Decidable & Semidecidable (2)

Kuliah Teori Bahasa dan Automata
Program Studi Ilmu Komputer
Fasilkom UI

Prepared by:
Suryana Setiawan

Review

- Dua bahasa yang sudah kita ketahui kategorinya:
  - **Bahasa** H = {<M, w> : Mesin Turing M halt untuk w}, atau
    problem view: "jika diberikan suatu algoritma (mesin turing M) serta inputnya (string w) apakah algoritma itu **<u>akan</u>** halt untuk input tersebut?"
    ➔ adalah **semidecidable** (SD).

  - **Bahasa** $\neg H$ = {<M, w> : Mesin Turing M tidak halt untuk w}, atau
    problem "jika diberikan suatu algoritma (mesin turing M) serta inputnya (string w) apakah algoritma itu **tidak <u>akan</u>** halt untuk input tersebut?"
    ➔ adalah **non-semidecidabe** ($\neg$ SD).

Bahasa-bahasa D, SD dan ¬SD Lain

The Problem View	***The Language View***
Given a Turing machine M, does M halt on the empty tape?	$H_\varepsilon = \{<M> : \text{TM } M \text{ halts on } \varepsilon\}$
Given a Turing machine M, is there any string on which M halts?	$H_{ANY} = \{<M> : \text{there exists at least one string on which TM } M \text{ halts}\}$
Given a Turing machine M, does M accept all strings?	$A_{ALL} = \{<M> : L(M) = \Sigma^*\}$
Given two Turing machines M_a and M_b, do they accept the same languages?	$\text{EqTMs} = \{<M_a, M_b> : L(M_a) = L(M_b)\}$
Given a Turing machine M, is the language that M accepts regular?	$TM_{REG} = \{<M> : L(M) \text{ is regular}\}$

- Dalam pembahasan selanjutnya kita akan melihat apakah bahasa-bahasa ini D, SD-D atau ¬SD.
- Tidak ada pumping theorem untuk D / SD, untuk melihat suatu bahasa bukan D / SD memerlukan cara lain.

Dua Cara Pembuktian

- Metoda Reduksi,
  - berdasarkan Mapping Reduction (materi kuliah ini)
  - Tidak berdasarkan Mapping Reduction (materi kuliah berikutnya)
- Rice’s Theorem (materi kuliah berikutnya)

Metoda Reduksi

- Berdasar dua strategi:
  - "divide-and-conquer" mereduksi probem L_1 (*known*) menjadi L_2 (*unknown*); dan
  - "proof by contradiction" dengan asumsi X berlaku pada L_2 menyebabkan sifat X berlaku pada L_1 padahal L_1 tidak bersifat X sehingga L_2 juga tidak X.

"Divide and Conquer"

- Problem direduksi menjadi satu atau beberapa problem lainnya yang **lebih sederhana** dan **sudah ada solusinya** sehingga solusi untuk problem semula diperoleh dari solusi problem-problem reduksinya.
- Contoh dalam pembuktian teorema, misalnya kita akan membuktikan $Q(A)$, sementara kita memiliki teorema:

$$\forall x(R(x) and\ S(x) and\ T(x) \rightarrow Q(x))$$

  - Pembuktian $Q(A)$ dapat direduksi menjadi pembuktian $R(A)$, $S(A)$ dan $T(A)$.

Proof By Contradiction w/ **Oracle**

- Oracle adalah mesin turing hipotetis.
  - Arti sebenarnya: paranormal pada jaman Yunani kuno yang selalu bisa menjawab ya/tidak pada setiap pertanyaan.
- *Proof by Contradiction*: dalam reduksi P_1 menjadi P_2, jika P_2 diasumsikan true maka P_1 true, tapi jika ternyata P_1 false maka asumsi tsb. (bhw P_2 true) gugur.
- Bila L_1 dapat direduksi menjadi L_2,
  - Jika L_2 D oleh suatu **Oracle**, maka decidernya dapat digunakan untuk men-*decide* L_1.
  - Tetapi karena L_1 adalah $\neg D$, berarti Oracle tsb tidak ada (alias L_2 adalah $\neg D$)."

Mapping Reducibility

- Salah satu cara reduksi adalah dengan memetakan langsung pada setiap string ke string hasil reduksinya.
- L_1 adalah *mapping reducible* menjadi L_2, ditulis $L_1 \leq_M L_2$, iff terdapat fungsi komputabel f sehingga

$$\forall x \in \Sigma^* \; (x \in L_1 \textbf{ iff } (f(x) \in L_2)$$

- Jadi dengan $x' = f(x)$, pertanyaan “Apakah $x \in L_1$?” dipetakan menjadi “Apakah $x' \in L_2$?”
- Jadi jika $L_1 \leq_M L_2$, dan terdapat *Oracle* yang memutuskan L_2, maka TM C berikut

$$C(x) = Oracle(R(x))$$

juga dapat memutuskan L_1.

