri. Fungsi transisinya dinyatakan dengan: $\delta(q_1, a) = (q_1, a, R)$ (2)

dibaca pada *state* q_1, *head* menunjuk karakter 'a' pada pita menjadi *state* q_1, *head* bergerak ke kanan.

Prinsip dalam menggerakkan Mesin Turing , yaitu 1) melihat *state* semula dan simbol yang ditunjuk *head*. 2) berdasarkan fungsi transisinya menentukan *state* berikutnya, dan melakukan penulisan ke pita, serta menggerakkan *head* ke kanan atau ke kiri. 3) bila dari pasangan (*state*, yang ditunjuk *head*) tidak ada lagi transisi, berarti mesin turing berhenti. 4) bila Mesin Turing berhenti di dalam *final state* berarti *input* diterima, jika sebaliknya berarti *input* ditolak.

2.5. Deskripsi Instantaneous (ID) untuk Mesin Turing

ID digunakan untuk mengetahui apa yang dikerjakan oleh Mesin Turing. ID direpresentasikan oleh string $X_1X_2X_3\ldots X_{i-1}qX_iX_{i+1}\ldots X_n$, dimana:

- q adalah *state* dari Mesin Turing
- *Tape head* memindai simbol ke-i dari kiri.
- $X_1X_2\ldots X_n$ adalah bagian dari *tape* diantara *non-blank* pada sel paling kiri dan paling kanan.

Pergerakan Mesin Turing M = (Q, Σ, Γ, δ, q_0, B, F) dinyatakan oleh notasi ⊢ atau ⊢ (dibaca: berubah menjadi). $\vdash^*_M$ atau $\vdash^*$ digunakan untuk menunjukkan nol, satu atau lebih pergerakan dari Mesin Turing. Asumsikan bahwa $\delta(q, X_i) = (p, Y, L)$, yaitu pergerakan selanjutnya adalah ke kiri. Maka $X_1X_2\ldots X_{i-1}qX_iX_{i+1}\ldots X_n \vdash X_1X_2\ldots X_{i-2}pX_{i-1}YX_{i+1}\ldots X_n$. Pergerakan ini menyatakan perubahan ke *state* p. *Tape head* sekarang diposisikan di sel i-1. Terdapat dua pengecualian:

– Jika i=1, maka M bergerak ke *blank* ke bagian kiri dari X1. Dalam kasus ini, $qX_1X_2\ldots X_n \vdash pBYX_2\ldots X_n$

– Jika i = n dan Y = B maka simbol B yang ditulis pada Xn berhubungan dengan urutan tak-hingga dari *blank-blank* yang mengikuti dan tidak muncul dalam ID selanjutnya. Dengan demikian $X_1X_2\ldots X_{n-1}qX_n \vdash X_1X_2\ldots X_{n-2}pX_{n-1}$

Anggap $\delta(q, Xi) = (p, Y, R)$, yaitu pergerakan selanjutnya adalah ke kanan. Maka $X_1X_2\ldots X_{i-1}qX_iX_{i+1}\ldots X_n \vdash X_1X_2\ldots X_{i-1}YpX_{i+1}\ldots X_n$. *Tape head* telah bergerak ke sel i+1. Terdapat dua pengecualian:

- Jika i = n, maka sel ke-i+1 menampung sebuah *blank*, dan sel tersebut bukan bagian dari ID sebelumnya. Dengan demikian $X_1X_2 \ldots X_{n-1}\, qX_n \vdash X_1\, X_2 \ldots X_{n-1}YpB$
- Jika i = 1 dan Y = B maka simbol B yang ditulis pada X1 berhubungan dengan urutan tak hingga dari *blank-blank* dan tidak muncul dalam ID selanjutnya. Dengan demikian $qX_1X_2 \ldots X_n \vdash pX_2 \ldots X_n$

2.6. Diagram Transisi untuk Mesin Turing

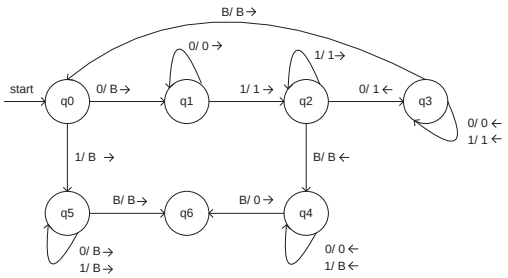
Diagram transisi terdiri atas sebuah himpunan *node-node* yang menyatakan *state-state* Mesin Turing. Sebuah *arc* dari *state* q ke *state* p diberi label oleh satu atau lebih item dengan bentuk X/Y D, dimana X dan Y adalah *tape* symbol, dan D adalah arah, kiri (L) atau kanan (R). Bahwa bila $\delta(q, X) = (p, Y, D)$ diperoleh label X/Y D pada *arc* dari q ke p. Dalam diagram arah D dinyatakan dengan tanda ← untuk *"left"* dan → untuk *"right"*. *Start state* ditandai dengan kata "start" dan sebuah panah yang masuk ke dalam *state* tersebut. *Final state* ditandai dengan putaran ganda. Sebagai contoh adalah Mesin Turing berikut menghitung fungsi yang dinamakan monus atau *proper substraction*. Fungsi ini didefinisikan oleh m n = max(m – n, 0). Bahwa, m n = m – n jika m ≥ n dan 0 jika m < n. Mesin Turing yang melakukan operasi ini adalah $M = (\{q_0, q_1, \ldots, q_6\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B)$.

Aturan untuk fungsi transisi δ disajikan pada tabel dibawah ini.

Tabel 1. Aturan bagi fungsi transisi δ

State	Simbol		
	0	1	B
q_0	(q_1, B, R)	(q_5, B, R)	-
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	-
q_2	$(q_3, 1, L)$	$(q_2, 1, R)$	(q_4, B, L)
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, B, R)
q_4	$(q_4, 0, L)$	(q_4, B, L)	$(q_6, 0, R)$
q_5	(q_5, B, R)	(q_5, B, R)	(q_6, B, R)
q_6	-	-	-

Diagram transisi dari mesin Turing M ditunjukkan oleh gambar berikut.



Gambar 2. Model diagram transisi Mesin Turing berdasar Table 1

3. CELLULAR AUTOMATA

Cellular Automata diperkenalkan pertama kali oleh John Von Neuman dan Stanislaw Ulam sekitar tahun 1940an sebagai model sederhana guna mempelajari proses biologi seperti *self reproduction organism*. Selanjutnya *Cellular Automata* lebih berkembang dengan dibuatnya *Game of live* oleh John Conway sekitar tahun 1960an.

3.1 Pengertian Celullar Automata

Cellular Automata adalah sebuah *array* dengan automata yang identik atau disebut juga sel yang saling berinteraksi satu dengan lainnya. *Array* tersebut dapat membentuk susunan sel 1, 2 maupun 3

dimensi. Susunan sel tersebut juga dapat membentuk grid segi empat sederhana maupun susunan lain, seperti sarang lebah yang terdiri dari bagian-bagian berbentuk segi enam.

Unsur pembentuk *Cellular Automata* terdiri atas:

a. Geometri, yaitu bentuk sel serta bentuk sistem yang disusun oleh sel-sel tersebut. Geometri *Cellular Automata* terdiri atas dimensi *Cellular Automata* tersebut (1-d, 2-d, dan seterusnya) serta bentuk geometri dari masing-masing sel penyusunnya.

b. *State* set adalah himpunan keadaan atau status yang dapat dimiliki oleh masing-masing sel *Cellular Automata* tersebut. Status ini dapat berupa angka atau sifat tertentu. Misalnya bila masing-masing sel merepresentasikan hutan maka status dapat merepresentasikan misalnya jumlah binatang pada masing-masing lokasi atau jenis pohon-pohon yang tumbuh. *State* set haruslah berhingga (*finite*, terbatas) dan terhitung (*countable*, diskrit)

c. *Neighbourhood* atau ketetanggaan ialah sel-sel yang dapat mempengaruhi status suatu sel pada *Cellular Automata*. Umumnya *neighbourhood* suatu sel hanya meliputi sel-sel yang berada di sekitarnya. Berdasarkan strukturnya ada beberapa jenis *neighbourhood* yang telah dikenal secara umum, antara lain geometri dua dimensi, yaitu Von Neuman *neighbourhood*, Moore *neighbourhood*, dan Margolus *neighbourhood*.

d. Fungsi transisi adalah aturan yang menentikan bagaimana status suatu sel berubah berdasarkan status sekarang dan status tetangganya.

e. Status awal sel adalah status yang dimiliki oleh masing-masing sel pada saat sistem mulai berjalan.

3.2 Definisi Formal Celullar Automata

Secara formal, *Cellular Automata* didefinisikan sebagai 5-tuples, yaitu (L, Q, N, δ, Co), dengan

- L: geometri (bentuk sel serta bentuk sistem yang disusun oleh sel-sel tersbut) dan d adalah dimensi bentuk gemetrinya;
- Q: himpunan berhingga *state* dengan k adalah jumlah *state* masing-masing sel;
- N: himpunan sel yang mempengaruhi *state* tersebut pada langkah berikutnya dengan n adalah jumlah *neighbourhood* ke salah satu sisi yang mempengaruhi sel tersebut dan r adalah jari-jari *neighbourhood*;
- δ: $Q^{n+1} \rightarrow Q$. (Q^{n+1}: tuple yang terdiri atas *state* dirinya sendiri dan *state* sel-sel tetangganya). Fungsi transisi, dioperasikan pada sistem untuk sejumlah langkah, pada tiap langkah fungsi dilakukan serentak pada semua sel. *Input* fungsi ini adalah dirinya sendiri dan *state* tetangganya pada time step sebelumnya;
- Co: konfigurasi awal sistem. Konfigurasi awal adalah himpunan yang berisi *state* tiap-tiap sel *Cellular Automata* pada time step 0 atau pada waktu sistem mulai berjalan.

3.3 Karakteristk Celullar Automata

Karakteristik *Cellular Automata* antara lain sebagai sistem diskret yang dinamis, *locality, parallelism*, dan *emergent*. Adanya karakteristik ini maka *Cellular Automata* cocok digunakan untuk diimplementasikan pada lingkungan parallel.

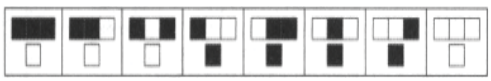
a. Sistem diskret yang dinamis, yaitu sistem memiliki spesifikasi berikut:
   - Memiliki entitas yang berubah seiring dengan berjalannya waktu. Perubahan ini disebabkan oleh adanya faktor internal dari sistem itu sendiri.
   - Entitas-entittas yang menyusun penyusun sistem terhitung (*countable*)
   - Perubahan entitas terjadi dalam waktu yang diskret (*per time step*)

b. *Locality*, berarti ketika sebuah sel berubah, status barunya hanya dipengaruhi oleh status lama dan status tetangganya. Oleh karena umunya tetangga suatu sel hanya meliputi sel-sel sekitarnya saja maka dapat dikatakan bahwa perubahan status dari tiap sel hanya bergantung pada dirinya dan sel-sel disekitarnya saja.

c. *Parallism* berarti perubahan masing-masing sel dapat dilakukan dengan tidak bergantung pada sel lain sehingga semua sel dapat diperbaharui secara serentak. Oleh karenya *Cellular Automata* pada dasarnya cocok untuk diimplementasikan pada lingkungan parallel.

d. *Emergent*, yaitu tiap sel penyusun *Cellular Automata* hanya melakukan fungsi-fungsi sederhana yang sepertinya kurang terlalu bermanfaat. Akan tetapi ketika sel-sel tersebut dilihat sebagai satu kesatuan maka akan menjadi sebuah sistem yang dapat menghasilkan sesuatu yang besar. Sehingga nampaknya seolah-olah sistem tersebut muncul dengan tiba-tiba (*emerges*).

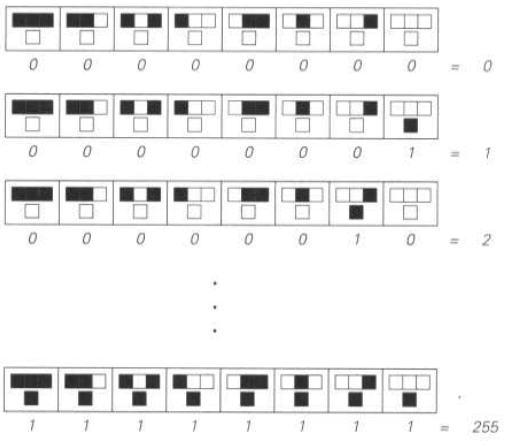
Cellular Automata satu dimensi berada di sebuah *array* horisontal tak-hingga pada sel-sel. Pada paper ini ditinjau sebuah *Cellular Automata* dengan sel persegi yang terbatas hanya oleh dua kemungkinan *state* per sel, yaitu putih dan hitam. Aturan *Cellular Automata* dalam menentukan pengaturan tak-hingga sel hitam dan putih yang diperbarui dari waktu ke waktu didasarkan pada skema tetangga terdekat. Ini artinya bahwa untuk menentukan *state* dari sebuah sel di posisi p pada langkah waktu *(time step)* t +1, diamati *state-state* dari sel-sel di posisi p - 1, p, dan p + 1 yang kesemua dalam langkah waktu t. Untuk masing-masing dari delapan kemungkinan pola-pola sel putih dan hitam, dipilih keadaan sel p pada langkah waktu t + 1 baik sebagai hitam atau putih. Pada Gambar 3 ditampilkan delapan peluang/kemungkinan pola masukan, serta sejumlah 256 kemungkinan output yang berbeda.



Gambar 3. Pada bagian atas dari gambar diilustrasikan delapan kemungkinan pola sel tiga, dua-*state* secara berurut dari kiri ke kanan. Sementara pada bagian bawah adalah salah satu kemungkinan dari kumpulan output-output. Secara

keseluruhan, ada 256 kemungkinan output yang berbeda. Gambar ini dipindai dari Wolfram (2002:53).

Guna menganalisa perilaku program ini Wolfram (2002) mengembangkan konvensi penamaan, yaitu sebuah metode yang memandang kondisi awal dan hasil dari beberapa iterasi secara sekaligus. Pada *Cellular Automata* satu dimensi yang dijelaskan di atas, sebuah hirarki diberikan pada delapan kemungkinan pola dengan warna hitam-hitam-hitam-hitam di sisi paling kiri dan putih-putih-putih pada sisi paling kanan. Setiap kombinasi merepresentasikan suatu tempat dalam sistem penomoran biner. Hitam-hitam-hitam, misalnya, adalah representasi tempat ke-27. Penetapan nilai nol ke putih dan satu ke hitam memberikan masing-masing susunan yang mungkin dari skenario-skenario pembaharuan suatu bilangan biner yang berkisar dari 0 sampai 255 dalam basis sepuluh. Gambar 4 menyajikan beberapa contoh skema penamaan tersebut.

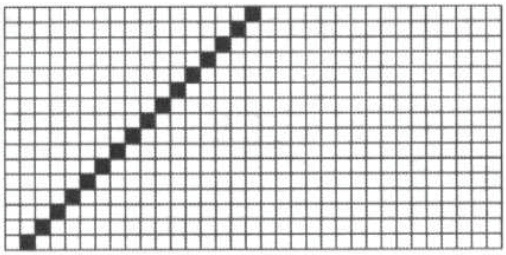


Gambar 4. Ini adalah tiga aturan pertama dan yang terakhir. Urutan angka satu dan nol adalah bilangan biner dengan basis sepuluh. (Wolfram, 2002:53).

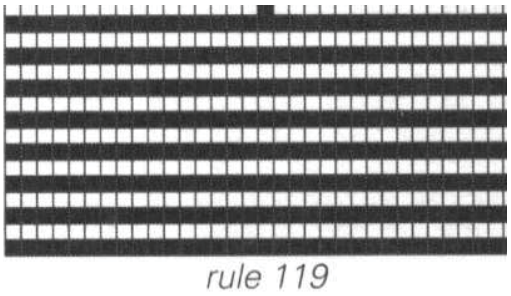
Kondisi awal pada keseluruhan aturan tersebut, 0-255 terdiri dari satu sel hitam dengan sinar-sinar sel putih yang memanjang hingga tak terbatas di kedua sisi. Untuk melihat beberapa iterasi sekaligus, setiap langkah waktu ditetapkan di bawah sebelum posisi-posis masing-masing sel yang tidak berubah, sebagaimana ditunjukkan pada Gambar 5. Pada Gambar 5 tersebut disajikan sebuah gambar dua dimensi dari beberapa iterasi sebuah *Celullar Automata* yang memungkinkan untuk dianalisis perilakunya. Perlu dicatat bahwa tingkat maksimum perjalanan dari kotak hitam di tengah adalah satu persegi lateralis per iterasi.

Wolfram (2002) mengklasifikasikan perilaku yang diamati dalam empat kelas yang berbeda dari *Celullar Automata*. Yang pertama adalah Kelas I, yang berisi perilaku berulang sederhana. Hal ini dapat berkisar dari sebuah baris vertikal atau diagonal tunggal (kondisi awal tetap) seperti dalam aturan 100 dan aturan 106, atau serangkaian iterasi yang bertukar-tukar semua putih dan semua hitam seperti dalam aturan 119 dan 21. Pada Gambar 5, gambar (a) dan (b) menunjukkan aturan 106 dan 119 masing-masing untuk contoh perilaku Kelas I. Perilakunya dapat dikenali dengan mudah yang mana mengandung unsur-unsur berulang dengan ukuran sama yang mencakup seluruh program. Sekitar 86% dari 256 dasar *Celullar Automata* adalah kelas ini. Kelas II ditandai dengan *Celullar Automata* yang mempunyai pola-pola tersarang *(nested)*. Pola tersarang adalah konfigurasi yang berulang sendiri dalam skala yang semakin meningkat. Artinya, representasi skala yang lebih kecil

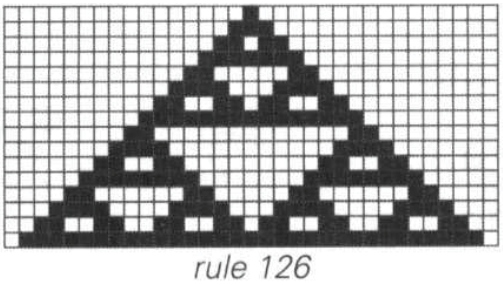
dari wilayah yang dipilih terjadi di dalam daerah itu sendiri. Sekitar 9% dari dasar *Celullar Automata* adalah kelas ini. Gambar 3(c) menyajikan gambar pola tersarang Kelas II tersebut dalam aturan 126.



(a)

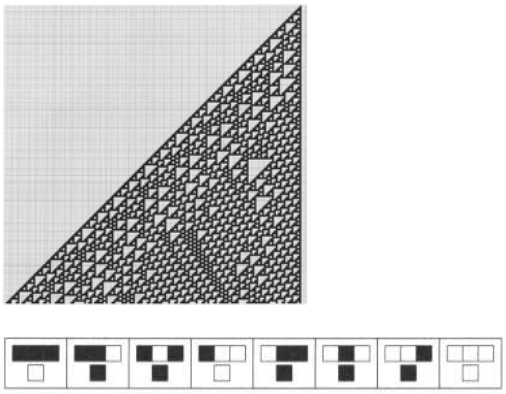


(b)



Gambar 5. 16 iterasi aturan 106, 119, dan 126. Peraturan 106 dan 119 adalah contoh perilaku Kelas I, dan 126 adalah salah satu perilaku Kelas II. (Wolfram, 2002:54).

Perilaku kelas III adalah benar-benar acak. *Celullar Automata* di kelas ini memiliki bentuk yang berulang, tapi lokasi dan frekuensinya acak. Kelas ini berisi sekitar 4% dari dasar *Celullar Automata*. Kelas terakhir adalah perilaku Kelas IV, yaitu kombinasi dari perilaku Kelas I dan perilaku Kelas III. *Celullar Automata* dalam kelas ini menyajikan suatu kombinasi yang kompleks dari perilaku acak dan pola yang berulang, sebagaimana ditunjukkan oleh contoh pada Gambar 6. Hanya ada 4 dari 256 dasar *Celullar Automata* yang menunjukkan perilaku ini, dan keempatnya pada dasarnya sama ketika dipertimbangkan simetri hitam-putih dan simetri kiri-kanan. Dua diantaranya adalah suatu bayangan cermin satu sama lain, pembalikan sumbu vertikal ditempatkan di lokasi sel hitam dalam kondisi awal. Dua lainnya adalah sama dengan dua yang pertama di mana keadaan setiap sel dibalik, (setelah langkah pertama kalinya).



Gambar 6. 150 iterasi aturan 110 dan aturan memperbaruinya. (Wolfram, 2002:32).

4. UNIVERSALITAS KOMPUTASI

Universalitas komputasi adalah kemampuan dari sebuah mesin atau program untuk menghitung iterasi dari mesin atau program yang lain. Hal inilah yang merupakan konsep lahirnya revolusi komputer. Dengan menggunakan terminologi dari komputasi modern, mesin komputasi secara universal adalah analogis dengan "*hardware*", sementara tugas yang mungkin dapat dilakukan (dari

mesin lain) adalah analogis untuk "*software*". *Hardware* dan *software* berbeda hanya dalam hal bahwa yang pertama adalah universal secara komputasi dan yang kedua adalah tidak. Konsep universalitas komputasi pertama kali ditemukan oleh Alan Turing pada saat bekerja dengan mesin Turing di tahun 1950-an.

Mesin Turing dimaksudkan untuk melakukan perhitungan algoritmis dengan mengikuti langkah-langkah yang seorang manusia akan pekerjakan. Langkah-langkah dasar yang diambil manusia dipecah kedalam elemen-elemen penting dalam bentuk membaca dan menggantikan simbol, dan bergerak dari simbol ke simbol. Untuk menirukan proses itu, mesin Turing menggunakan sejumlah set simbol, sejumlah set *state*, dan rekaman (*tape*) tak-hingga dari sel-sel (sama seperti *Celullar Automata* satu dimensi) dan sebuah *head* mesin sebagaimana telah dipaparkan pada bagian di atas. Mesin Turing universal bahkan mampu melakukan algoritma dari mesin yang lebih rumit (lebih banyak simbol atau *state*) dari yang ada pada dirinya sendiri. Ini adalah sebuah konsekuensi bahwa algoritma yang lain yang mana dapat dilakukan oleh sebuah mesin Turing maka dapat dilakukan oleh mesin Turing universal. Hal ini juga konsekuensi bahwa tidak ada program yang bisa lebih kompleks secara komputasional daripada sebuah mesin universal secara komputasi.

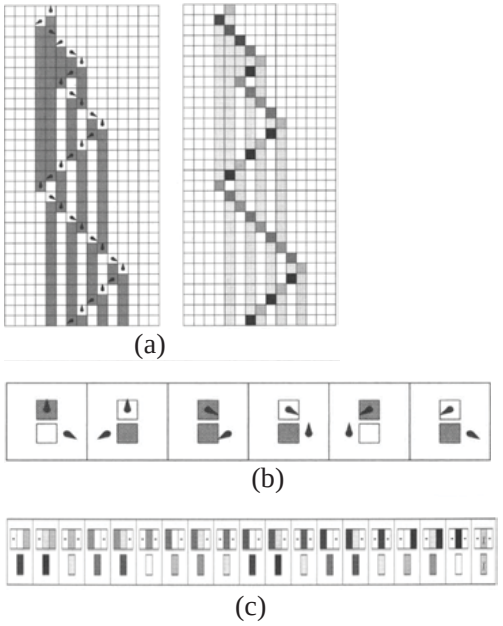
5. EMULASI DAN UNIVERSALITAS CELULLAR AUTOMATA

Banyak program atau mesin lain, seperti misalnya *Celullar Automata* , mesin register, sistem-sistem substitusi, atau mesin tag yang juga dapat terbukti sebagai komputasi yang universal. Karena bukti yang ada dari universalitas komputasi hanya berkaitan dengan Mesin Turing asli, semua program-program lain dibuktikan universalitasnya secara komputasional melalui suatu emulasi.

Emulasi berarti bahwa serangkaian iterasi dalam satu program menghasilkan suatu representasi setara (*equivalent*) dengan setiap langkah komputasi/perhitungan program yang ditiru. Untuk mengemulasikan sebuah mesin Turing dengan sebuah *Celullar Automata*, iterasi dari mesin Turing harus ditampilkan secara vertikal seperti dalam *Celullar Automata* sebagaimana dibahas di atas. Pada setiap iterasi mesin Turing menunjukkan simbol-simbol ini dan posisi head. Dalam *Celullar Automata* yang mengemulasi mesin Turing, sebuah warna ditujukan untuk setiap kemungkinan kombinasi *state* dan simbol, serta satu warna untuk setiap simbol jika tidak terfokus oleh *head*, dan dengan demikian tidak terhubung ke sebuah *state*. Gambar 7 (a) menunjukkan mesin Turing dengan dua simbol (warna) dan tiga *state* dan setara *Celullar Automata*. *Celullar Automata* yang mengemulasi Mesin Turing memiliki delapan warna (2 symbols * 3 *state* + 2 symbols). Untuk tujuan organisasional, sel-sel dari mesin Turing dimana *head* tidak terfokus diwakili oleh dua warna paling terang di dalam *Celullar Automata*. Enam warna lebih gelap mewakili gerakan dan *state* dari *head*. Sekumpulan aturan tetangga terdekat untuk komputasi *Celullar Automata* kemudian dihasilkan dari tabel mesin Turing. Pada Gambar 7, (b) dan (c) ditampilkan aturan-aturan dari tabel mesin Turing dan *Celullar Automata*.

Contoh emulasi ini dapat dikembangkan pada mesin Turing dengan jumlah simbol dan *state* yang lebih besar. Jumlah warna yang digunakan dalam *Celullar Automata* meningkat dengan cepat, seperti halnya jumlah kasus

yang aturan-aturannya perlu untuk dibuat. Penurunan aturan *Celullar Automata* tetap cukup sederhana oleh karena hanya satu sel yang diperbarui per iterasi.



Gambar 7. Gambar (a) adalah emulasi dari mesin Turing dengan sebuah *Celullar Automata*. Setiap dari kombinasi 6 symbol-*state*, serta kedua kombinasi simbol-no-*state*, diberi suatu warna dalam *Celullar Automata*. Gambar (b) adalah tabel mesin untuk mesin Turing. Pointer dan warna mewakili *state* dan simbol. Gambar (c) menunjukkan aturan *Celullar Automata* yang berasal dari tabel mesin Turing. Sel-sel putih dengan sebuah garis horizontal pada Gambar (c) mengartikann bahwa warna lainnya dapat di sel itu. (Wolfram, halaman 658)

Adanya metode pengemulasian mesin Turing dengan *Celullar Automata* akan mengakibatkan kerumitan yang ekstrim pada *Celullar Automata* bahkan untuk yang paling sederhana pun dari mesin Turing universal.

6. AMBANG UNIVERSALITAS

Konsep universalitas komputasional menyiratkan bahwa sekali tingkat kerumitan tertentu tercapai, tidak ada keuntungan didalam kemampuan komputasional. Hal ini secara khusus tersirat oleh kemampuan mesin Turing universal untuk meniru perilaku mesin Turing dengan tabel mesin yang lebih rumit daripada yang dimilikinya sendiri. Tingkat komplikasi di mana universalitas tercapai disebut ambang universalitas (*threshold of universality*). Intuisi-intuisi tradisional dikembangkan selama revolusi komputer menempatkan ambang ini menjadi cukup tinggi. Terdapat asumsi bahwa sebuah mesin yang mampu melakukan perhitungan kompleks tersebut terbuat baik dari bagian-bagian yang rumit atau bagian sederhana yang disatukan dengan cara yang sangat kompleks. Hasil-hasil paparan pada sesi di atas menunjukkan bahwa dalam kenyataannya salah satu dari 256 *Celullar Automata* yang paling sederhana adalah universal. Ambang batas ini sebenarnya cukup rendah. Meskipun aturan *Celullar Automata* 110 adalah satu-satunya contoh dalam paparan ini, sebenarnya ada banyak jenis dari mesin dan program lain dalam *"A New Kind of Science"* yang menampilkan komputasi universal dan dalam bentuk yang sederhana. Hampir semua contoh ini termasuk dalam perilaku Kelas IV, sementara segelintir kecil saja dari Kelas III.

Pada dasarnya, Wolfram berteori bahwa semua sistem di alam semesta yang menunjukkan kelas III atau berperilaku Kelas IV adalah universal secara komputasional dan dengan demikian pada dasarnya adalah setara. Jika semua sistem yang dilaksanakan menghitung perilakunya sendiri, dan sistem tersbut adalah universal secara komputasional, maka ini dapat mereproduksi perilaku sistem lain, dan semua sistem yang setara. Kunci untuk argumen ini tentu saja bahwa ambang universalitas dicapai oleh semua sistem yang memproduksi perilaku Kelas III dan Kelas IV.

Terdapat dua sudut tantangan pada prinsip di atas. Yang pertama adalah bahwa terdapat proses yang lebih rumit daripada apa yang dilihat dalam program universal, seperti proses yang terus menerus atau pemikiran manusia. Wolfram merespon contoh pertama dengan menegaskan bahwa hal itu mempunyai tantangan sama halnya untuk meniru sistem kontinu dengan model diskrit atau untuk meniru sistem diskrit dengan model kontinu. Hal ini menyiratkan bahwa tingkat kompleksitasnya adalah sama. Wolfram merespon isu pemikiran manusia dengan mengekspresikan keyakinannya bahwa kemajuan dalam ilmu neuro akan mengarah pada pemahaman tentang otak dalam hal program komputasi sederhana.

Tantangan lainnya adalah bahwa terdapat sejumlah proses Klas III dan program-program seperti aturan 30, yang tidak universal. Wolfram berspekulasi bahwa banyak aturan seperti Kelas III akan ditampilkan untuk komputasi yang universal di masa depan.

7. KESIMPULAN

Penelitian yang telah dilakukan oleh Stephen Wolfram pada perilaku program dan konsep universalitas komputasi memiliki makna yang besar bagi bidang ilmu komputer. Selain menambah catatan sejarah ilmu komputer, hasil penelitian Wolfram juga menunjukkan potensi untuk berkontribusi pada teknologi berbasiskan komputer.

Mesin Turing dimaksudkan untuk melakukan perhitungan algoritmik dengan mengikuti langkah-langkah yang akan dikerjakan seorang manusia. Karena bukti yang ada dari universalitas komputasi hanya berkaitan dengan Mesin Turing asli, semua program-program lain dibuktikan universalitasnya secara komputasional melalui emulasi. Adanya metode pengemulasian mesin Turing dengan *Celullar Automata* akan mengakibatkan kerumitan yang luar biasa pada *Celullar Automata* bahkan untuk hal yang paling sederhana pun dari mesin Turing universal.

Prinsip ekivalensi komputasi adalah mengenai sifat komputasi dan kompleksitas yang diimplementasikan pada sistem yang lain selain yang ada di dalam komputer. Proses komputasi dan ambang universalitas dihipotesiskan akan hadir di setiap sistem alami di alam semesta ini. Pada dasarnya semua proses-proses yang ada adalah sebuah sistem yang melakukan suatu komputasi dengan kokpleksitas yang setara. Satu-satunya perbedaan antara perhitungan *Celullar Automata* dan proses alami adalah bahwa kita tidak mengetahui aturan yang proses alami ikuti.

DAFTAR PUSTAKA

Davis, M. 2000. The Universal Computer: The Road from Leibniz to Turing. New York: Norton.

Hopcroft, J.E., R. Motwani, and J. D. Ullman. 2001. Introduction to Automata Theory, Languange, and Computation. Edisi ke-2. Addison-Wesley

Maida, K., dan C. Sakama. 2007. Identifying Celullar Automata rules, in Journal of Celullar Automata, Vol. 2, pp. 1-20.

Martin, O., A. M. Odlyzko, and S. Wolfram. 1984. Algebraic properties of Celullar Aotumata, Communications in Mathematical Physics, Springer-Verlag.

Mitchell, M. 1998. Computation in Celullar Automata, in Nonstandard Computation, pp. 95-140. Weinheim

Sipser, M. 2013. Introduction to the theory of computation, 3rd Ed, Cengage Learning, Boston USA.

Wegener, I. 2003. Complexity Theory: Exploring the limits of efficient algorithms, Springer-Verlag, Berlin.

Wolfram, S. 2002. A New Kind of Science. Champaign, IL: Wolfram Media Inc.

Zvi Kohavi, Z. 2005. Switching and Finite Automata Theory, McGraw-Hill.