Decidable & Semidecidable

Kuliah Teori Bahasa dan Automata
Program Studi Ilmu Komputer
Fasilkom UI

Prepared by:
Suryana Setiawan

Bahasa-Bahasa Decidables (D)

- 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).
- Untuk versi TM yang halt (menerima) atau crash (menolak). TM M **decide** L, **iff**: $\forall w \in \Sigma^*$:
  - Jika $w \in L$ maka M **menerima** w (halt), dan
  - Jika $w \notin L$ maka M **menolak** w (crash).

- L **decidable** ($L \in$ D) **iff** terdapat TM M yang decide L (atau L disebut suatu *decidable language*).
  - Istilah lain: Recursive Languages.

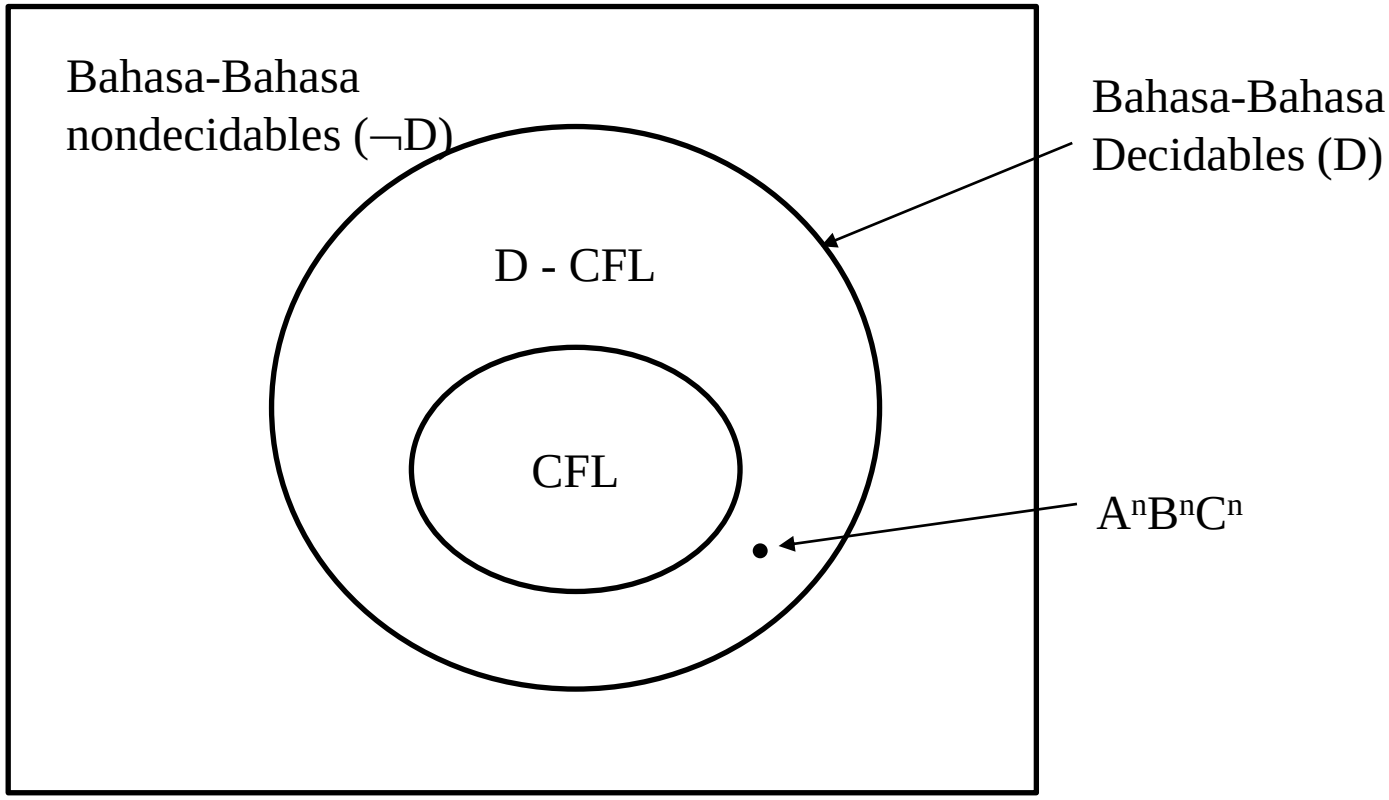
CFL Proper Subset dari D

- Setiap bahasa CF adalah Decidable karena setiap PDA dapat disimulasikan oleh TM yang selalu halt.
- Terdapat sejumlah bahasa Decidable tetapi bukan context-free. Misalnya:
  - $A^nB^nC^n$, WW, WcW, dll.
- Apakah $\{1^n : n$ bilangan prima$\}$ decidable? Bilangan prima terbesar yang diketahui (per januari 2020) adalah $2^{82,589,933} - 1$, sebuah bilangan dengan 24,862,048 deretan dijit desimal.
  - Algoritma pencarian bilangan prima (sieve, dll) secara lojik sudah benar (decide), hanya waktu komputasi yang sangat lama (tahunan per bilangan prima).

CFL, D, ¬D (nondecidables)

Reguler ⊂ CFL ⊂ D

D – CFL tidak kosong, contohnya $A^nB^nC^n$



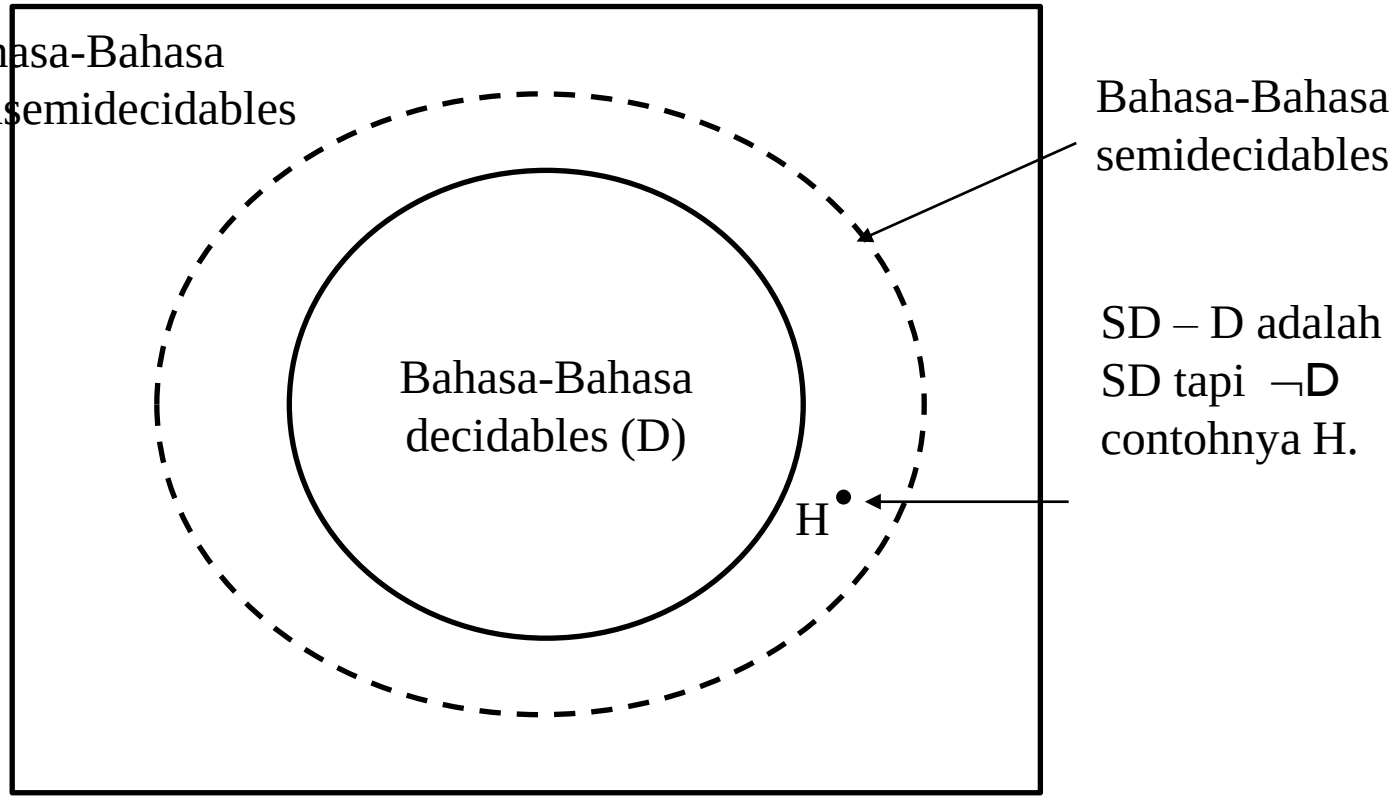
Bahasa-Bahasa Semidecidables (SD)

- 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, crash, atau tidak halt sama sekali).
- L **semidecidable** ($L \in$ SD) **iff** terdapat TM M yang **semidecide** L.
  - Nama lainnya: recursively enumerable languages.
- Bahasa-Bahasa D adalah juga SD karena sesuai definisi tsb.
- Tapi tidak setiap Bahasa SD adalah D karena ada Bahasa SD yang menyebabkan mesin turing tidak halt.

D, SD – D, ¬SD (nonsemidecidables)

$D \subset SD \subset 2^{\Sigma^*}$ (Universe)

$\neg D = \neg SD \cup (SD-D)$

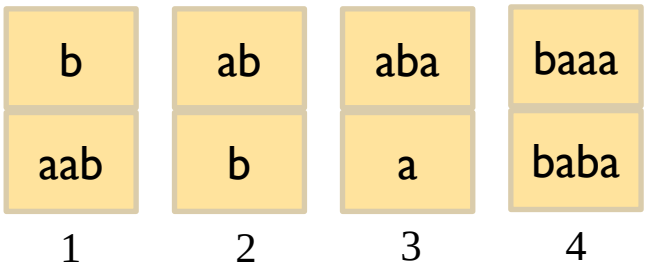


Halting Problem

- 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_1UTM tidak halt (karena M tidak halt untuk w).
- Berarti UTM **semidecide** H, dan selanjutnya berarti H adalah **semidecidable**.
- Contoh:
  - dalam tugas terkumpul m mesin: M_1, M_2, …M_m.
  - Grader mengujinya dengan n string: w_1, w_2, …, w_n.
  - Terdapat pasangan mxn kemungkinan <M_i, w_j>.
  - Di antaranya ada pasangan
    - yang halt (crash dikategorikan halt juga yaitu h_{no}), berarti anggota H, dan
    - yang tidak halt (infinite loop), berarti bukan anggota H.

Intermezzo: PCP-Games

- PCP No 1:

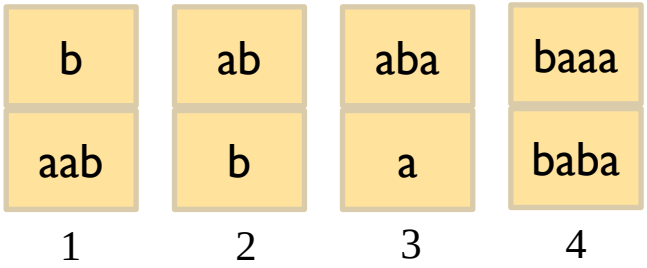


- Terdapat empat macam kartu “domino PCP”.
- Pada setiap hartu tertera string-string spt pd gambar.
- Dapatkan susunan horizontal tanpa terbalik tapi boleh berulang, supaya di bagian atas terbentuk string yang sama dengan yang terbentuk bagian bawah.

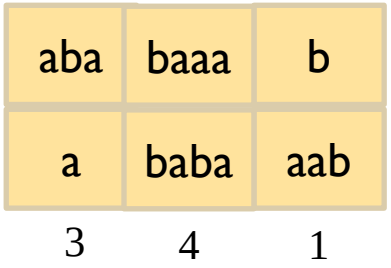
Tekan spasi untuk lanjut

Intermezzo: PCP-Games

- PCP No 1:



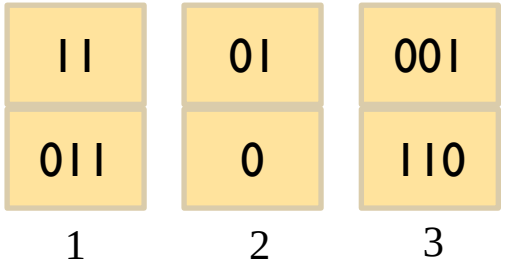
- Salah satu solusi deretan 3,4,1:



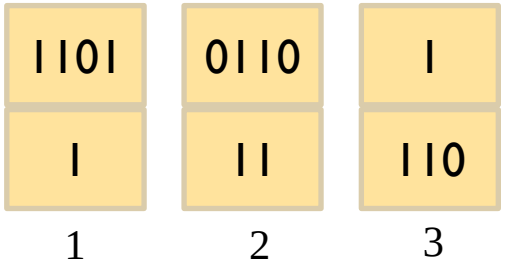
- Solusi lain deretan 3,4,1,3,4,1,…3,4,1

Intermezzo: masih PCP Games

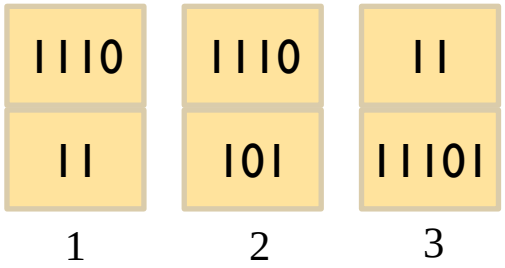
- PCP no 2:



- PCP no 3:



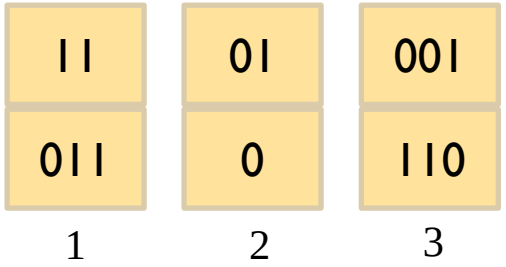
- PCP no 4:



Tekan spasi untuk lanjut

Intermezzo: masih PCP Games

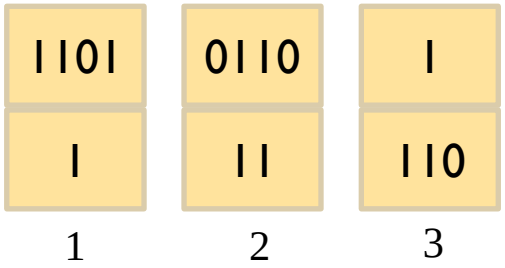
- PCP no 2:



- Maaf, PCP ini **tidak memiliki solusi**.
- Penjelasan:
  - Dari root solusi tidak mungkin dimulai dari 1 atau 3 karena berbeda digit pertama; jadi solusi harus dimulai dari 2.
  - Setelah itu bagian bawah kartu berikutnya harus dimulai digit 1 untuk menyamakan dengan atas.
  - Tetapi, tidak ada kemungkinan yang bisa dipilih

Intermezzo: masih PCP Games

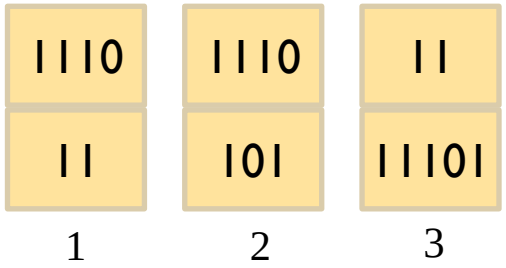
- PCP no 3:



- Bisa mendapatkannya? Silakan mencobanya.
- Mungkin anda berpikir tidak ada solusinya.
- Menurut referensi kita, ada tapi sangat Panjang: 252 kartu (perlu diverifikasi dengan program).
- Bahkan ada solusi lain yang lebih panjang lagi!

Intermezzo: masih PCP Games

- PCP no 4:



- Setelah mencoba sekian panjangnya belum juga bisa disebutkan ada atau tidak ada solusinya.

Post Correspondence Problem (PCP)

- Diberikan dua deret string dengan panjang yang sama: X[1], X[2], …, X[*n*], dan Y[1], Y[2], …, Y[*n*].
- Adakah solusinya yang berupa deret sejumlah bilangan $i_1, i_2, \ldots, i_m$, dengan $i_j \in \{1,..,n\}$, sehingga terjadi kesamaan string hasil konkatanasinya, $X[i_1]X[i_2]\ldots X[i_m] = Y[i_1]Y[i_2]\ldots Y[i_m]$?

Algoritma Pencarian Solusi PCP

- Jika solusi ada, solusi dapat diperoleh dengan pembentukan tree pencariaan dari suatu root node, bercabang ke semua kemungkinan kartu, hingga menemukan cabang yang memenuhi kriteria kesamaan string itu. Sbb.:
  - Cabang-cabang yang mungkin dari root adalah setiap i yang mana $X[i_1]$ prefiks dari $Y[i_1]$ atau sebaliknya $Y[i_1]$ prefiks dari $Y[i_1]$.
  - Setiap Cabang i_1 bercabang lagi dengan ke i_2 jika terdapat i_2 sehingga $X[i_1]X[i_2]$ prefiks dari $Y[i_1]Y[i_2]$ atau sebaliknya $Y[i_1]Y[i_2]$ prefiks dari $X[i_1]X[i_2]$.
  - Dan seterusnya, dari i_2 bercabang ke i_3 jika $X[i_1]X[i_2]X[i_3]$ prefiks dari $Y[i_1]Y[i_2]Y[i_3]$ atau sebaliknya.
  - Jawaban pertanyaan adalah YA jika pada ahirnya terdapat $X[i_1]X[i_2]\ldots X[i_m] = Y[i_1]Y[i_2]\ldots Y[i_m]$.

PCP Sebagai Masalah Keputusan Bahasa

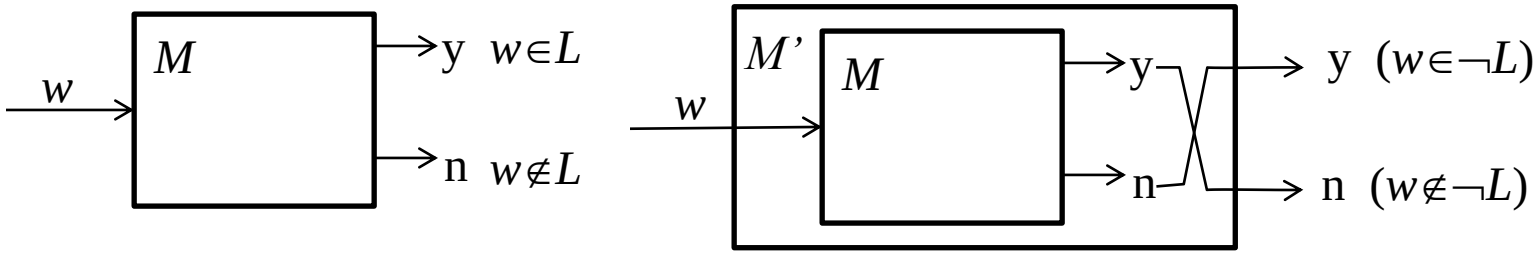
- PCP sebagai masalah keputusan Bahasa:
  - Dengan suatu alfabet Σ, string <*P*> didefinisikan <*P*> = $((x_1,x_2,\ldots,x_n),(y_1,y_2,\ldots,y_n))$, dimana $\forall j\ (x_j,\ y_j \in \Sigma^+)$.
  - Solusi dari PCP *P* ini adalah suatu deret *m* bilangan-bilangan bulat $i_1, i_2, \ldots, i_m$: $\forall 1 \leq j \leq m,$ $\left(1 \leq i_j \leq n \text{ dan } x_{i_1} x_{i_2} \ldots x_{i_m} = y_{i_1} y_{i_2} \ldots y_{i_m}\right)$
- PCP = {<*P*>: *P* suatu instan dari PCP dan *P* memiliki solusi}.
  - Soal 1 dan 3 adalah anggota PCP (decided)
  - Soal 2 adalah bukan PCP (decided)
  - Soal 4 adalah bukan PCP (undecided)

PCP adalah SD-D

- Dengan contoh-contoh di atas secara intuitif problem PCP ini adalah (*SD* – *D*).
  - Contoh 4 menunjukkan adanya kemungkinan pencarian yang tidak pernah selesai ➔ $\neg D$
- Pembuktian formal sebagai SD
  - dijelaskan dengan string generator secara proper-order yang menghasilkan semua kemungkinan deret dengan panjang mulai dari k=1, 2, … Untuk $<P> \in$ PCP, suatu ketika akan ketemu solusi pada harga k tertentu.
- Pembuktian formal sebagai $\neg D$
  - ditunjukkan dengan reduksi dari *H*.
  - Disini pembuktiannya tidak diuraikan (Silakan mengacu pada referensi yang diberikan).

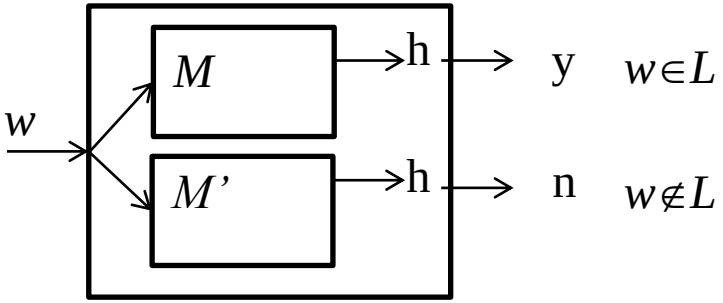
D Closure pada Komplemen

- 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).

Jika *L* sekaligus SD dan D

- Bilamana "*L* dan ¬*L* keduanya SD", berarti juga "*L* dan ¬*L* keduanya D".
  - Jika *L* dan ¬*L* adalah SD maka *L* adalah D.
  - Jika *L* adalah D, maka ¬*L* adalah D.
- Jika *L* adalah SD – D maka ¬*L* bukan SD.

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 setidaknya ada language yang ¬SD .**

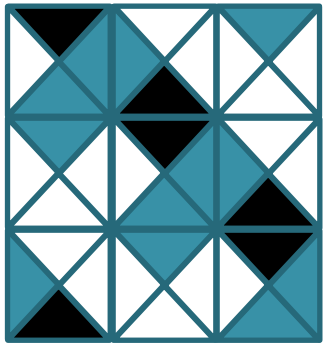
$\neg H$ bukan SD

- 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 !

Problem Tiling (Dengan Solusi)

- Diberikan tile (ubin) dalam jumlah tak berhingga namun hanya terdiri dari beberapa pola saja, contohnya:

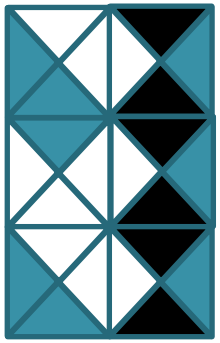
- Susunlah ubin-ubin menutupi lantai tanpa boleh dirotasi, tanpa membentuk gap/pergeseran, dan setiap sisi bersebelahan harus berwarna sama.
- Hasilnya



Problem Tiling (Tanpa Solusi)

- Bagaimana jika pola-polanya seperti ini?

- Hanya dapat menutupi lantai secara terbatas sbb.



Definisi Tiling Sebagai Problem Bahasa

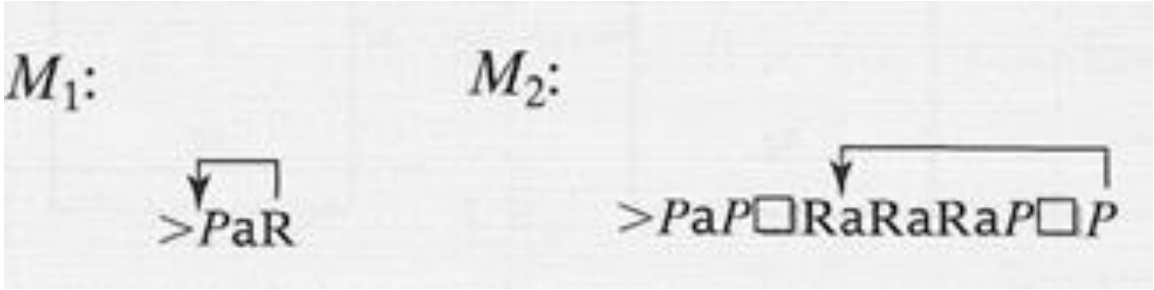
- Pola-pola tile didefinisikan sebagai string kode warna terurut clock-wise dari yang teratas.
  - pada contoh pertama T_1 = {GWWW, WWBG, BGGW}
  - Pada contoh kedua T_2 = {GWWW, WWGG, BGBW}
- TILES = {<T>: setiap bidang dapat ditutup oleh tile-tile dari tile set T sesuai dengan aturan penyusunan di atas}.
- Apakah problem TILES decidable?
- Dapat ditunjukkan bahwa
  - TILES adalah problem $\neg SD$,
  - $\neg TILES$ adalah problem $(SD - D)$.

Catatan mengenai Bahasa-Bahasa ¬SD

- Beberapa hal dari pembahasan sebelumnya:
  - Bahasa H adalah SD-D yang memiliki komplemen ¬H yang merupakan ¬SD.
  - Bahasa PCP adalah SD-D yang memiliki komplemen ¬PCP yang merupakan ¬SD.
  - Bahasa Tiles adalah ¬SD. Yang memiliki komplemen ¬Tiles yang merupakan SD.
- Jangan salah menyimpulkan:
  - Terdapat banyak Bahasa ¬SD yang memiliki komplemen juga ¬SD.
- Yang pasti adalah jika : Bahasa SD-D maka komplemennya ¬SD.

Mesin Enumerator

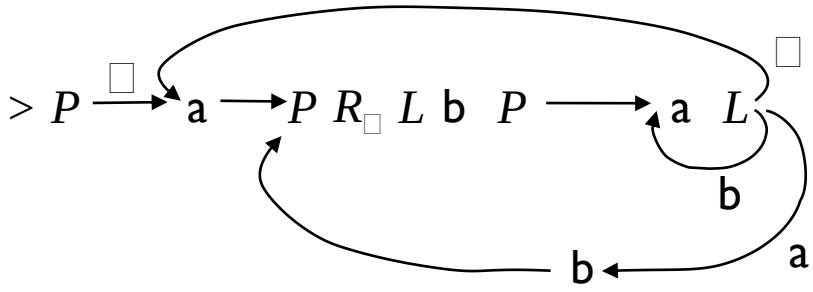
- Mesin Enumerator bahasa *L* men-generate setiap string anggota dari bahasa *L* tsb. Bisa secara proper-order atau secara acak.
- Mesin Turing enumerator dapat didefinisikan dengan adanya notasi makro *P* untuk "mengirimkan/mencetak" isi tape yang telah berisi string. Contoh:



- 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 $\Sigma = \{a,b\}$:



- 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.

