Decidable & Semidecidable: Metoda Reduksi dan Teorema Rice

Kuliah Teori Bahasa dan Automata
Program Studi Ilmu Komputer
Fasilkom UI

Prepared by:
Suryana Setiawan

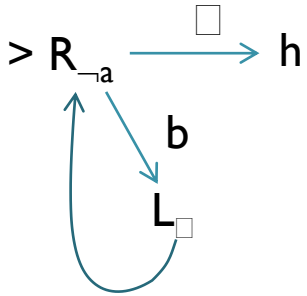
Review

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

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

Contoh

- Mesin M berikut akan halt untuk string-string a*.



- <*M*, ε>, <*M*, a>, <*M*, aa>,… string-string ∈ *H*.
- <*M*, b>, <*M*, ab>, <*M*, ba>, …. string-string ∈ ¬*H*.

Bahasa-bahasa D, SD-D 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?	$EqTMs = \{<M_a, M_b> : L(M_a) = L(M_b)\}$
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 termasuk D, SD-D atau ¬SD.

Dua Cara Pembuktian

- Tidak ada pumping theorem untuk D / SD; cara lain:
  - Metoda Reduksi
    - Berdasarkan **divide-and-conquer** dan **prove-by-contradiction**.
    - Pendekatan khusus yang dibahas **Mapping Reducibility**
  - Rice's Theorem
    - Mengidentifikasi sebagai **fungsi nontrivial** dari definisi masalah yang diberikan sehingga teorema dapat diaplikasikan.

"Divide and Conquer"

- Dalam *Theorem Proving*
  - Dalam pembuktian sifat $Q(A)$, terdapat teorema:

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

    - Pembuktian $Q(A)$ identik dengan pembuktian tiga sifat: $R(A)$, $S(A)$ dan $T(A)$
  - jika $R(A)$ dan $S(A)$ sifat yang true,
    - maka pembuktian sifat $Q(A)$ ditransformasi menjadi pembuktian sifat $T(A)$.
- Dalam reduksi Bahasa L_1 menjadi L_2:
  - Jika terdapat fungsi-fungsi *komputabel*, misalnya f dan g, $L_2 = f(g(L_1))$, memeriksa apakah L_2 *decidable* adalah dengan memeriksa L_1 *decidable*.

Prove by Contradiction

- Dalam pembuktian L_2 adalah *nondecidable* ($\neg$D):
  - mula-mula berasumsi L_2 adalah *decidable*,
  - menyebabkan L_1 juga *decidable*
  - padahal L_1 *nondecidable* (diketahui sebelumnya!)
  - Jadi L_2 juga *nondecidable*.

Mesin Turing Hipotetis **Oracle**

- Arti Oracle:
  - Paranormal di jaman Yunani kuno yang selalu bisa menjawab ya/tidak pada setiap pertanyaan yang diajukan.
  - Mesin yang **selalu** dapat men-*decide* (menjawab Ya/Tidak).
- Bila L_1 dapat direduksi oleh mesin R menjadi L_2, dan seandainya L_2 dapat di-*decide* oleh Oracle,
  - maka komposisi R dan Oracle pada L_2 dapat men-*decide* L_1
    - Oracle(R(L_1)) = Oracle(L_2) = {yes, no}
  - Tdp mesin M yang mendecide L_1, M(L_1) = Oracle(R(L_1)).
- Tetapi karena diketahui L_1 adalah $\neg D$, berarti kontradiksi
  - Oracle tsb **tidak ada** alias $\boldsymbol{L_2}$ **adalah** $\boldsymbol{\neg D}$.

