IF5110 Teori Komputasi

4. Undecidabality (Bagian 3)

Oleh: Rinaldi Munir

Program Studi Magister Informatika STEI-ITB

Reduksi

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

- Misalnya, persoalan perkalian direduksi menjadi persoalan penjumlahan.

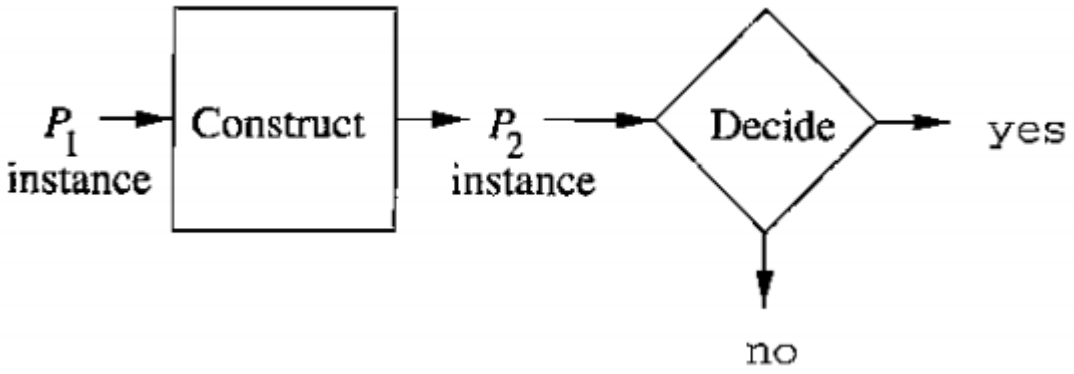
  Contoh: 5 x 6 = 6 + 6 + 6 + 6 + 6

- 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*.

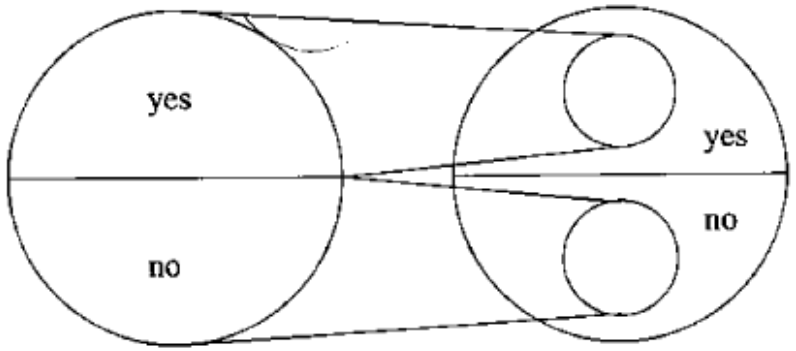
- Caranya: Reduksi P1 menjadi P2 (instans persoalan P1 dikonversi menjadi instans persoalan P2)

  Kita katakan: *P1 reduced to P2*



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

- Proses reduksi memberikan jawaban sebagai berikut:
  - Jika P1 memberikan jawaban “yes”, maka jawaban tersebut dikembalikan menjadi jawaban P2 juga (Decide → “yes”)
  - Jika P2 memberikan jawaban “no”, maka jawaban tersebut dikembalikan menjadi jawaban P2 juga (Decide → “no”)



Proses reduksi dinyatakan dalam langkah-langkah berikut:

1. Diberikan instans P1, yaitu diberikan string *w,* lalu lakukan konstruksi untuk menghasilkan string *x*.
2. Asumsikan P2 *decidable*.
3. Uji apakah *x* anggota 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 anggota P2, 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*.

- Secara formal, reduksi dari P1 ke P2 adalah mesin Turing yang mengambil instans P1 yang tertulis pada pitanya dan berhenti dengan instans P2 pada pitanya.

- Secara praktik, reduksi digambarkan dengan sebuah program yang mengambil instans P1 sebagai input dan menghasilkan instans P2 sebagai output.

- **Teorema**: Jika terdapat reduksi dari P1 ke P2, maka

  a) Jika P1 *undecidable*, maka P2 juga *undecidable*

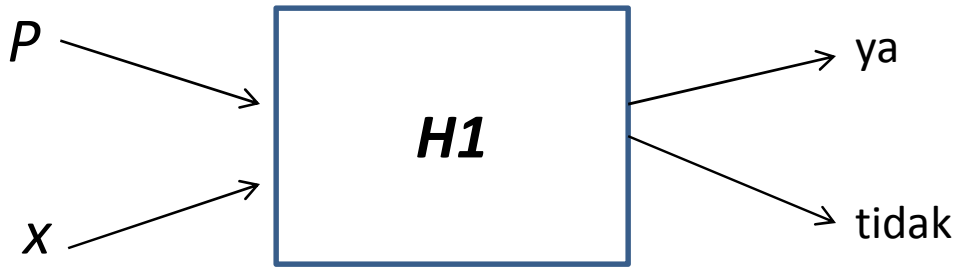
  b) Jika P1 non-RE, maka P2 juga non-RE

Contoh: Misalkan,

P1 : Persoalan *Hello-World*

Diberikan program *P* dan input *x*. Apakah *P*, bila diberikan input *x*, mencetak "Hello, world!" sebagai luaran pertamanya?

