Algoritmid ja andmestruktuurid

- Rasked ja lootusetud ülesanded
- Sissejuhatus NP teooriasse

$$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$$

Rasked ülesanded

- Kust läheb piir “mõistlike” ja “raskete” ülesannete vahel?

  lg n, n, n lg n, n^k, k^n, $n!$, n^n , ...

	10	50	100	300	1000
5n	50	250	500	1500	5000
n lg n	33	282	665	2467	9966
n^3	1000	125000	10^6	10^8	10^9
2^n	1024	10^{16}	10^{31}	10^{91}	10^{302}
n!	10^7	10^{65}	10^{161}	10^{623}	(∞)

Sisendi suurus

- Andmestruktuuridel selle elementide arv
- Stringidel tähtede/sümbolite arv
- Numbrilisel sisendil
  - sisendi väärtus
  - bittide või baitide arv, mis on vajalik kodeerimiseks

Millise keerukusega on algarvulisuse kontroll?

```
bool prime (int n)
{ int i;
  for( i = 2; i < n; i++)
    if( n % i == 0 )
      return( false );
  return( true );}
```

Sisendi suuruse (bittide arvu) suhtes eksponentsiaalne!

Ülesannete klassid

- ülesanded mille jaoks on olemas polünomiaalne algoritm

  otsimine, sorteerimine

- ülesanded mille kohta on tõestatud, et sellist algoritmi ei eksisteeri
  - mittepolünomiaalse suurusega väljund

    Kõik teed graafis punktist A punkti B
  - mõistliku suurusega väljund

    Presburgeri aritmeetika
- ülesanded, mille kohta pole polünomiaalset algoritmi, ega tõestust selle puudumisest

  rändkaupmehe, seljakoti ülesanne

- algoritmiliselt mittelahenduvad ülesanded

  algoritmi peatuvuse probleem

Ülesannete liigitus väljundi järgi

- Ammendav lahenduste otsingu ülesanne

  väljundiks on kõikvõimalikud lahendid

  N: Leida graafi kõik Hamiltoni tsüklid (tsüklid, mis läbivad kõiki tippe ainult ühe korra)

- Optimiseerimisülesanne

  väljundiks parim lahend

  N: Leida lühim Hamiltoni tsükkel

- Otsustusülesanne

  väljundiks on kas “jah” või “ei”

  N: Kas esineb Hamiltoni tsükkel, mille kogupikkus < *k*

Otsustus ja optimiseerimisülesanded

- Otsustusülesande resultaadiks on jah/ei
- Igale otsustuülesandele vastab optimiseerimisülesanne ja vastupidi
  - Rändkaupmees

    Leidaˇlühim tee, mis läbib kõiki graafi punkte ühe korra

    Leida kas eksisteerib tee, mille pikkus on väiksem kui d
  - 0-1 seljakoti pakkimine

    Leida maksimaalne väärtus, mida saab kotti paigutada

    Leida kas kotti saab paigutada asju väärtuses v
- Otsustus- ja optimiseerimisülesanded on üksteiseks teisendatavad polünomiaalse keerukusega algoritmiga. Kuidas?

Ülesannete klass P

- Klass P on otsustusülesannete hulk, mida saab lahendada polünomiaalse ajalise keerukusega algoritmiga.
- Neid ülesandeid loetakse “mõistliku” keerukusega ülesanneteks
- On lahtine, kas rändkaupmehe ja seljakoti pakkimise ülesanne kuulub klassi P
  - keegi pole leidnud polünomiaalse keerukusega algoritmi
  - keegi pole tõestanud, et sellist algoritmi ei ole olemas

Polünomiaalne kontrollitavus

On antud

- otsustusülesanne
- tulemus (string, jada, muutujate väärtustus vms), mis on väidetavalt lahenduseks

Ülesanne on polünomiaalselt kontrollitav, kui suudame polünomiaalse algoritmiga kontrollida, kas pakutud tulemus on tegelikult lahenduseks

- kui kontrolli tulemus on positiivne, siis on otsustusülesande vastuseks "jah"
- vastasel korral ei saa ülesande kohta midagi väita. Saab öelda, et see ei olnud lahendus, aga võibolla on olemas mõni teine lahendus.