Langkah-langkah praktis Membuktikan $L_2 \notin$ D

- Pilih bahasa L_1 untuk reduksi, dengan syarat:
  - Sudah diketahui $L_1 \notin$ D, dan
  - L_1 dapat direduksi menjadi L_2.
- Definisikan algoritma reduksi R dan deskripsikan C sebagai komposisi dari R dengan *Oracle* yang memutuskan L_2.
  - R dapat dimplementasikan sebagai satu atau beberapa TM
- Tunjukkan C *decide* L_1 jika *Oracle* ada dengan menunjukkan:
  - jika $x \in L_1$, maka $C(x)$ menerima, sementara jika $x \notin L_1$, maka $C(x)$ menolak ; C(x) = *Oracle*(R(x)) = *Oracle*(x')
- Kontradiksi: karena $L_1 \neg D$ maka *Oracle* tsb tidak ada (alias L_2 adalah $\neg D$).

H_ε Semidecidable

- H_ε= {<M>: mesin Turing M halt pada ε}
- Problem view: "Apakah suatu algoritma *M* akan halt untuk input ε?"
- Dengan membuat UTM dan mengemulasikan M untuk *w*=ε, jika M halt, maka UTM accept, selanjutnya
  - Untuk setiap <M> ∈ H_ε, UTM akan halt/accept.
  - Untuk setiap <M> ∉ H_ε, UTM tidak akan halt.
- Jadi H_ε adalah SD.

H_ε Tidak Decidable

- *R* adalah mapping reduction dari H ke H_ε, sbb:
  - <*M*,*w*> menjadi <*M#*> mensyaratkan saat *M* halt untuk *w*, maka M# halt untuk ε.
  - terdefinisi sebagai R(<*M*,*w*>) yang membuat dan mereturn <M#>, dengan M# akan melakukan (untuk isi tape x):
    - Menghapus tape.
    - Menulis *w* pada tape.
    - Jalankan *M* pada *w*.
- Jika *Oracle* ada dan men-*decide* H_ε,
  maka C = *Oracle(R*(<*M*,*w*>)) dapat men-*decide* H,
  tapi, ternyata tidak ada yang bisa men-*decide* H,
  maka *Oracle* juga tidak ada, berarti juga H_ε adalah $\neg D$.
- Kesimpulan gabungan: $H_\varepsilon \in (SD - D)$, sebagaimana H.

H_{ANY} Semidecidable

- H_{ANY} = {<*M*> : terdapat sekurangnya satu string yang dapat membuat M halt}
- Membuat enumerator Σ^* dan memeriksa secara dovetailing string-string yang dihasilkan dengan emulasi dari M.
- Jika terdapat satu string menyebabkan halt maka keseluruhan juga halt, ie,
  - untuk <M> $\in H_{ANY}$, emulasi akan halt
  - untuk <M> $\notin H_{ANY}$, emulasi tidak akan halt
- Maka H_{ANY} *SD*.

H_{ANY} Tidak Decidable

- R mapping reduction dari H ke H_{ANY}, sbb:
  - Mapping <*M*,*w*> menjadi <*M#*> mensyaratkan saat *M* halt untuk *w*, maka M# harus halt jika tape yang berisi *w*..
  - terdefinisi sebagai R(<*M*,*w*>) yang membuat dan mereturn <M#>, dengan M# akan melakukan (untuk isi tape x):
    - Periksa *x*
    - Jika *x* = w, jalankan M pada *x*, jika tidak loop.
- Jika *Oracle* ada dan men-*decide* H_{ANY},
  maka C = *Oracle*(*R*(<*M*,*w*>)) dapat men-*decide* H,
  tapi, ternyata tidak ada yang bisa men-*decide* H,
  maka *Oracle* juga tidak ada, berarti juga H_{ANY} adalah $\neg D$.
- Kesimpulan gabungan: $H_{ANY} \in$ (SD – D), seperti halnya H.

Alternatif R untuk H ke H_{ANY}

- Bisa terdapat sejumlah cara reduksi R yang berbeda.
- Suatu kebetulan, R untuk H ke H_{ANY} bisa menggunakan cara seperti untuk reduksi dari H ke H_{ε}
- Tapi tidak selalu demikian!

H_{ALL} Tidak Semidecidable

- H_{ALL} = {<M> : M halt untuk setiap input $x \in \Sigma^*$}
- H_{ALL} adalah $\neg$SD dengan pembuktian reduksi dari $\neg$H (Pembahasan akan dilakukan nanti).

H_{ALL} Tidak Decidable

- R mapping reduction dari H_ε ke H_{ALL}, sbb:
  - Mapping <*M*> menjadi <*M#*> mensyaratkan saat *M* halt, maka M# harus halt apapun isi tapenya.
  - terdefinisi sebagai R(<*M*>) yang membuat dan mereturn <*M#*>, dengan M# akan melakukan (untuk isi tape x):
    - Hapus isi tape
    - Jalankan *M*.
- Jika *Oracle* ada dan men-*decide* H_{ALL},
  maka C = *Oracle*(*R*(<*M*,*w*>)) dapat men-*decide* H_ε,
  tapi, ternyata tidak ada yang bisa men-*decide* H_ε,
  maka *Oracle* juga tidak ada, berarti juga H_{ALL} adalah $\neg$D
- (Bahkan telah disebutkan $\neg$SD pada hal sebelumnya!).

Problem A

- A = {<*M*,*w*> : mesin turing M menerima *w*}
  = {<*M*,*w*> : M mesin Turing dan w $\in L(M)$}
- A berbeda dari H dengan adanya dua kondisi halt: halt-yes (accept) dan halt-no (reject).
- Tanpa R memperhatikan itu, Jika M halt untuk w, maka *Oracle*(*R*(<M,w>)) dapat accept atau reject w.
- Jadi, R harus menyatakan untuk setiap kondisi halt dari M, menjadi halt-yes bagi #M hasil reduksinya.

A Tidak Decidable

- R mapping reduction dari H ke A, sbb:
  - Mapping <*M*,*w*> menjadi <*M*#,w> mensyaratkan saat *M* **halt** untuk *w*, maka M# harus **accept** untuk *w*.
  - terdefinisi sebagai R(<*M*,*w*>) yang membuat dan mereturn <*M*#,w>, dengan M# akan melakukan (untuk isi tape x):
    - Menghapus tape.
    - Menuliskan *w* pada tape.
    - Jalankan *M* pada *w*.
    - Accept.
- Jika *Oracle* ada dan men-*decide A*,
  maka C = *Oracle*(*R*(<*M*,*w*>)) dapat men-*decide* H,
  tapi, ternyata tidak ada yang bisa men-*decide* H,
  maka *Oracle* juga tidak ada, berarti juga A adalah ¬D.

Pertanyaan-pertanyaan Sejenis yang juga $\neg$D

- Berikut ini adalah juga $\neg$D:
  - $A_{\varepsilon} = \{<M> : \text{TM } M \text{ menerima } \varepsilon\}$
  - $A_{ANY} = \{<M> :$ terdapat setidaknya satu string yang diterima TM $M\}$
    $= \{<M> : \text{M mesin Turing dan L(M)} \neq \emptyset\}$
  - $A_{ALL} = \{<M> : M \text{ adalah mesin Turing dan L(M)} = \Sigma^*\}$

- Catatan: untuk $<M> \in A_{ANY}$ dengan $L(M)$ bahasa reguler atau CFL adalah D! Tetapi karena terdapat $<M> \in A_{ANY}$ dengan $L(M)$ bahasa D/SD maka itu menjadi $\neg$D.

EqTMs Tidak Decidable

- EqTMs = {<M_a,M_b> : M_a dan M_b dua mesin Turing dan L(M_a) = L(M_b)}
- R mapping reduction dari RqTMs ke A_{ALL}, sbb:
  - Mapping <*M*> menjadi <*M*,*M#*> mensyaratkan *M#* selalu accept sehingga ketika dibandingkan dengan *M* oleh Oracle dampaknya hanya memerik M saja.
  - terdefinisi sebagai R(<*M*>) yang membuat dan mereturn <*M*,*M#*>, dengan M# akan melakukan (untuk isi tape x):
    - Accept
- Jika *Oracle* ada dan men-*decide* EqTMs,
  maka C = *Oracle*(*R*(<*M*>)) dapat men-*decide* A_{ALL},
  tapi, ternyata tidak ada yang bisa men-*decide* A_{ALL},
  maka *Oracle* juga tidak ada, berarti juga EqTMs adalah ¬D
- (Bahkan EqTMs adalah ¬SD!).