Mapping Reducibility

- Reduksi di tingkat Bahasa sangat sulit dilakukan, yang lebih mudah adalah di tingkat string.
  - **pemetaan** setiap string dari L_1 ke string dari L_2.
- L_1 adalah *mapping reducible* (notasi "$\leq_M$ ") menjadi L_2, ditulis $L_1 \leq_M L_2$ yang didefinisikan:
  - Jika tdp fungsi f komputabel (D) dan
    $\forall x \in \Sigma^*.(x \in L_1)$ **iff** $(f(x) \in L_2)$
  - Dengan $x' = f(x)$, pertanyaan "*Apakah* $x \in L_1$*?*" dipetakan menjadi "*Apakah* $x' \in L_2$*?*"

Oracle pada *Mapping Reducibility*

- Jika $L_1 \leq_M L_2$, dan terdapat *Oracle* yang menjawab "Apakah $x' \in L_2$?",
- Sementara, *R* adalah mesin untuk menkonversi *x* menjadi *x*' atau *x*' = *R*(*x*).
- TM *C* dapat dibangun mengkompisisi *R* dan *Oracle*:

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

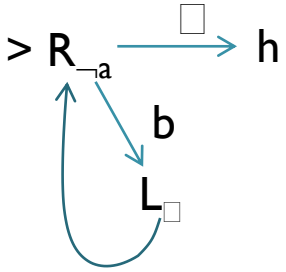
- Maka, *C*(*x*) juga harusnya menjadi mesin Turing yang dapat men-*decide* L_1.
- Selanjutnya L_1 menjadi *decidable*, tapi...??

Langkah-langkah praktis Pembuktian

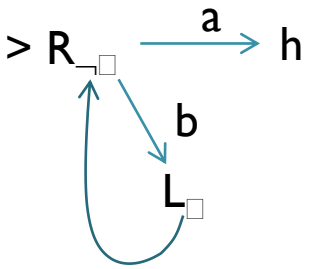
- Diketahui bahasa L_2, pilih bahasa L_1 untuk reduksi, dengan syarat:
  - Sudah diketahui $L_1 \notin$ D, dan
  - Setiap string di L_1 dapat dipetakan menjadi string di L_2.
- Definisikan R sebagai mesin pemetaan (reduksi) tersebut:
  - *R* dapat berupa satu atau beberapa TM.
- Asumsikan oracle untuk L_2 ada, bahwa dan diharapkan $C(x) = Oracle(R(x)) = Oracle(x') \in \{\text{yes, no}\}$.
- *C* men-*decide* L_1 dengan menunjukkan:
  - jika $x \in L_1$, maka $C(x)$ menerima, sementara jika $x \notin L_1$, maka $C(x)$ menolak.
- Kontradiksi: karena L_1 $\neg D$ maka *Oracle* tsb tidak ada (alias L_2 adalah $\neg D$).

H_ε = {<*M*>: mesin Turing *M* halt pada ε}

- Problem view: "Apakah suatu algoritma *M* akan halt untuk input ε?"
- Mesin berikut halt pada ε



- Mesin berikut tidak halt pada ε



H_ε Semidecidable

- Dengan membuat UTM dan mengemulasikan M untuk $w=\varepsilon$, jika M halt, maka UTM accept, selanjutnya
  - Untuk setiap $<M> \in H_\varepsilon$, UTM akan halt/accept.
  - Untuk setiap $<M> \notin H_\varepsilon$, UTM tidak akan halt.
- Jadi H_ε adalah SD.

H_ε Tidak Decidable

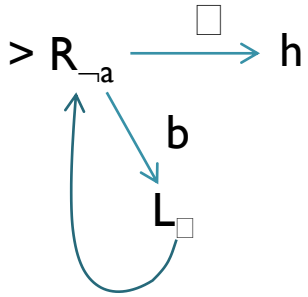
- *R* adalah mapping reduction dari H ke H_ε, sbb:
  - <*M*,*w*> menjadi <*M*#> mensyaratkan saat *M* halt untuk *w*, maka M# halt untuk ε.
  - Fungsi R(<*M*,*w*>) mereturn <M#>, dengan M# akan melakukan (untuk isi tape yang ada *x*):
    - Menghapus *x* dari tape.
    - Menulis *w* pada tape.
    - Jalankan *M* pada *w*.
- Analogi: data w di-hard-coded Bersama source *M* jadi *M*#.
- 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.