Mittedeterministlik algoritm

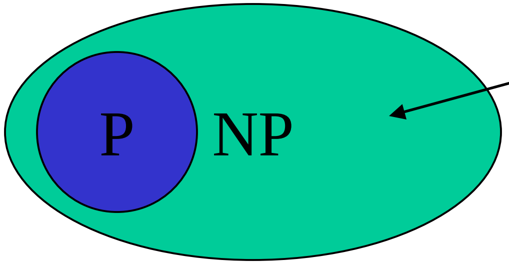
- Arvamine (mittedeterministlik osa - oraakel)

  Probleemi alusel pakutakse välja tulemus, mis võiks olla lahenduseks
- Kontroll (deterministlik osa)
  - Sisendiks on probleemi sisend ja pakutud tulemus.
  - Kontrolli teostav deterministlik algoritm
    - vastab “true”, kui tulemus lahendab probleemi
    - vastab “false”, kui tulemus ei lahenda probleemi
    - jääb lõpmatusse tsüklisse
- Probleemi lahendus:
  - “jah”kui mingi tulemuse korral saadakse “true”
  - “ei” kui iga võimaliku tulemuse korral saadakse “false”

Ülesannete klass NP

- *polünomiaalse ajaga mittedeterministlik algoritm* on mittedeterministlik algoritm, mille kontrolli osa on polünomiaalse ajalise keerukusega
- Klass NP on otsustusülesannete hulk, mida saab lahendada polünomiaalse ajalise keerukusega mittedeterministliku algoritmiga.
- NP ülesannete jaoks ei pruugi olemas olla polünomiaalse ajalise keerukusega deterministlikku algoritmi

$$P \subseteq NP$$

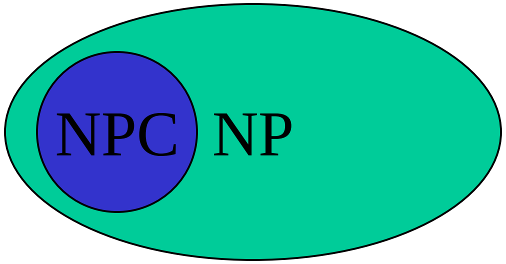


hulgas NP\P pole teada ühtegi probleemi

$$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$$

NPC (NP-täielikud) ülesanded

- NP otsustusülesannete hulk, mis on ekvivalentse keerukusega selles mõttes, et kui üks kuulub klassi P, siis kuuluvad sinna ka kõik teised NPC ülesanded.
- Omavahelised ekvivalentsiseosed luuakse polünomiaalse keerukusega teisendustega ühest ülesandest teise
- NPC klassi kuulub palju praktilisi ülesandeid
  - rändkaupmehe ülesanne
  - täisosaline seljakoti pakkimise ülesanne
  - Hamiltoni tsüklite ülesanne
  - CNF kehtestatavuse ülesanne
  - jne



