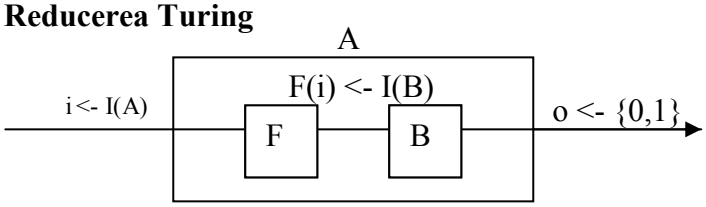
Metode de a demonstra (ne)decidabilitatea unor probleme
- brief -

Reducerea Turing



A = algoritm pt problema PA (pe care o reduc)
B = algoritm pt problema PB (la care reduc)
F = procedura de transformare a intrarilor pt A in intrari pt B

Daca exista o astfel de procedura care transforma **orice** instanta i a problemei PA intr-**o** instanta F(i) a problemei PB a.i. **A(i)=1 ⇔ B(F(i))=1**, atunci spunem ca PA se reduce Turing la PB: **$PA \leq_T PB$**

Semnificatia este ca PA este "mai usoara" ca PB: daca stim sa rezolvam PB, stim sa rezolvam si PA: nu trebuie decat sa aplicam procedura F si sa transformam intrarea oarecare pt PA intr-o intrare data pt PB, apoi sa folosim algoritmul de rezolvare pentru B.

Obs1: faptul ca reducerea nu merge (neaparat) si de la PB la PA se vede in sublinierea cuvintelor "orice" si "o". Cand reducem PA la PB luam o intrare oarecare pt PA si **construim** in mod convenabil o intrare pt PB. Practic, F(i) acopera, de obicei, doar o parte dintre toate intrarile posibile pentru problema B.

Obs2: demonstratia faptului ca o problema se reduce Turing la alta nu e completa daca nu demonstrati **implicatia in ambele sensuri**! Intai trebuie demonstrat ca A(i)=1 => B(F(i))=1, apoi ca B(F(i))=1 => A(i)=1. Daca demonstrati doar prima parte garantati doar ca B nu transforma instantele-da ale lui A in instante-nu, insa nu garantati faptul ca nu inventeaza instante-da noi ("*false positives*"). Doar prin demonstrarea implicatiei inverse, se garanteaza ca problema B nu inventeaza instante-da noi pentru problema A.

Aceasta observatie este valabila in sens larg pt orice echivalenta: de fiecare data cand aveti de demonstrat o echivalenta, trebuie demonstrata dubla implicatie, altfel demonstratia e incompleta!

Consecinte ale stabilirii unei relatii de reducere Turing:

1) **$A \leq_T B$ /\ A nu este decidabila => B nu este decidabila** (altfel ar deveni si A decidabila) (aici intra si problemele semidecidabile si cele nedecidabile)
2) **$A \leq_T B$ /\ B decidabila => A decidabila** (altfel ar deveni si B semidecidabila sau nedecidabila)

Aceste consecinte se folosesc pt a demonstra (ne)decidabilitatea unor probleme, folosind reduceri (de) la probleme a caror (ne)decidabilitate este cunoscuta.

Post Correspondence Problem (problema care s-a demonstrat ca nu este decidabila)

- dandu-se 2 liste de cate n cuvinte (w1,w2,...,wn) si (x1,x2,...,xn), exista o secventa nevida de k intregi i1,i2,...,ik a.i. wi1 wi2 ... wik = xi1 xi2 ... xik? (unde wi1 wi2 ... wik reprezinta concatenarea cuvintelor wi1,wi2,...,wik, iar secventa nu trebuie sa fie de intregi diferiti, acelasi indice poate fi folosit de mai multe ori)
- intuitiv: pt fiecare k (1,2,3,…) am putea incerca toate combinatiile de indici; daca gasim o solutie, intoarcem DA; daca insa nu gasim, nu avem niciodata garantia ca nu va exista o solutie pt un k si mai mare; problema ar fi decidabila daca am putea impune o limita superioara asupra lui k, insa in enuntul curent nu putem. Rezulta ca problema PCP este **semidecidabila**.

