Decidable & Semidecidable (1)

Kuliah Teori Bahasa dan Automata
Program Studi Ilmu Komputer
Fasilkom UI

Prepared by:
Suryana Setiawan

Mengingat Kembali D dan SD

- TM M **decide** L, **iff**: $\forall w \in \Sigma^*$:
  - Jika $w \in L$ maka M menerima w (halt di h_y), dan
  - Jika $w \notin L$ maka M **menolak** w (halt di h_n).
- L **decidable** ($L \in$ D) **iff** terdapat TM M yang decide L.

- TM **semidecide** L, **iff**: $\forall w \in \Sigma^*$:
  - Jika $w \in L$ maka M menerima w (halt di h_y), dan
  - Jika $w \notin L$ maka M **tidak menerima** w (yaitu, halt di h_n atau tidak halt sama sekali).
- L **semidecidable** ($L \in$ SD) **iff** terdapat TM M yang **semidecide** L.

CFL Proper Subset dari D

- Setiap bahasa CF adalah Decidable karena setiap PDA dapat disimulasikan oleh TM yang selalu halt.
- Terdapat sejumlah bahasa Decidable tetapi bukan context-free. Misalnya:
  - $A^nB^nC^n$
  - WcW
  - WW

Halting Problem

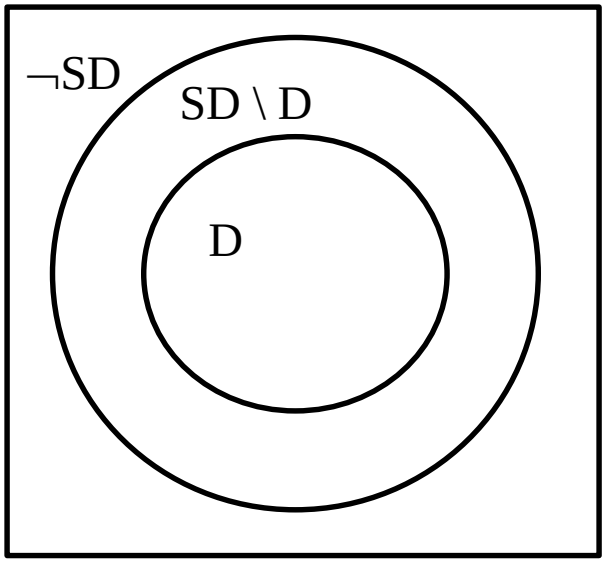
- H ={<M,w>: Mesin Turing M halt pada string input w}
  - Jika <M,w> $\in$ H, saat UTM mengemulasi M untuk w akan juga halt (karena M halt untuk w), tetapi jika <M,w> $\notin$ L_1 tidak UTM halt (karena M tidak halt untuk w).
- Berarti UTM **semidecide** H, dan selanjutnya berarti H adalah **semidecidable**.

D Proper Subset dari SD

- Setiap bahasa D adalah juga bahasa SD.
- Terdapat sejumlah bahasa SD yang bukan D.
  - Contohnya adalah *H*.
  - Contoh lain, *L* = {w: w di dari seseorang yang akan merespon status facebook terbaru anda}
    - Jika w merespon anda akan menerima, jika anda belum menerimanya, w mungkin merespon, mungkin juga tidak!

Hirarki Bahasa

- D adalah bahasa-bahasa decidable yang berada dalam lingkaran terdalam.
- SD adalah bahasa-bahasa semidecidable yang berada dalam lingkaran lebih besar.
- SD \ D (atau SD – D) adalah area di luar D tapi di dalam SD yaitu semidecidable.
  - Minimal ada satu bahasa, yaitu H.



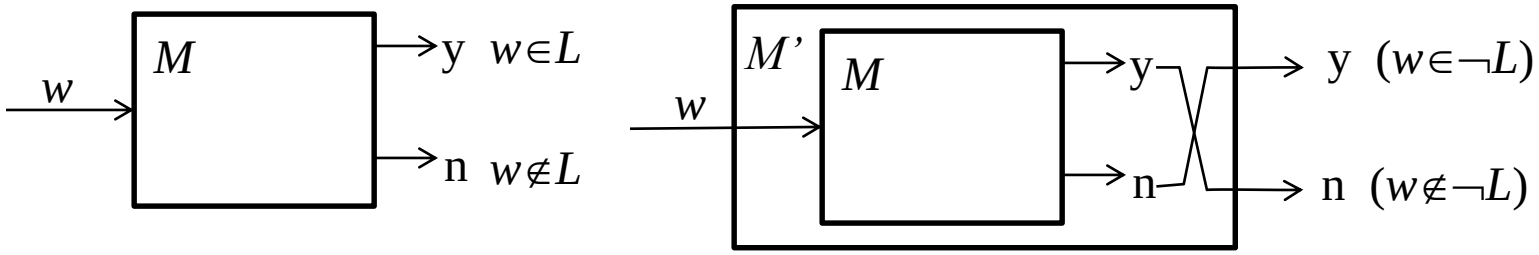
- ¬SD adalah area di luar SD yang berisi bahasa-bahasa yang tidak semidecidable.
  - Minimal ada satu bahasa (akan dibahas dl slide berikutnya)

Adakah bahasa di luar SD?

- Semua kemungkinan mesin turing adalah *infinite*, tetapi karena spesifikasinya menggunakan entitas-entitas yang finite, setiap mesin turing dapat dienumerasi mulai dari yang berjumlah status 1, 2, …dst. Maka himp. semua mesin turing adalah *countably infinite*
- Karena setiap mesin turing terkait dengan bahasa SD tertentu maka **SD adalah *countably infinite*** pula.
- Namun, karena setiap bahasa dalam Σ adalah subset dari Σ^*, sementara Σ^* adalah *infinite*, maka himpunan seluruh **bahasa dalam Σ adalah *uncountable infinite***.
- **Karena *countably infinite* << *uncountably infinite*, maka banyaknya bahasa ¬SD juga *finite*.**

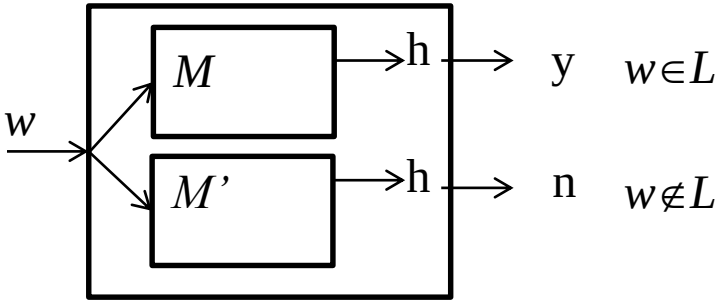
D Closure pada Komplemen

- Ingat bahwa: bahasa reguler bersifat clusure pada operasi komplemen sementara CFL tidak closure.
- Bahasa-bahasa D **closure** pada operasi komplemen.
  - Jika $L \in$ D, terdapat suatu TM M yang akan selalu halt di h_y untuk setiap $w \in$ L dan halt di h_n untuk $w \notin$ L.
  - M' dapat dibuat untuk $\neg L = \{w : w \notin L\}$ dari M dengan men-swap y dan n sehinga M' selalu halt.



SD Tidak Closure pada Komplemen

- Dengan kontradiksi:
  - jika setiap bahasa L dalam SD memiliki $\neg L$ yang juga SD, maka terdapat M yang semideciding L berarti terdapat M' yang semideciding $\neg L$
  - maka dapat dibuat suatu TM $M^{\#}$ yang akan decides L dengan cara simulasi paralel M dan M':



    - Saat M halt untuk $w \in L$, maka $M^{\#}$ juga halt di y, dan saat M' halt untuk $w \in \neg L$ ($w \notin L$), maka $M^{\#}$ juga halt di n.
  - Berarti, jika benar closed maka setiap L dalam SD adalah D.
- Tetapi karena ada bahasa dalam SD yang tidak D (yaitu H), maka berarti tidak closure (pada komplemen).

L sekaligus SD dan D

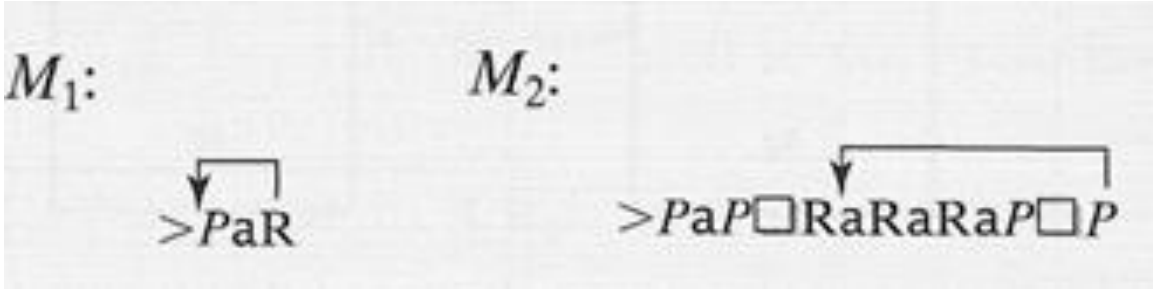
- Bilamana "L dan $\neg L$ keduanya SD", berarti juga "L dan $\neg L$ keduanya D".
- Penjelasannya adalah seperti untuk pembuktian closure pada komplemen.

$\neg H$ bukan SD

- H ={<M,w>: Mesin Turing M halt pada string input w} dan
  $\neg H$ ={<M,w>: Mesin Turing M tidak halt pada string input w}
- $\neg H$ merupakan bahasa yang bukan SD.
  - Dengan kontradiksi, jika $\neg H$ adalah juga SD sebagaimana sebelumnya berimplikasi pada H dan $\neg H$ adalah juga D. Padahal sudah diketahui sebelumnya bahwa H adalah SD – D !

Mesin Enumerator

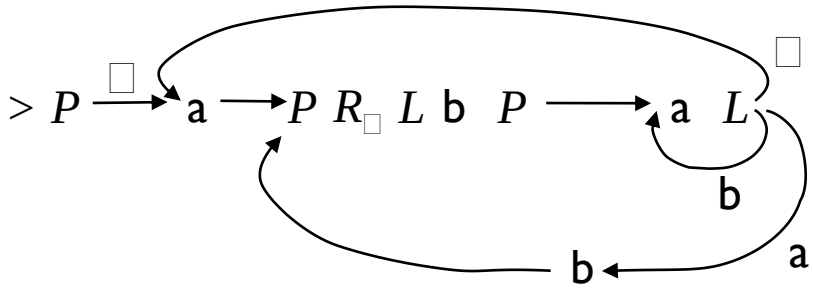
- Mesin Enumerator bahasa *L* men-generate setiap string anggota dari bahasa *L* tsb. Bisa secara proper-order atau secara acak.
- Mesin Turing enumerator dapat didefinisikan dengan adanya notasi makro *P* untuk "mengirimkan/mencetak" isi tape yang telah berisi string. Contoh:



- Jika *L finite* maka mesin suatu saat harus halt, jika *L infinite* maka *M* akan beriterasi tanpa pernah halt.

Generator String secara Properorder

- Mesin enumerator Σ^* secara proper order dapat dengan mudah dibuat, misalnya untuk Σ = {a,b}:



- Mesin enumerator Σ^* dapat digunakan sebagai string generator (SG) untuk enumerator bahasa-bahasa lain

String Enumerator WW menggunakan String Generator

- $WW = \{ww : w \in \Sigma^*\}$
- Enumerator M untuk WW sbb.
  1. Mesin M memanggil SG
  2. setiap saat *P* dijalankan oleh SG, string yang dihasilkan dalam tape diperiksa oleh TM M_{ww} (mesin yang mengenali WW),
     saat M_{ww} halt di h_y maka M menjalankan *P*.
  3. Ulangi mulai langkah 1.

Dovetailing

- Dapatkah SG digunakan dlm enumerator H?
- Tidak, jika caranya seperti untuk WW.
  - Misalkan w_1 mendahului w_2 dalam urutan properorder oleh enumerator Σ^*, jika $w_1 \notin$ H dan $w_2 \in$ H, maka ada kemungkinan saat w_1 diperiksa, M tidak akan pernah memeriksa w_2 karena saat pemeriksaan w_1 dapat terjadi infinite-loop.
- Enumerator dapat dilakukan dengan teknik **devotailing** (pemeriksaan dijalankan step-by-step setiap kali string w_i dihasilkan SG, $i = 1, 2, \ldots$) sbb:
  - Setelah start, lalu SG menghasilkan w_1, lalu w_1 diperiksa dalam langkah pertama
  - Saat SG menghasilkan w_2, kemudian w_1 diperiksa dalam langkah kedua, dan w_1 diperiksa dalam langkah pertama.
  - Saat SG menghasilkan w_3 kemudian w_1 diperiksa dalam langkah ketiga, w_1 langkah kedua, dan w_3 langkah pertama.
  - Dst.
- Urutan string bisa acak karena sesuai dengan urutan halt.

$\varepsilon^{(1)}$										
$\varepsilon^{(2)}$	$a^{(1)}$									
$\varepsilon^{(3)}$	$a^{(2)}$	$b^{(1)}$								
$\varepsilon^{(4)}$	$a^{(3)}$	$b^{(2)}$	$aa^{(1)}$							
$\varepsilon^{(5)}$	$a^{(4)}$	$b^{(3)}$	$aa^{(2)}$	$ab^{(1)}$						
$\varepsilon^{(6)}$	$a^{(5)}$	$b^{(4)}$	$aa^{(3)}$	$ab^{(2)}$	$ba^{(1)}$					
$\varepsilon^{(8)}$	$a^{(6)}$	$b^{(5)}$	$aa^{(4)}$	$ab^{(3)}$	$ba^{(2)}$	$bb^{(1)}$				
$\varepsilon^{(8)}$	$a^{(8)}$	$b^{(6)}$	$aa^{(5)}$	$ab^{(4)}$	$ba^{(3)}$	$bb^{(2)}$	$aaa^{(1)}$			
$\varepsilon^{(10)}$	$a^{(8)}$	$b^{(8)}$	$aa^{(6)}$	$ab^{(5)}$	$ba^{(4)}$	$bb^{(3)}$	$aaa^{(2)}$	$aab^{(1)}$		
$\varepsilon^{(11)}$	$a^{(10)}$	$b^{(8)}$	$aa^{(8)}$	$ab^{(6)}$	$ba^{(5)}$	$bb^{(4)}$	$aaa^{(3)}$	$aab^{(2)}$	$aba^{(1)}$	
---	---	---	---	---	---	--	---	---	---	---

Turing Enumerable

- Bahasa *L* disebut **turing enumerable** jika dapat dienumerasikan (baik secara proper-order maupun acak)
- Bahasa *L* disebut **turing enumerable secara proper-order** jika dapat dienumerasikan secara proper-order
  - Suatu bahasa *L* adalah SD **iff** L turing enumerable.
  - Suatu bahasa *L* adalah D **iff** L turing enumerabel secara proper-order