CNF kehtestatavuse ülesanne

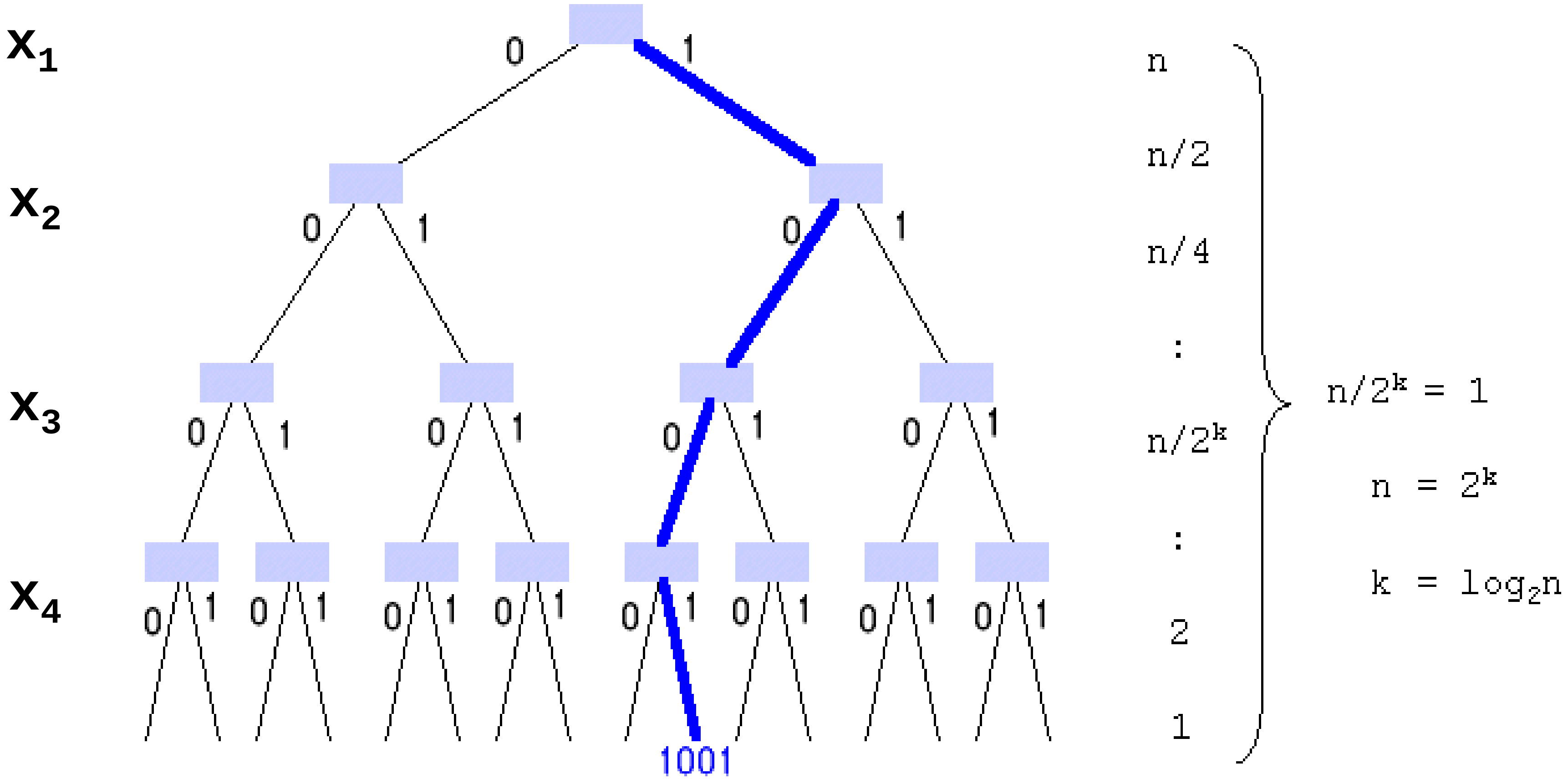
- CNF - lausearvutuse valemite konjuktiivne normaalkuju

$$(\overline{x}_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee x_3 \vee x_4)$$

- CNF kehtestatavuse otsustusülesanne:
  Leida, kas CNF kujul antud valemile esineb muutjate väärtustust, mis muudaksid valemi tõeseks.
- On tõestatud (S. Cook 1971), et kui selle ülesandel on P algoritm, siis P = NP.

CNF kehtestatavuse olekuruum

$$(\overline{x}_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_4) \wedge (\overline{x}_2 \vee x_3 \vee x_4)$$



Tulemuse kontroll

Ülesanne:

$$(\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee x_3 \vee x_4)$$

Väljapakutud väärtustus:

$$(\bar{x}_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1)$$

Väärtustuse kontroll:

$$(\bar{1} \vee 0 \vee \bar{0}) \wedge (1 \vee \bar{1}) \wedge (\bar{0} \vee 0 \vee 1) = 1$$

Teisendused ülesannete vahel

- Teisenduse kasutamine
  - Peame lahendama otsustusülesande A
  - Meil on algoritm ülesande B lahendamiseks
  - Kirjutame algoritmi, mis teisendab iga A sisendi B sisendiks, nii, et algoritm B annab õige vastuse (jah/ei)
- Näide

A: Kas vähemalt üks loogiline muutuja x_i etteantud n muutujast on tõene?

B: Kas suurim arv etteantud n täisarvust k_i on positiivne?

Teisendus:

$k_i = 1$ kui x_i on tõene

$k_i = 0$ kui x_i on väär

Otsustusülesande teisendatavus

- Teisendatavuse definitsioon

  Kui on olemas polünomiaalne mitu-ühele algoritm A teisendamiseks B-ks, siis on A polünomiaalselt teisendatav B-ks. Lühidalt:

  $$A \propto B$$

- Mitu-ühele:

  Mitu erinevat A sisendit võidakse teisendada samaks B sisendiks.

- Teoreem:

  Kui B kuulub klassi P ja $A \propto B$, siis
  A kuulub klassi P.

NP-täielikkus (NPC) formaalselt

- Def: Ülesanne B on NP-täielik (NPC) kui:
  - see on NP (on lahenduv mittedeterministliku polünomiaalse algoritmiga)
  - kõigi ülejäänud NP probleemide A kohta kehtib

    $A \propto B$

  $\Rightarrow$ kui mingi NPC probleem kuulub klassi P, siis NP = P
- Teoreem (Cook'i teoreem):

  CNF kehtestatavus on NP-täielik
- Teoreem: Ülesanne C on NP-täielik kui:
  - see on NP
  - mõne NP-täieliku probleemi B kohta kehtib

    $B \propto C$
- Tõestuse idee: $A \propto CNF \propto B \propto C$

Rändkaupmehe ülesanne on NPC

CNF kehtestatavus $\propto$

Hamiltoni tsüklite otsustusülesanne $\propto$

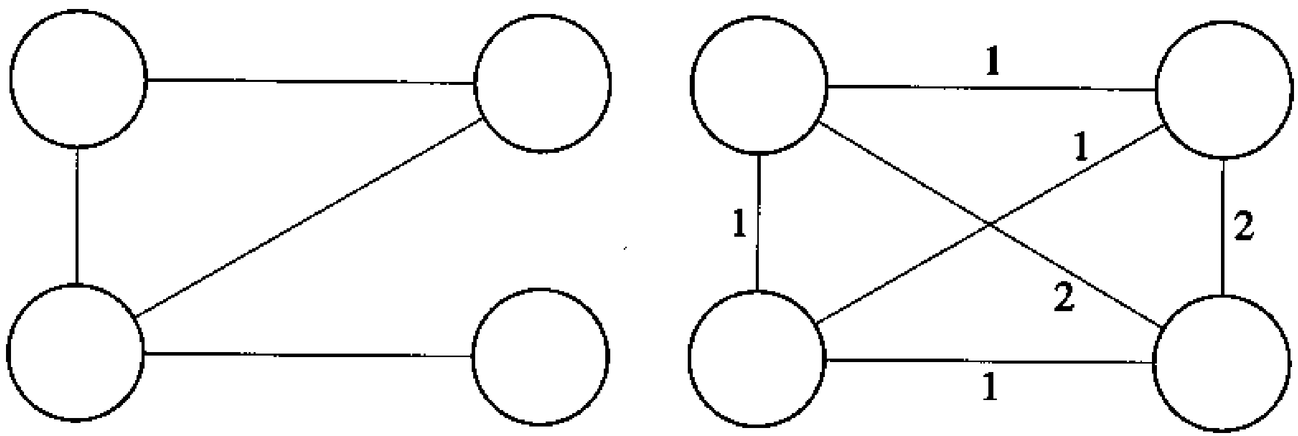
Rändkaupmehe ülesanne suunata graafil $\propto$