Reduksi H ke H_ε

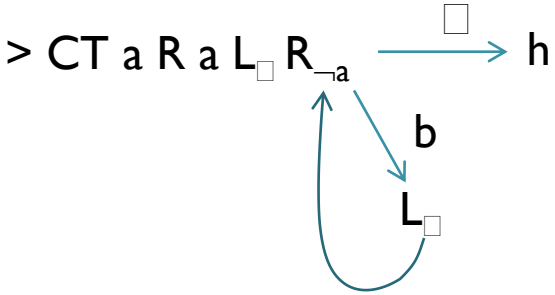
- Mengapa H ?
  - karena belum ada bahasa selain H yang diketahui $\neg D$.
- *M*apping reduction dari H ke H_ε: setiap $\langle M,w \rangle$ "dimodifikasi" oleh R menjadi $\langle M\# \rangle$, sbb:
  - Jika M halt untuk w, maka $M\#$ halt untuk ε
  - Jika M tidak halt untuk w, maka $M\#$ tidak halt untuk ε
- $M\#$ akan mengemulasi M untuk w, sbb.
  - Menghapus tape (sebagai working tape, jika sebelumnya sudah berisi x, agar tidak mengganggu proses berikutnya)
  - Menulis w pada tape (penulisan w sebagai δ dalam $M\#$)
  - Emulasi M pada w (M adalah modul di dalam $M\#$)

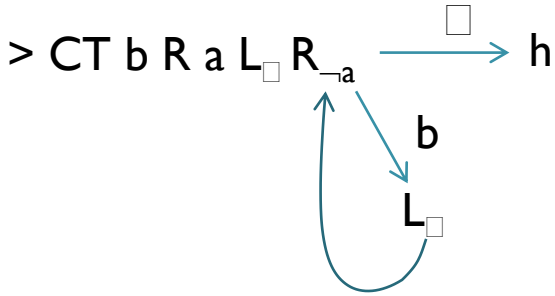
Ilustrasi Reduksi

- Dari contoh *H*, Mesin *M* disamping ini halt untuk a*



- Untuk <*M*, aa> dan <*M*, ba> masing-masing menjadi dua <*M#*> sbb (CT = mesin untuk membersihkan tape).

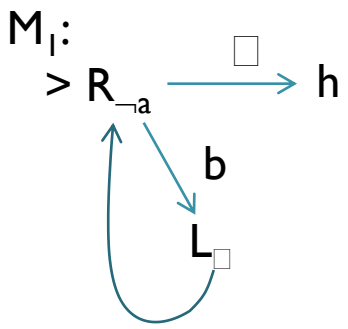


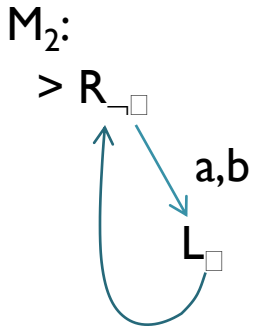


- Jika Oracle bisa mendecide <*M#*> maka kedua mesin *M#* akan halt, berarti pula <*M*, aa> dan <*M*, ba> **decidable**, tapi faktanya *H* **undecidable** maka Oracle tidak ada.

H_{ANY} = {<M> : terdapat sekurangnya satu string yang dapat membuat M halt}

- Problem view: adakah string yang membuat M halt?
- Untuk mesin M_1 disamping ini <M_2> $\in$ H_{ANY} karena ada string yang dapat membuat halt M_1 yaitu a*.
- Sementara mesin M_2 tidak pernah halt (tidak ada string yang dapat membuat M_2 halt), berarti <M_2> $\notin$ H_{ANY}





H_{ANY} Semidecidable

- 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} adalah *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 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, lakukan infinite-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.

Penjelasan Reduksi H ke H_{ANY}

- H dan H_ε dapat digunakan sebagai L_1 tetapi reduksi menjadi H_{ANY} lebih mudah dari H.
- *M*apping reduction:
  - dimana setiap <*M*,*w*> "dimodifikasi" sebagai <*M*#> yang halt untuk any string (dan itu adalah *w* sendiri saat memproses *w*), sbb:
  - Jika *M* halt untuk *w*, maka juga *M*# halt untuk *w*
  - Jika *M* tidak halt untuk *w*, maka *M*# tidak pernah halt untuk apapun.
- Maka *M*# untuk *x*, akan mengemulasi *M* dengan *w* yang sudah hardcoded.
  - jika *x* = *w*, jalankan *M* pada *x* itu, selain itu masuklah *M*# dalam infinite-loop.