Exemplu: L1 = (a, bb, a), L2 = (aa, b, bb)
O viziune mai convenabila e sa ne imaginam perechile cu acelasi indice ca pe niste dominouri pe care trebuie sa le lipim unul de altul pana cand partea de sus arata ca partea de jos (cu precizarea ca putem folosi acelasi domino de oricate ori). Rezulta:

a	bb	a
---	---	---
aa	b	bb

O solutie ar fi (1,3,2,2) care in ambele cazuri produce aabbbb:

a	a	bb	bb
---	---	---	---
aa	bb	b	b

Exercitii: gasiti, daca exista, solutii pt urmatoarele instante PCP:

1) L1 = (ab, baa, aba), L2 = (aba, aa, baa)
2) L1 = (bb, ab, c), L2 = (b, ba, bc)

Exemplu de reducere Turing: MPCP $\leq_T$ PCP
MPCP = PCP cu restrictia ca solutiile trebuie sa inceapa cu indicele 1 (perechea (w1,x1) e mereu la inceputul solutiei)

De facut: dandu-se o instanta oarecare i pt MPCP, trebuie sa ne construim o instanta convenabila F(i) pt PCP, a.i. MPCP(i)=1 $\Leftrightarrow$ PCP(F(i))=1.
Truc: introducem un simbol nou, *.
Constructie:

- fata de cum arata **L1** in MPCP, in PCP introducem *** dupa** fiecare simbol
- fata de cum arata **L2** in MPCP, in PCP introducem *** inainte** de fiecare simbol
- exemplu: perechea (ab,bb) devine (a*b*,*b*b)
- pt perechea (w1,x1) cu indice 1 adaugam in plus o pereche in care * sunt plasate dupa regula de mai sus, doar ca w1 mai primeste o * in fata (o numim perechea 0)
- exemplu: perechea cu indice 1 (baa,ab) genereaza o pereche 0 (*b*a*a*,*a*b)
- in acest fel ne-am asigurat ca aceasta va fi prima aleasa in solutie, pt ca este singura pereche in care cuvintele incep cu acelasi simbol
- intrucat trebuie sa facem ceva similar si cu finalul (altfel partea de sus se va termina cu * iar cea de jos nu), introducem si perechea finala ($,*$)

Demonstratie a corectitudinii reducerii:

- MPCP(i)=1 => PCP(F(i))=1: daca MPCP are o solutie, atunci solutia pt instanta construita mai sus pt PCP se obtine din perechea 0 in locul perechii 1 din solutia MPCP, urmata de fix perechile corespunzatoare solutiei din MPCP (cu acelasi indice), urmate de perechea finala.
- PCP(F(i))=1 => MPCP(i)=1: daca PCP are o solutie, ea trebuie sa inceapa cu perechea 0 (pt ca nicio alta pereche nu are cuvinte care sa inceapa cu acelasi simbol) si sa se termine cu perechea finala (din motive similare). Stergand * si $ din solutia PCP vom obtine o solutie pt MPCP.

Utilizari frecvente ale PCP: demonstrarea nedecidabilitatii unor proprietati ale gramaticilor. Vom exemplifica si pt aceasta vom introduce notiunea de gramatici independente de context (context free grammars in literatura in lb engleza).

Gramatici independente de context (GIC)
Gramatica = set de reguli pentru a forma cuvinte intr-un anumit limbaj.
Gramatica (formala oarecare) = {N,T,S,P}
N = neterminali (variabile care se pot inlocui conform regulilor de productie din P)
T = terminali (simboluri dintr-un anumit alfabet, de ex literele de la a la z)
S = simbol de start (este un neterminal care va evolua conform regulilor din P)
P = reguli de productie

GIC = gramatici cu proprietatea ca toate regulile din P sunt de forma V->w
V = un singur neterminal
w = sir de terminali si/sau neterminali, posibil vid

Independenta de context se refera la faptul ca neterminalii pot fi rescrisi indiferent de contextul in care apar (ceea ce nu se intampla la o regula de genul w1Vw2->w3, de exemplu).

Limbajul generat de o gramatica G (notat L(G)) reprezinta multimea tuturor cuvintelor care se pot obtine folosind regulile din G.

Exemplu:
S->a | aS | bS (S trece in a sau in aS sau in bS)
Limbajul generat este multimea tuturor cuvintelor nevide formate din simbolurile a si b, care se termina obligatoriu in a.