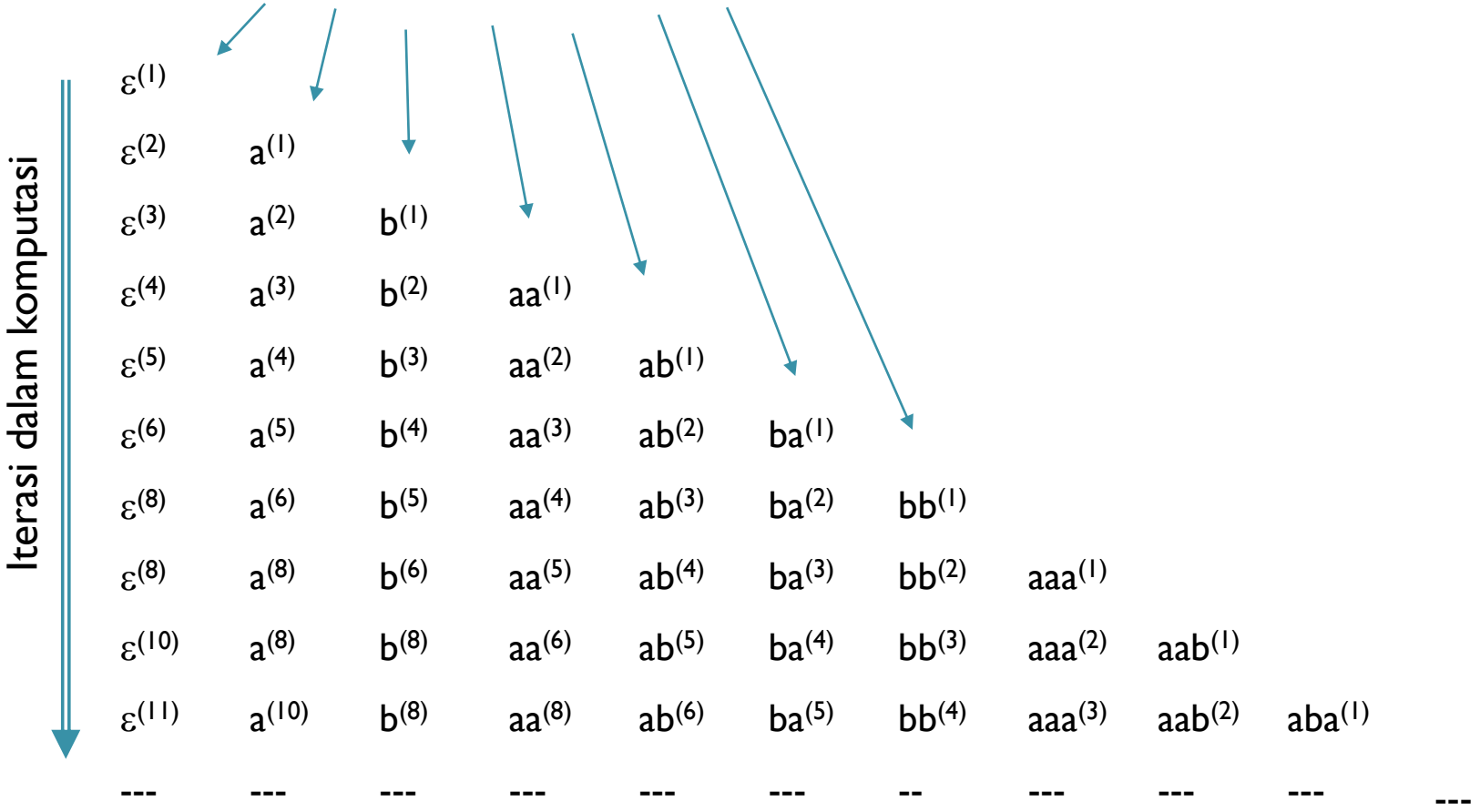
Dovetailing

- 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 **dovetailing** (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.

Ilustrasi Dovetailing

String Generator menghasilkan:

ε, a, b, aa, ab, ba, bb, … dst



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

Butir-butir Penting!

- Bahasa yang dapat di-*decide* mesin-mesin Turing disebut **Decidable** dan bersifat **Turing Enumerable secara *proper-order***, serta closure terhadap komplemen.
- Bahasa yang di-*decide* mesin-mesin Turing khususnya untuk anggota bahasa saja disebut **Semidecidable**, dan bersifat **Turing Enumerable**, serta tidak closure terhadap komplemen.
- Bahasa yang secara umum tidak Turing enumerable disebut **Nonsemidecidable,** dan ada dua sub-class:
  - Yang memiliki komplemen Semidecidable.
  - Yang memiliki komplemen Nonsemidecidable.