IF5110 Teori Komputasi

4. Undecidabality (Bagian 2)

Oleh: Rinaldi Munir

Program Studi Magister Informatika STEI-ITB

Mengenumerasi String Biner

- String biner dapat dipandang sebagai integer.
- Jika *w* adalah string biner, maka 1*w* adalah integer biner ke-*i*, dilambangkan dengan w_i.
- Contoh:

  - ε adalah string biner ke-1 atau w_1 (karena $1\varepsilon = 1$)
  - 0 adalah string biner ke-2 atau w_2 (karena 10 = 2)
  - 1 adalah string biner ke-3 atau w_3 (karena 11 = 3)
  - 00 adalah string biner ke-4 atau w_4 (karena 100 = 4)
  - 01 adalah string biner ke-5 atau w_5 (karena 101 = 5)
  - 101 adalah string biner ke-13 atau w_{13} (karena 1101 = 13)
  - 0101 adalah string biner ke-21 atau w_{21} (karena 10101 = 21)

- String biner apakah w_{37}?

  Jawaban: 37 = 100101

  Dengan membuang 1 di depan,

  maka w_{37} = 00101

- String biner apakah w_{100}?

- Pengurutan kanonik semua string biner:

  - { ε, 0, 1, 00, 01, 10, 11, 000, 100, 101, 110, … }

  - {w_1, w_2, w_3, w_4, …, w_i, … }

Pengkodean Mesin Turing

- Ingatlah kembali bahwa mesin Turing dapat dikodekan menjadi string biner.

  Review: Pembentukan kode yang menggambarkan karakteristik suatu mesin Turing $T = (Q, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$ filakukan dengan cara:

  (a) Simbol-simbol 0, 1, dan B dilambangkan berturut-turut sebagai simbol X_1, X_2, dan X_3. Simbol-simbol lainnya dapat dikodekan dengan X_4, X_5, dan seterusnya.

  (b) Arah gerakan L dan R dilambangkan sebagai simbol D_1 dan D_2.

(c) Setiap gerakan mesin Turing T, $\delta(q_i, X_j) = (q_k, X_l, D_m)$, dapat dituliskan sebagai 5-t*upl*e (i, j, k, l, m) yang dikodekan sebagai string biner $C = 0^i10^j10^k10^l10^m$

(d) Jika mesin Turing T memiliki sebanyak r gerakan, maka seluruh pergerakan yang ada dapat dikodekan sebagai:

$$C_1 11 C_2 11 \ldots 11 C_r$$

- **Contoh**: Misalkan terdapat mesin Turing yang memiliki gerakan seperti pada tabel berikut:

	0	1	B
q_1		$(q_2, 0, R)$	
q_2	$(q_3, 1, L)$	$(q_2, 1, R)$	$(q_3, 1, L)$
q_3	$(q_4, 0, R)$	$(q_3, 1, R)$	$(q_4, 0, R)$
q_4			

- Pengkodean setiap gerakan ditunjukkan pada tabel berikut:

Gerakan	Kode
$\delta(q_1, 1) = (q_2, 0, R)$	0 1 00 1 00 1 0 1 00
$\delta(q_2, 0) = (q_3, 1, L)$	00 1 0 1 000 1 00 1 0
$\delta(q_2, 1) = (q_2, 1, R)$	00 1 00 1 00 1 00 1 00
$\delta(q_2, B) = (q_3, 1, L)$	00 1 000 1 000 1 00 1 0
$\delta(q_3, 0) = (q_4, 0, R)$	000 1 0 1 0000 1 0 1 00
$\delta(q_3, 1) = (q_3, 1, L)$	000 1 00 1 000 1 00 1 0
$\delta(q_3, B) = (q_4, 0, L)$	000 1 000 1 0000 1 0 1 00

- Kode mesin Turing dapat dituliskan sebagai:

  0100100101001**1** 0010100010010**1**100100100100**1**1

  0010001000100**1**011 0001010000101001**1**00010010001001

  0**1**1000100010000010100

- Perhatikan bahwa bentuk kode mesin Turing merepresentasikan sebuah *integer*.

  Contoh:

  01001001010011 0010100010010

  integer

- Sehingga kita dapat mengatakan Mesin Turing ke-*i* adalah mesin Turing *M* yang kodenya adalah w_i.

- Mesin Turing ke-*i* dilambangkan dengan M_i.

- Pengurutan kanonik mesin Turing:

  $\{M_1, M_2, M_3, M_4, \ldots, M_i, \ldots\}$

- Namun banyak *integer* yang tidak berkoresponden dengan mesin Turing.

  Contoh: 11001 → bukan kode mesin Turing, karena tidak dimulai dengan 0

  001011110100101 00 → tidak valid karena memiliki 111

- Jika w_i bukan kode mesin Turing yang valid, maka kita katakan M_i adalah mesin Turing dengan satu status tetapi tidak memiliki transisi (*no move*).

- Jadi, untuk nilai-nilai *i* ini, M_i aadalah mesin Turing yang segera berhenti apapun input yang diberikan. Jadi, $L(M_i) = \varnothing$ jika w_i gagal menjadi kode mesin Turing yang valid.

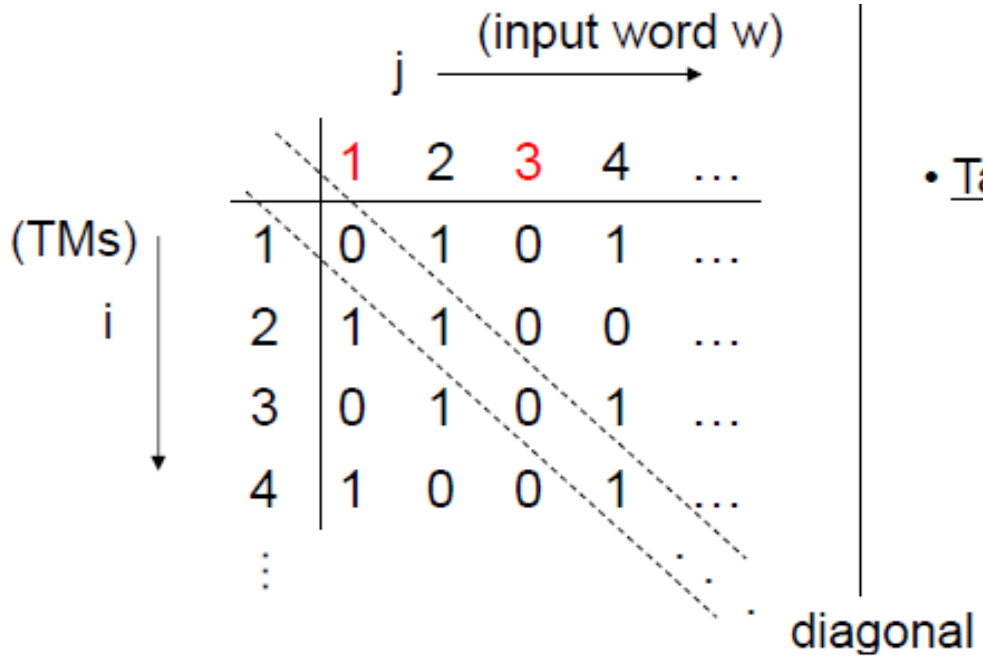
Bahasa Diagonalisasi

- Bahasa diagonalisasi (L_d) adalah himpunan string w_i sedemikian sehingga w_i bukan elemen $L(M_i)$.

  $L_d = \{ w_i \mid w_i \notin L(M_i)\}$

- Dengan kata lain, bahasa diagonalisasi adalah bahasa yang berisi semua string yang mana mesin Turing yang berkoresponden tidak menerima dirinya sendiri (kode mesin Turing itu sendiri).

- Jadi, L_d terdiri dari semua string *w* sedemikian sehingga mesin Turing *M* yang kodenya *w* tidak menerima *w* bila diberikan *w* sebagai input.



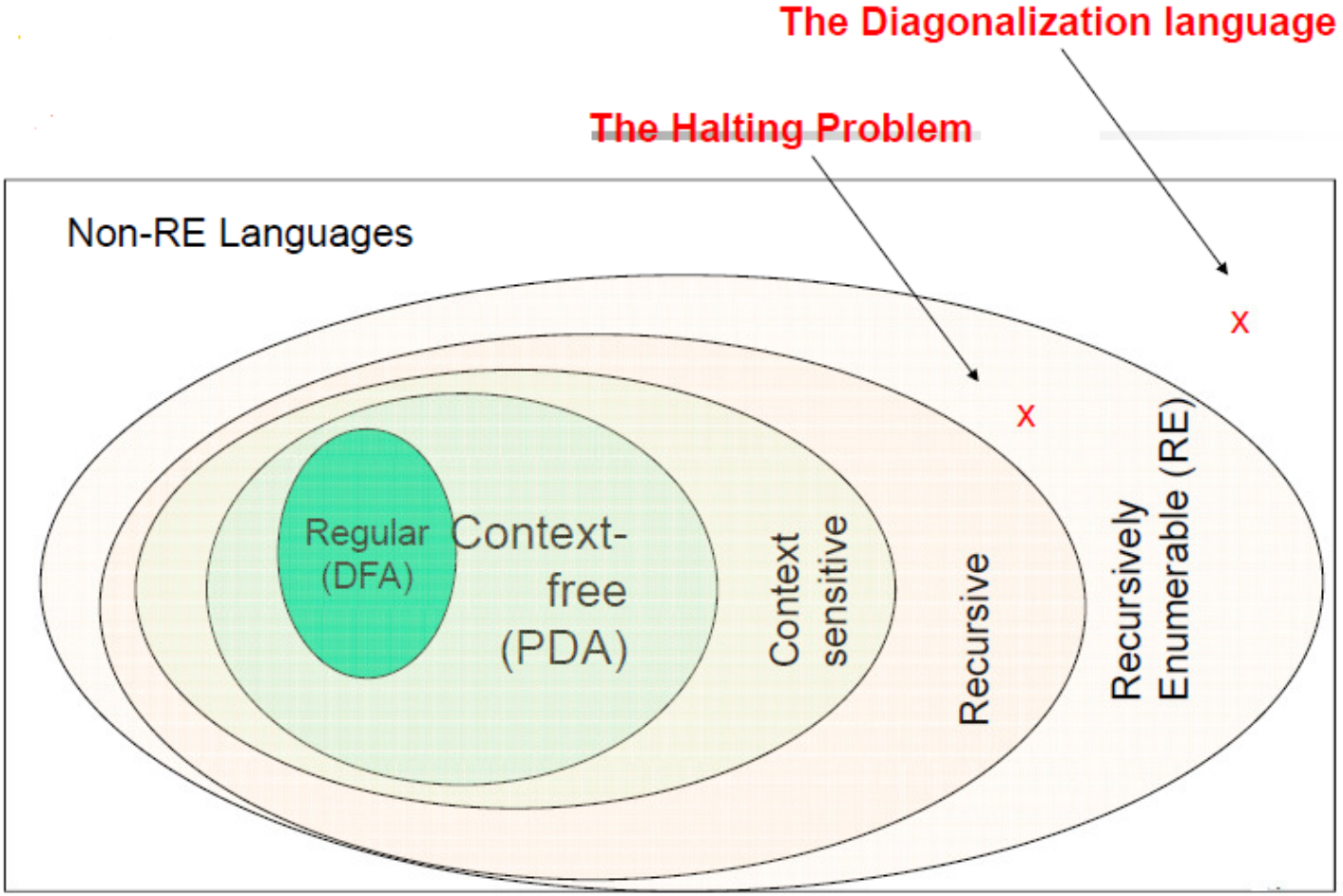
- Table: T[i,j] = 1, if M_i accepts w_j
  = 0, otherwise.

- Make a new language called
  $L_d = \{w_i \mid T[i,i] = 0\}$

Pada contoh di atas, L_d mengandung string w_1, w_3, dan seterusnya.

L_d bukan RE

- Akan dibuktikan bahwa L_d bukan bahasa RE (non RE), yaitu tidak ada mesin Turing yang menerima L_d .



- Pembuktian dengan cara kontradiksi:

  Misalkan L_d adalah RE, berarti ada mesin Turing M untuk L_d atau kita tulis $L_d = L(M)$.

  Karena L_d adalah bahasa terhadap alfabet {0, 1}, maka M harus sama dengan salah satu kode untuk M, misalkan k, yaitu $M = M_k$.

  Misalkan w_k adalah sebuah string.

  Pertanyaan: Apakah $w_k \in L_d$?

  1. Jika $w_k \in L(M_k) \Rightarrow T[k,k] = 1 \Rightarrow w_k \notin L_d$ (kontradiksi)
  2. Jika $w_k \notin L(M_k) \Rightarrow T[k,k] = 0 \Rightarrow w_k \in L_d$ (kontradiksi)

  KONTRADIKSI!!!

  Kesimpulan: Mesin Turing untuk L_d tidak ada, sehingga L_d bukan RE

Bahasa Universal

- Bahasa universal (L_u) adalah himpunan string biner yang terdiri dari pasangan kode <*M*, *w*> sedemikian sehingga *w* adalah anggota *L*(*M*).

  L_u = { <*M*, *w*>| *M* menerima w}

- Jadi, L_u terdiri dari semua string yang merepresentasikan kode biner mesin Turing dan input yang diterima oleh mesin Turing tersebut.

- Bahasa universal dikenali oleh mesin Turing universal *U* sedemikian sehingga L_u = *L(U*).

- Karena input yang diberikan kepada *U* adalah string biner, maka *U* sama dengan beberapa M_j yang terdapat di dalam daftar kanonik mesin Turing.

- Mesin Turing universal *U* mensimulasikan perilaku mesin *M* bila diberikan input *w*.

- *U* menerima pasangan kode <*M*, *w*> jika dan hanya jika *M* menerima *w*.

Simulasi oleh Mesin Turing Universal

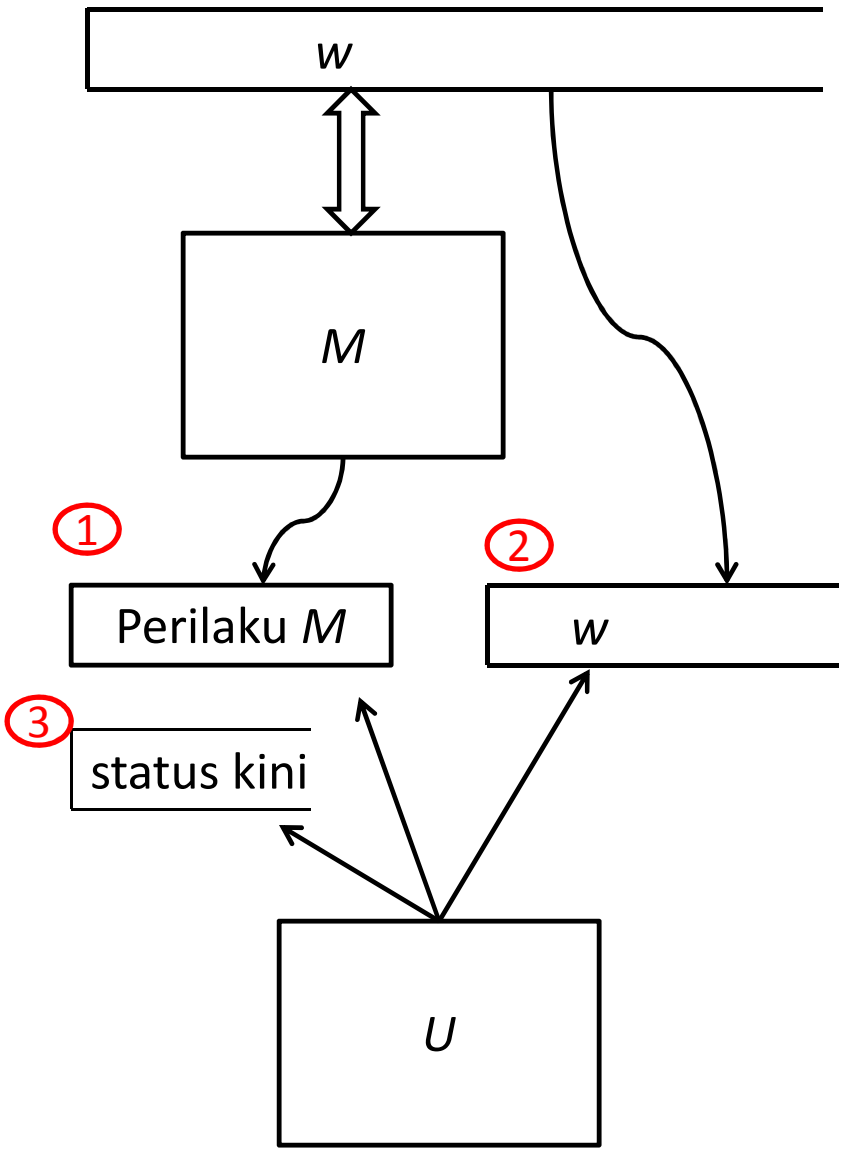
- Andaikan mesin Turing universal *U* akan mensimulasikan pengenalan string masukan *w* oleh mesin Turing *M*

  $M = (Q, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$

  seperti ditunjukkan pada Gambar di halaman berikut.

- Untuk membantu kerjanya, mesin Turing *U* dilengkapi oleh tiga pita.

- Pita pertama berisi deskripsi mesin Turing *M* yang akan disimulasikan, pita kedua berisi rangkaian simbol yang akan dikenali oleh *M*, dan pita ketiga berisi status kini dari mesin *M*.



Simulasi *M* oleh mesin Turing Universal *U*

- Mesin Turing universal U bekerja dengan cara berikut:

  1. Pita 2 akan diinisialiasi dengan input *w*, dan pita 3 diisi dengan simbol 0 untuk menyatakan status awal *M*, yaitu q_1.

  2. Jika pita 3 berisi simbol 00 maka pensimulasikan *M* oleh U dihentikan karena berarti mesin Turing sudah mencapai status akhirnya, q_2.

  3. Misalkan X_j adalah simbol yang sedang dibaca pada pita 2 dan pita 3 berisi simbol 0^i yang menyatakan status kini q_i dari *M*. Mesin Turing *U* harus memeriksa pita 1 untuk menemukan string yang dimulai dengan 110^i10^j1 (yang menandakan transisi $\delta(q_i, X_j)$). Ada dua kemungkinan kasus yang terjadi:

Kasus 1: Jika tidak ditemukan *string* tersebut, maka simulasi dihentikan dan berarti input *w* tidak diterima oleh *M*.

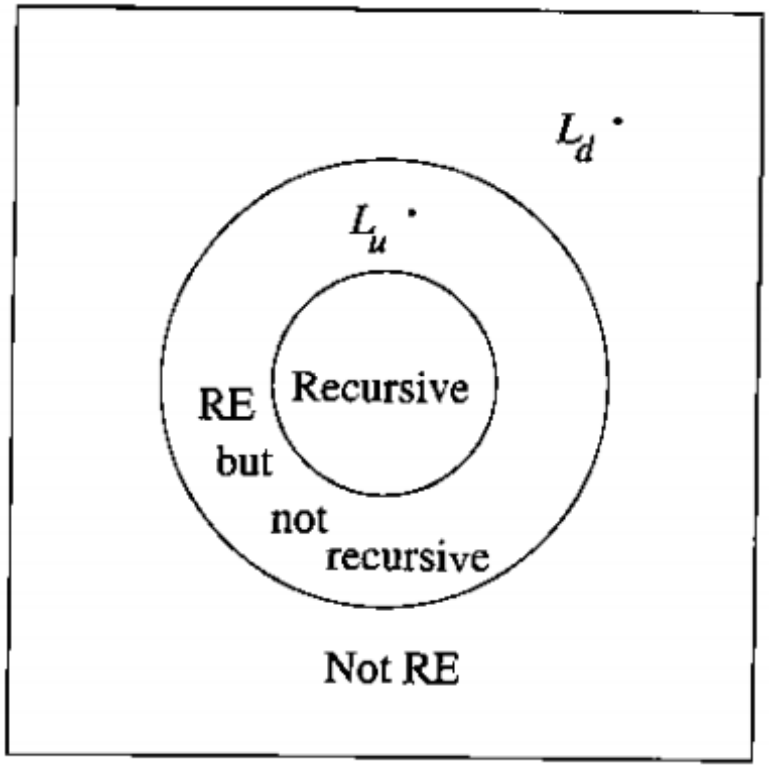
Kasus 2: Jika ditemukan, string $110^i10^j10^k10^l10^m$, maka

(a) simpan 0^k pada pita 3

(b) tuliskan simbol X_l pada sel yang sedang dibaca pada pita 2

(c) gerakkan *head* 2 ke arah D_m.

- **Teorema**. L_u adalah RE tetapi tidak rekursif



Ringkasan tentang L_d dan L_u

1. L_d adalah *undecidable* (tidak rekursif), tetapi bukan RE (non-RE)
2. L_u adalah *undecidable*, tetapi RE.
3. $\overline{L_u}$ seperti L_d, bukan RE (non-RE).
4. $\overline{L_d}$ seperti L_u, yaitu RE.

Bahasa Kosong (L_e) dan BahasaTidak Kosong (L_{ne})

- Misalkan *w* adalah string biner, yang merepresentasikan mesin Turing M_i.
- Jika $L(M_i) = \varnothing$, artinya M_i tidak menerima input apapun, maka *w* adalah elemen L_e.
- Jadi, L_e adalah bahasa yang berisi semua kode mesin Turing yang bahasanya kosong.