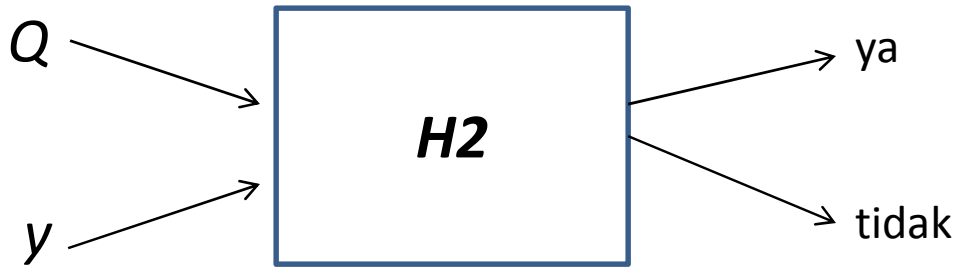
→ Sudah dibuktikan *undecidable*



P2: Persoalan *Call-foo*

Diberikan program *Q* dan input *y*. Apakah *Q*, bila diberikan input *y*, pernah memanggil fungsi *foo*?

→ Akan ditunjukkan *undecidable*



```
void foo(char* str)
{  printf("%s", str);
}
```

Kasus 1: Program *Q* tidak memiliki fungsi *foo*, maka jawaban persoalan ini mudah ditentukan → No

Kasus 2: Program *Q* memiliki fungsi *foo*, tetapi mungkin pernah atau tidak pernah dipanggil dengan input *y*.

Contoh pemanggilan : `if (y % 2 == 0) { foo(“Hello, world!”); }`

→ jika *x* ganjil, maka *foo* tidak pernah dipanggil

- Proses reduksi dilakukan sebagai berikut:

  A. Konstruksi instans P1, yaitu <P, x> menjadi instans P2, yaitu <Q,y>.

  Misalkan hasil konstruksi adalah program R dengan input z, atau <R,z>. Program *R* dengan input *z* memanggil *foo* jika dan hanya jika *Q* dengan input *y* mencetak “Hello, world!”. Metode konstruksi R adalah sebagai berikut:

  1. Jika Q memiliki fungsi yang memanggil *foo*, ubah nama (*rename*) fungsi tersebut serta semua pemanggilan ke fungsi tersebut. Sebut nama program hasil modifikasi ini adalah Q1.

2. Tambahkan fungsi *foo* ke Q1. Fungsi ini ini tidak dipanggil. Sebut program hasil modifikasi ini Q2.

3. Modifikasi Q2 untuk mengingat string “Hello, world!” yang akan ia cetak dengan cara menyimpan string “Hello, world!” sepanjang 13 karakter ke dalam larik *A*. Sebut program hasil modifikasi adalah Q3.

4. Modifikasi Q3 sedemikian sehingga bila ia mengeksekusi suatu perintah cetak, ia memeriksa larik *A* untuk melihat apakah ia mencetak string sepanjang 13 karakter atau lebih. Jika ya, periksa apakah string yang dicetak itu adalah “Hello, world!”. Jika ya, panggil fungsi *foo* yang ditambahkan pada langkah 2.

5. Program yang dihasilkan dari langkah 1 s/d 4 adalah *R* dengan input $z = y$.

B. Decides

- Asumsikan persoalan persoalan *Call-foo decidable*, yaitu terdapat algoritma yang bisa memecahkan Persoalan *Call-foo*.

- Misalkan *Q* dengan input *y* mencetak “Hello, world!” sebagai luaran pertamanya, maka *R* akan memanggil *foo*.
- Namun, jika Q dengan input *y* tidak mencetak “Hello, world”, maka R tidak akan pernah memanggil *foo*.
- Jika kita dapat memutuskan apakah R dengan input *z* memanggil *foo*, maka kita juga tahu apakah *Q* dengan input *z* (ingat *y* = *z*) mencetak “Hello, world!”.
- Karena kita sudah tahu bahwa tidak ada algoritma untuk memecahkan persoalan *Hello-world*, maka asumsi bahwa terdapat algoritma untuk memecahkan persoalan *call-foo* tenyata salah.
- Tidak ada algoritma untuk memecahkan persoalan *call-foo*, dengan kata lain persoalan *call-foo* adalah *undecidable*.

- **Contoh**: Kita sudah mengetahui bahwa *membership problem* adalah *undecidable*. Bahasa yang berkorepsonden dengan *membership problem* adalah*:*

  L1 = L(M) = { <M,w>| Mesin Turing M menerima w }

  Kita ingin menunjukkan bahwa bahasa yang berkoresponden dengan *halting problem* adalah juga *undecidable:*

  L2 = L(M) = { <M,w>| Mesin Turing M berhenti pada input w }

  Kita akan menunjukkan bahwa L1 dapat direduksi menjadi L2.

Proses reduksinya adalah sebagai berikut:

- Asumsikan L2 *decidable*, yaitu terdapat mesin Turing R yang mensimulasikan L2 dan memberikan jawaban ”yes” atau ”no”.
- Konversi input <M, w> menjadi <M2, w2>, dalam hal ini M2 = M dan w2 = w.
- Jalankan R dengan input <M2, w2>

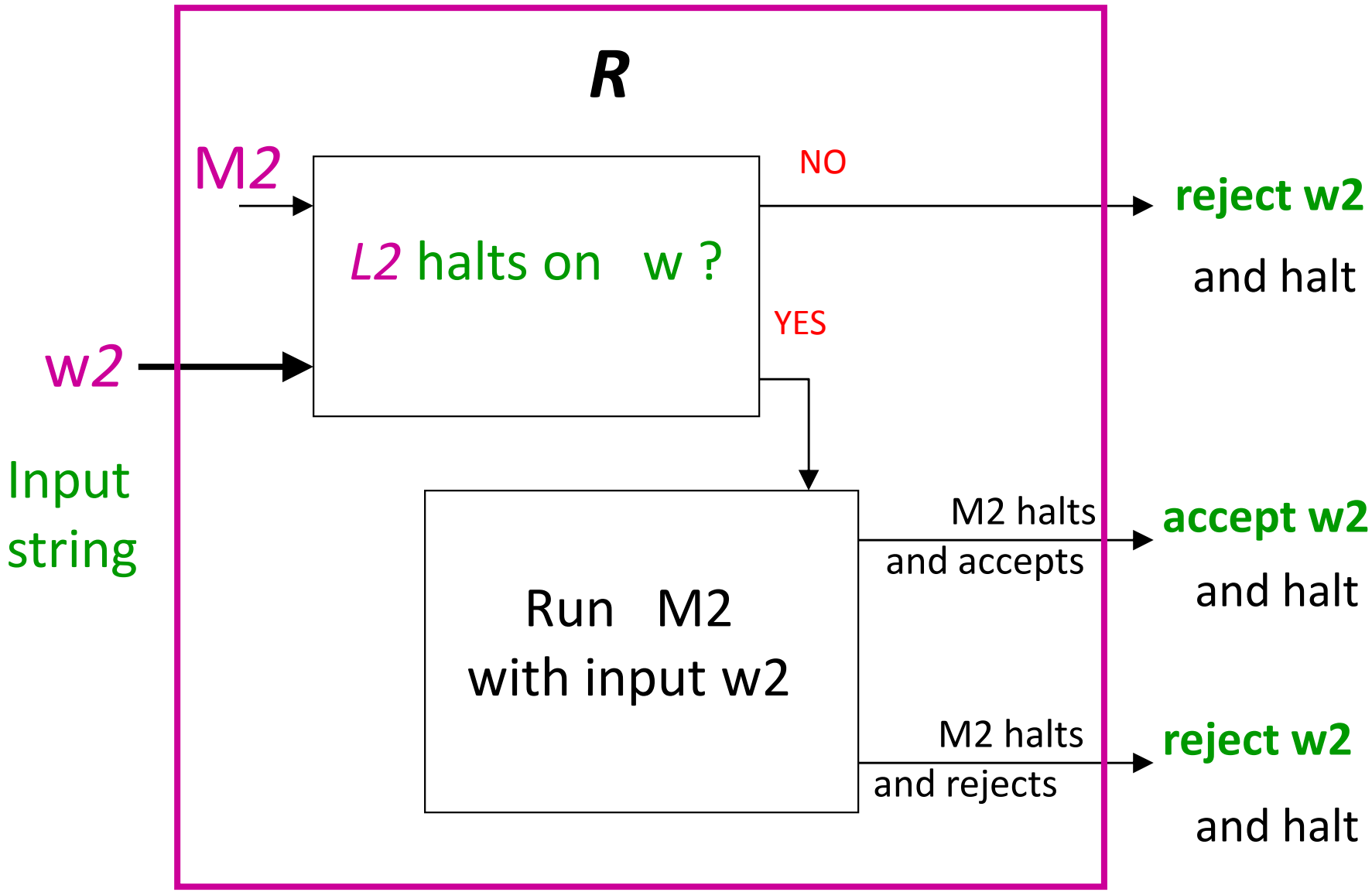
  a) Jika R menolak <M2, w2>, maka ”decides” → ”no”

  b) Jika R menerima <M2, w2>, yaitu M2 berhenti pada w2, maka jalankan M2 dengan input w2, ada dua kemungkinan:

  – M2 menerima w2, maka ”decides” → ”yes”

  – M2 menolak w2, maka ”decides” → ”no”

Decider for *L2*



Jadi, R dapat memberikan jawaban ”yes” atau ”no”.

Namun, kita mengetahui bahwa L1 *undecidable*.

Jadi, R tidak mungkin ada, sehingga L2 haruslah *undecidable.*

Referensi

1. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, *Introduction To Automata Theory , Languanges, and Computation 3rd Edition,* Addison Wesley, 2007.

2. Costas Busch – RPI, Fall 2006, *Undecidable Problems (unsolvable problems) and Decidable Languages*