Demonstrarea nedecidabilitatii $L(G1) \cap L(G2) \neq \emptyset$, cu G1 si G2 GIC oarecare

Plan: aratam ca PCP $\leq_T$ PI (problema intersectiei). Cum PCP este nedecidabila si este in acest caz problema "mai usoara" => PI este tot nedecidabila.
De facut: dandu-se o instanta oarecare i pt PCP, trebuie sa ne construim o instanta convenabila F(i) pt PI, a.i. PCP(i)=1 $\Leftrightarrow$ PI(F(I))=1.
Truc: introducem simbolurile noi (care nu se aflau in alfabetul instantei PCP) a1,a2,...,an (n simboluri, cat era lungimea unei liste din instanta PCP).

Constructie:

- definim G1 si G2 – GIC pe alfabetul instantei PCP reunit cu simbolurile a1,...,an
- S1 -> w1S1a1 | w2S1a2 | ... | wnS1an
- S1 -> w1a1 | w2a2 | ... | wnan
- S2 -> x1S2a1 | x2S2a2 | ... | xnS2an
- S2 -> x1a1 | x2a2 | ... | xnan

Demonstratie a corectitudinii reducerii:

- PCP(i)=1 => PI(F(i))=1: daca PCP are o solutie, atunci folosim indicii din solutie (i1,i2,...,ik) si regulile S1 -> wi1S1ai1 -> wi1wi2S1ai2ai1 ... -> wi1wi2...wikaik...ai2ai1 si similar pt S2 -> xi1xi2...xikaik...ai2ai1 si observam ca obtinem acelasi cuvant, deci intersectia e nevida => PI(F(i))=1
- PI(F(i))=1 => PCP(i)=1: daca intersectia limbajelor din PI e nevida inseamna ca exista o serie de indici (i1,i2,...,ik) ai regulilor de productie alese a.i. wi1wi2...wikaik...ai2ai1 = xi1xi2...xikaik...ai2ai1 (trebuie sa fie aceiasi indici pt ca e singura posibilitate ca cele 2 cuvinte sa se termine in acelasi sir de simboluri a). Acest sir de indici devine, evident, solutie pt PCP.

Obs: Aceasta problema e nedecidabila pt 2 GIC oarecare. Nu inseamna ca daca imi iau 2 GIC anume nu pot demonstra ca intersectia lor e nevida, ci ca nu pot avea un algoritm/program care imi primeste ca parametri 2 GIC oarecare (despre care nu mai stie nimic altceva apriori) si decide daca intersectia lor e vida sau nu. Observatia e usor de generalizat si pt alte probleme nedecidabile.

Exercitiu: demonstrati nedecidabilitatea ambiguitatii GIC.
Indicatii:

- o GIC G este ambigua daca exista un cuvant w<-L(G) care poate fi derivat in 2 modalitati diferite (folosind, desigur, regulile de productie din G)
- foarte util este sa intelegeti intai demonstratia anterioara

Concluzii:

- notiunile de calculabilitate si decidabilitate au un istoric mai complicat decat forma in care ni se prezinta noua azi (v. Hilbert, Goedel)
- notiunea abstracta de calculabilitate nu se traduce direct intr-o notiune practica de calculabilitate (v. teza Church-Turing)
- probleme decidabile si semidecidabile apar adesea in matematica
- problema opririi (HALT) este exemplul clasic de problema semidecidabila (care nu e si decidabila); nu incercati sa construiti algoritmi care depind de faptul ca un anumit program oarecare se termina, este inutil
- alte probleme nedecidabile faimoase: PCP, Hilbert 10th Problem; toate acestea se folosesc frecvent pt a demonstra (ne)decidabilitatea altor probleme, folosind mecanismul de reducere
- reducerea unei probleme la alta este un mecanism de baza in AA (il vom reintalni si in alte demonstratii, v. capitolul “clase de probleme”)

- cunoasterea nedecidabilitatii unei probleme ne poate conduce la a cauta cazuri particulare pe care problema sa fie decidabila; adesea problemele nedecidabile au cazuri particulare a caror decidabilitate ramane o problema (vezi PCP limitat la 3-6 dominouri)