  $L_e = \{ M \mid L(M) = \varnothing \}$
- Jika $L(M_i) \neq \varnothing$, maka *w* adalah elemen bahasa L_{ne}

  $L_{ne} = \{ M \mid L(M) \neq \varnothing \}$
- Baik L_e maupun L_{ne} saling komplemen satu sama lain.

Teorema-teorema mengenai L_e dan L_{ne}

- **Teorema**. L_{ne} adalah RE
- **Teorema**. L_{ne} tidak rekursif
- **Teorema**. L_e adalah non-RE

Reduksi

- Sebuah persoalan dapat direduksi menjadi persoalan lain namun menghasilkan jawaban sama.

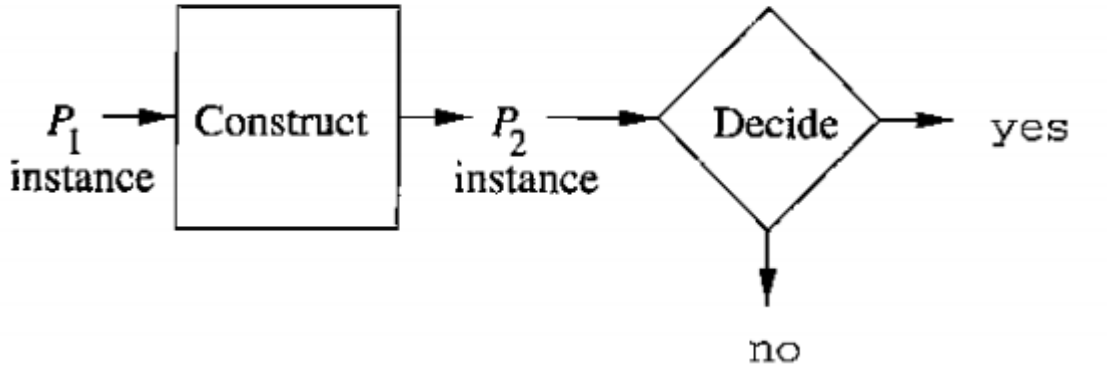
- Misalnya, persoalan perkalian direduksi menjadi persoalan penjumlahan.

  Contoh: 5 x 6 = 6 + 6 + 6 + 6 + 6

- Reduksi berguna untuk membuktikan sebuah persoalan *undecidable* apabila diberikan persoalan lain yang sudah diketahui *undecidable*.

- Misalkan P1 diketahui *undecidable*, dan kita ingin membuktikan bahwa sebuah persoalan baru, P2, *undecidable*.

- Caranya: Reduksi P1 menjadi P2 (instans persoalan P1 dikonversi menjadi instans persoalan P2)



P2 sedikitnya sesukar P1, artinya jika P1 tidak rekursif, maka P2 tidak mungkin rekursif. Jika P1 non-RE, maka P2 tidak mungkin RE

- Kotak *diamond* berlabel “Decide” adalah program yang mencetak “yes” atau “no”, bergantung pada apakah instans persoalan P2 adalah elemen atau bukan elemen bahasa yang berkoresponden dengan persoalan tersebut.

- Untuk membuktikan bahwa P2 *undecidable*, kita harus menemukan cara konstruksi instans P1 menjadi instans P2 yang memiliki jawaban sama.

- Cara konstruksinya:

  - sembarang string di dalam bahasa P1 dikonversi menjadi string di dalam bahasa P2,

  - sembarang string pada alfabet P1 yang tidak ada di dalam bahasa P1 dikonversi menjadi string yang tidak ada di dalam bahasa P2.

Dengan demikian, kita dapat memecahkan P1 sebagai berikut:

1. Diberikan instans P1, yaitu diberikan string *w* yang mungkin ada/tidak ada di dalam bahasa P1. Lakukan konstruksi untuk menghasilkan string *x*.
2. Uji apakah *x* di dalam P2, dan berikan jawaban yang sama tentang *w* dan P1.

   - Jika *w* anggota P1, maka *x* anggota P2, dan "Decide" memberikan jawaban "yes".

   - Jika *w* bukan anggota P1, maka *x* bukan anggotaP2, dan "Decide" memberikan jawaban "no" .

Jadi, kita telah menunjukkan bahwa P2 *decidable*.

Hal ini kontradiksi, karena kita sudah mengetahui P1 *undecidable* (karena algoritma untuk menentukan keanggotaan string di dalam P1 tidak pernah ada – ingatlah kembali bahwa *membership problem* adalah *undecidable*), oleh karena itu P2 haruslah *undecidable*.

- void foo(char* bar) { printf("%s", bar); }