$H_{ALL} = \{<M> : M \text{ halt untuk setiap input } x \in \Sigma^*\}$

- H_{ALL} adalah $\neg$SD dengan pembuktian reduksi dari $\neg H$ (Pembahasan akan dilakukan nanti).
- Secara intuitif, mungkinkan menjalankan M untuk setiap kemungkinan string yang ada? [Karena $\in \Sigma^*$ tidak berhingga, kapan pemeriksaan itu dapat selesai?]

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!).

Penjelasan Reduksi H_ε ke H_{ALL}

- Dari kemungkinan-kemungkinan H, H_ε, dan H_{ANY} , maka reduksi paling "dekat" adalah dari dari H_ε.
- *M*apping reduction dari instance H_ε ke instance H_{ALL}, dimana setiap <*M*> "dimodifikasi" sebagai <*M*#>:
  - Jika *M* halt untuk ε, maka juga *M*# halt untuk *x* apa saja
  - Jika *M* tidak halt untuk ε, maka *M*# tidak halt untuk *x* apa saja
- Maka *M*# untuk *x* apa saja akan mengemulasi *M*, sbb.
  - Agat M diperiksa pada tape kosong maka tape dikosongkan (*x* diabaikan saja).
- Selanjutnya <*M*#> siap diperiksa oleh Oracle (u/ *x* apa saja).

Problem A (Accepting)

- Sementara *H* himpunan mesin yang halt, disini *A* himpunan mesin yang accept
  - *A* adalah subset dari *H*, karena ada dua kondisi halt: halt-yes (accept) dan halt-no (reject).
- $A = \{<M,w> : \text{mesin turing } M \text{ menerima } w\}$
  $= \{<M,w> : M \text{ mesin Turing dan w } \in L(M)\}$
- Problem view: apakah mesin *M* akan menerima *w*?
- Tanpa *R* memperhatikan (accept/reject), jika *M* halt untuk *w*, maka jika *Oracle*(*R*(<*M*, *w*>)) menjawab Ya dapat berarti accept atau reject *w*.
- Jadi, *R* harus menyatakan setiap 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 (halt-yes)** untuk *w*.
  - terdefinisi sebagai R(<*M*,*w*>) yang membuat dan mereturn <*M*#,w>, dengan M# akan melakukan (untuk isi tape x):
    - Menghapus tape (memastikan isi tape nantinya digunakan untuk *w*).
    - Menuliskan *w* pada tape.
    - Jalankan *M* pada *w*.
    - Accept (Ini perlu untuk memastikan Oracle nantinya mencapai halt_{yes}).
- 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 ¬D

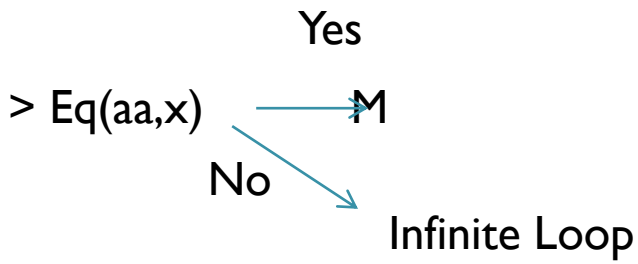
- Berikut ini adalah juga ¬D:
  - $A_{\varepsilon} = \{<M> : \text{TM } M \text{ menerima } \varepsilon\}$
  - $A_{ANY} = \{<M> : \text{terdapat setidaknya satu string yang diterima TM } M\}$
    $= \{<M> : M \text{ mesin Turing dan } L(M) \neq \varnothing\}$
  - $A_{ALL} = \{<M> : M \text{ adalah mesin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

- 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 ¬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 memeriksa 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!).

- Jika terdapat mesin turing Eq(w1,w2) untuk memeriksa apakah w1 sama dengan w1, dari contoh sebelumnya <*M*, aa> dan <*M*, ba> petakan dengan algoritma reduksi tsb maka
  - <*M*, aa> menjadi:



  - <*M*, ba> menjadi: