Teori Bahasa & Otomata

Heri Sutarno - 131410892
Pendilkom/Ilkom
Universitas Pendidikan Indonesia
Bandung, 2008

Buku Bacaan

- Aho, Alfred V., Ravi Sethi and Jeffrey D Ulman, *Compilers: Principles, Techniques, and Tools*, Addison-Wesley Publishing Company, Reading Massachusetts; 1986
- Hopcroft, John E., dan Jeffrey D.Ulman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing Company, Reading Massachusetts; 1979

- Teori Bahasa, Otomata, dan komputasi sering hendak dihindari mahasiswa ilmu informatika/komputer, karena pemahaman teori ini perlu abstraksi kuat.
- Teori ini sering terlupakan, padahal inilah ‘penggerak tak tampak’ perkembangan teknologi informasi yang sedemikian pesat baik pada perangkat keras maupun perangkat lunak.

- Sebelum komputer lahir, David Hilbert telah mencoba menciptakan algoritma umum untuk pembuktian persoalan matematika secara otomatis. Setelah itu lahir teori otomata. Teori dimulai di bidang sistem logika matematika yang mencoba membuat program yang mampu menentukan salah dan benarnya sembarang proposisi matematika.

- Teori otomata mempelajari model mesin komputer menggunakan model matematika. Namun matematika yang digunakan benar-benar berbeda dibanding matematika klasik dan kalkulus.

- Model yang digunakan adalah model mesin *state* (*state machine model*) atau model transisi *state* (*state transition model*).

- Pengembangan teori otomata, komputasi, dan teori bahasa berikutnya difasilitasi perkembangan bidang *psycho-linguistic*. Sekitar tahun 1950, Noam Chomsky menciptakan model matematika sebagai sarana untuk mendeskripsikan bahasa serta menjawab pertanyaan-pertanyaan sekitar *psycho-linguistic*.
- Sekitar tahun 1950, Noam Chomsky mengemukakan perangkat formal yang disebut ***grammar*** untuk memodelkan properti-properti bahasa.

- *Grammar* berisi sejumlah aturan serta menspesifikasikan bahasa tertentu. Bahasa berisi semua string yang dapat dihasilkan menggunakan aturan-aturan *grammar*.
- *Grammar* mempunyai manfaat/nilai sangat besar di ilmu informatika/komputer karena pencapaian ini digunakan untuk mendeskripsikan dan mendefinisikan sintaks bahasa pemrograman dan bahasa-bahasa formal yang lain.

Otomata dibedakan berdasar jenis memori sementara yang dimilikinya, yaitu:

1. *Finite automata* (FA) tidak memiliki memori sementara. FA adalah kelas mesin dengan kemampuan-kemampuan paling terbatas.
2. *Pushdown automata* (PDA) memiliki memori sementara dengan mekanisme LIFO, yaitu mekanisme *stac*k.

3. *Turing machine* (TM) memiliki memori dengan mekanisme pengaksesan acak (*random access memory*). TM merupakan model matematika untuk komputer saat ini.Contoh otomata ini adalah algoritma yang memiliki keampuhan komputasi paling tinggi.

Bab 1 Pendahuluan

* Komponen Ilmu Informatika
  - Ide & model fundamental yang mendasari komputasi
  - Teknik rekayasa untuk perancangan sistem komputasi

  Salah satu ide dan model fundamental penting adalah **model komputasi**.

- Bidang-bidang ilmu lain yang menyumbang perkembangan model komputasi diantaranya:

  1. Ahli biologi mempelajari *neuron nets* menemukan *finite automata.*

  2. Rekayawan listrik mengembangkan teori *switching* sebagai perancangan perangkat keras menggunakan *finite automata*.

  3. Matematikawan mengembangkan sistem logika formal untuk pembuktian.

- Gagasan dasar *finite automata* adalah sangat umum yaitu sistem pada satu saat berada di salah satu *state* dari sejumlah *state*, bergerak diantara *state*-*state* itu secara dapat diprediksi yang bergantung pada masukan ke sistem.
- Salah satu penerapan penting model komputasi yang paling dekat adalah kompilasi atau translasi bahasa pemrograman tingkat tinggi menjadi bahasa mesin yang ekivalen.

- Bahasa yang dikenali *finite automata* adalah bahasa sederhana namum mempunyai aspek penerapan sangat penting di ilmu informatika/komputer.

- Mesin Turing seperti komputer modern saat ini. Dapat mengolah masukan dan menghasilkan keluaran.

* Model Komputasi

Tiga Model Komputasi

- *Finite automata/finite state automata* (FSA)
  - Deterministic finite automata (DFA)
  - Nondeterministic finite automata (NDFA)
- *Pushdown automata* (PDA)
  - Deterministic pushdown automata (DPA)
  - Nondeterministic pushdown automata (NDPA)
- *Turing machine* (TM)

- **Finite automata dan ekspresi regular**

  Menjadi alat sangat berguna dalam perancangan bagian kompilator yang bertugas mengelompokkan karakter-karakter menjadi token-token, yaitu unit bahasa terkecil seperti *keyword*, *identifier* dsb, bertindak sebagai kata di bahasa sehari-hari.

- **Pushdown automata** dan **context free grammar**

  Chomsky menunjukkan bahasa *context free* ekivalen mesin abstrak *pushdown automata*. Maksud ekivalen adalah untuk sembarang *context free grammar* terdapat *pushdown automaton* yang dapat mengenali bahasa hasil *context free grammar* itu. *Pushdown automaton* mengolah sembarang string dan menentukan apakah string itu termasuk bahasanya.

- **Mesin Turing (*Alan Turing Machine*)**

  Merupakan pemodelan mesin komputasi yang ampuh. Berdasarkan mesin Turing dapat diidentifikasi ketidakmungkinan penulisan program. Bila dinyatakan tidak dapat dikomputasi mesin Turing berarti persoalan tidak mungkin dapat diselesaikan secara komputasi dengan mesin komputasi apapun.

* Teori Komputasi

- Apa yang dimaksud dengan mengkomputasi?
- Apa yang dapat dikomputasi?
- Seberapa kompleks untuk mengkomputasi sesuatu?

- Agar tidak bergantung pada komputer atau teknologi tertentu, teori mengusulkan mesin dan bahasa abstrak yang dipandang sebagai model komputasi. Mesin abstrak ini seampuh kompter nyata. Segala sesuatu yang dapat dikomputasi dengan mesin abstrak disebut *computable.*
- Komputasi sangat erat hubungannya dengan algoritma. Algoritma adalah prosedur yang untuk persoalan yang didefinisikan mangalami perhentian.

Bab 2 Matematika Dasar

*Himpunan

- Himpunan bagian, $A \subseteq B$
- Penggabungan, $A \cup B$
- Irisan, $A \cap B$
- Complement (Relative/Absolute)
- Cartesian Product,

$$A \times B = \{(x, y) \mid (x \in A) \text{ dan } (y \in B)\}$$

* Relasi

Sifat-sifat relasi:

- Reflexive, $\forall x \in X, xRx \Rightarrow (x,x) \in R$
- Symmetric, $x, y \in X, xRy \Rightarrow yRx$
- Transitive, $x, y, z \in X, xRy \ \& \ yRz \Rightarrow xRz$
- Irreflexive, $\forall x \in X \Rightarrow (x,x) \notin R$
- Antisymmetric $x, y \in X, xRy \ \& \ yRz \Rightarrow x = y$

Transitive Closure

- Definisi: Bila X adalah suatu himpunan berhingga dan R adalah relasi pada X. Relasi $R^+ = R \cup R^2 \cup R^3 \ldots$ pada X, disebut *transitive closure* R pada X.

- Teorema:

  Transitive closure R^+ relasi R pada suatu himpunan berhingga X adalah transitif.

  Juga untuk suatu relasi transitif P lain pada X dimana $R \subseteq P$, kita mempunyai $R^+ \subseteq P$. Dalam arti ini, R^+ adalah relasi transitif terkecil yang berisi R.

* Logika

$$\neg(\neg P) = P$$

$$(P \vee Q) = (\neg P \Rightarrow Q)$$

Hukum de Morgan

$$\neg(P \vee Q) = (\neg P \wedge \neg Q)$$

$$\neg(P \wedge Q) = (\neg P \vee \neg Q)$$

Hukum Distributif

$$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$$

$$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$$

Hukum Komutatif

$(P \wedge Q) = (Q \wedge P)$

$(P \vee Q) = (Q \vee P)$

Hukum Asosiatif

$((P \wedge Q) \wedge R) = (P \wedge (Q \wedge R))$

$((P \vee Q) \vee R) = (P \vee (Q \vee R))$

Hukum Kontrapositif

$(P \Rightarrow Q) = (\neg Q \Rightarrow \neg P)$

* Graf (*Graph*)

- Dua graph disebut ekivalen (*isomorphic*) jika keduanya berperilaku identik menurut kriteria-kriteria graph.
- Syarat perlu dua graph adalah *isomorphic*:
  - Jumlah simpul ke-2 graph sama
  - Jumlah busur ke-2 graph sama
  - Jumlah simpul yang sama dengan derajat yang diberikan

- Pohon (*tree*) adalah graph G dengan n simpul, jika:
  1. G terhubung dan tanpa sirkit, atau
  2. G terhubung dan n-1 busur, atau
  3. G tanpa sirkit dan mempunyai n-1 busur, atau
  4. Terdapat tepat satu path di antara pasangan simpul di G,atau
  5. G adalah graph terhubung minimal

- Algoritma Penelusuran Graph

  Terdapat beragam algoritma penentuan graph terhubung

  1. Algoritma permutasi baris dan kolom matriks
  2. Algoritma memanfaatkan DFT dan BFT
  3. Algoritma menggunakan operasi fusion
  4. Algoritma Warshall

Algoritma Warshall

Nilai 1 pada baris ke i dan kolom ke j menunjukkan keberadaan busur (v_i ,v_j) yaitu lintasan dengan panjang 1 dari v_i ke v_j. Bila elemen-elemen dari A^2 (yaitu A x A) ditunjukkan dengan $a_{ij}^{(2)}$ yaitu

$$a_{ij}^{(2)} = \sum_{k=1}^{2} a_{ik}a_{kj}$$

Untuk nilai k tetap, $a_{ik}a_{kj} = 1$ jika dan hanya jika a_{ik} dan a_{kj} bernilai 1, yaitu terdapat busur (v_i, v_k) dan (v_k, v_j) di graph. Dengan itu, elemen-elemen a_{ij} di baris ke i dan kolom ke j A^n menyatakan keberadaan jalur dengan panjang n. Elemen-elemen a_{ii} di diagonal menyatakan siklus dengan panjang n di simpul ke i.