Rändkaupmehe ülesanne

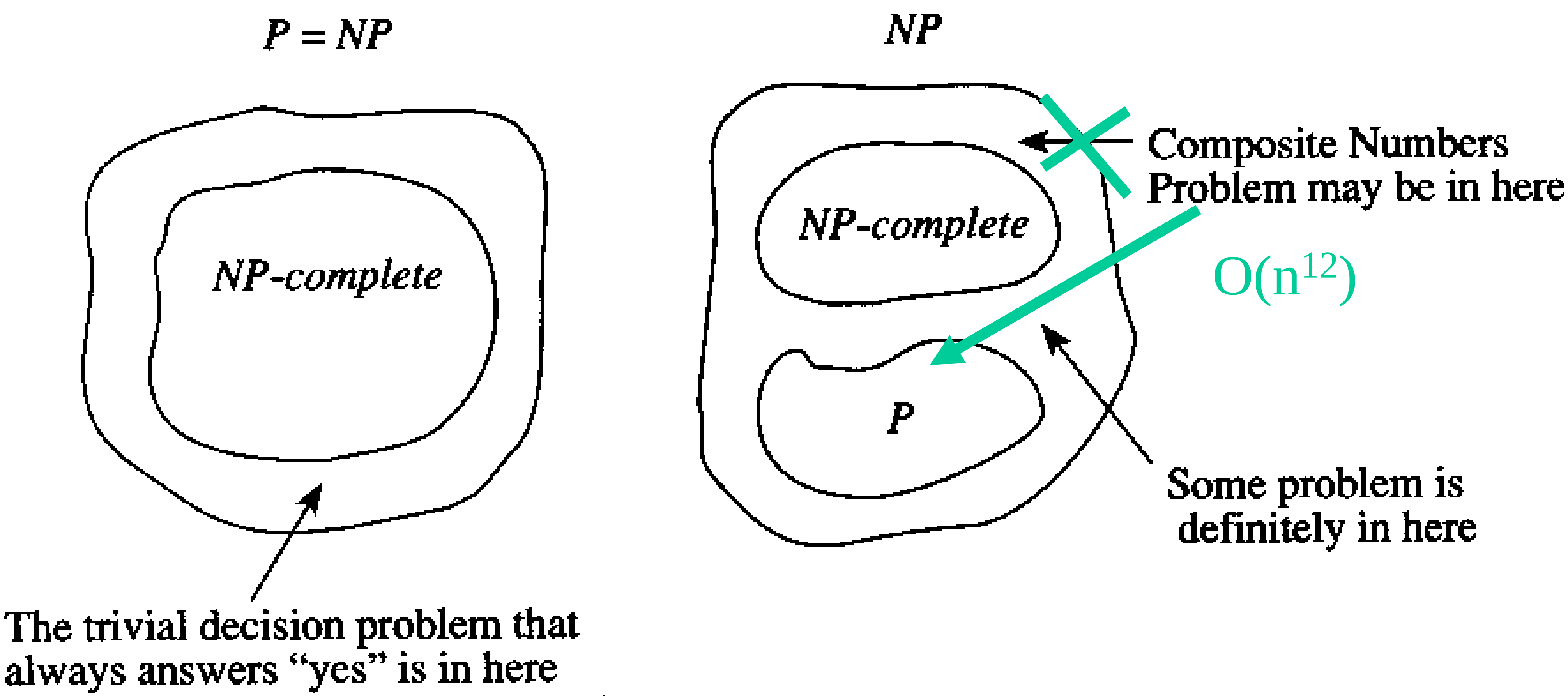
Hamiltoni tsüklite teisendamine rändkaupmehe ülesandeks

$$\text{Weight of } (u, v) \text{ equal to } \begin{cases} 1 \text{ if } (u, v) \in E \\ 2 \text{ if } (u, v) \notin E. \end{cases}$$

An example of this transformation is shown in Figure 9.6. Clearly, (V, E) has a Hamiltonian Circuit if and only if (V, E') has a tour with total weight no more than n, where n is the number of vertices in V. It is left as an exercise to complete this example by showing that the transformation is polynomial-time.



Üks kahest kehtib



Kordarvulisuse probleem

- On antud positiivne täisarv *n.* Kas eksisteerivad täisarvud $m > 1$ ja $k > 1$, nii et $n = mk$?
  - Sisendi suuruseks on *n* suurus (bittide arv)!!!
- See on NP probleem
  - Etteantud *m* ja *k* korral on kontroll klassis P

```
if( m*k == n )
```

- Keegi pole suutnud tõestada, et see on NPC
- Aastaid ei leitud polünomiaalset algoritmi
- Polünomiaalne $O(n^{12})$ algoritm leiti 2001 [Agrawal, Saxena, Kayal 2002]

Täiendülesanne

- Täiendülesanne vastab “ei” sisendile, millele ülesanne ütleb “jah” ja vastupidi.
- NP täiendülesandel on polünomiaalne testalgoritm “ei” vastuse kontrollimiseks
  - Kordarvulisuse testi täiendülesanne on algarvulisuse testi ülesanne
  - Need mõlemad ülesanded on NP
- Teoreem:
  - Kui mõne NPC ülesande täiendülesanne on NP, siis on seda ka kõigi ülejäänud NPC täiendülesanded.

NP-*hard* ülesanded

- Kehtib ka mitteotsustusülesannete kohta
- Turing teisenduse definitsioon

  Kui ülesannet A õnnestuks lahendada polünomiaalses ajas kasutades ülesande B hüpoteetilist polünomiaalset algoritmi, siis on olemas polünomiaalse ajaga Turing teisendus A-st B-ks

  $$A \propto_T B$$

- NP-*hard* definitsioon

  Ülesanne B on NP-*hard*, kui mõne NPC probleemi A kohta kehtib $A \propto_T B$

- NP-*hard* - vähemalt sama raske kui NPC
- Kui eksisteerib NP-*hard* probleem, millel on polünomiaalne algoritm, siis NP = P

NP-*easy* ja NP-*equivalent* ülesanded

- NP-*easy* definitsioon

  Ülesanne A on NP-*easy*, kui mõne NP probleemi B kohta kehtib A $\propto_T$ B,

  st A ei ole "raskem" kui B (kui NP)

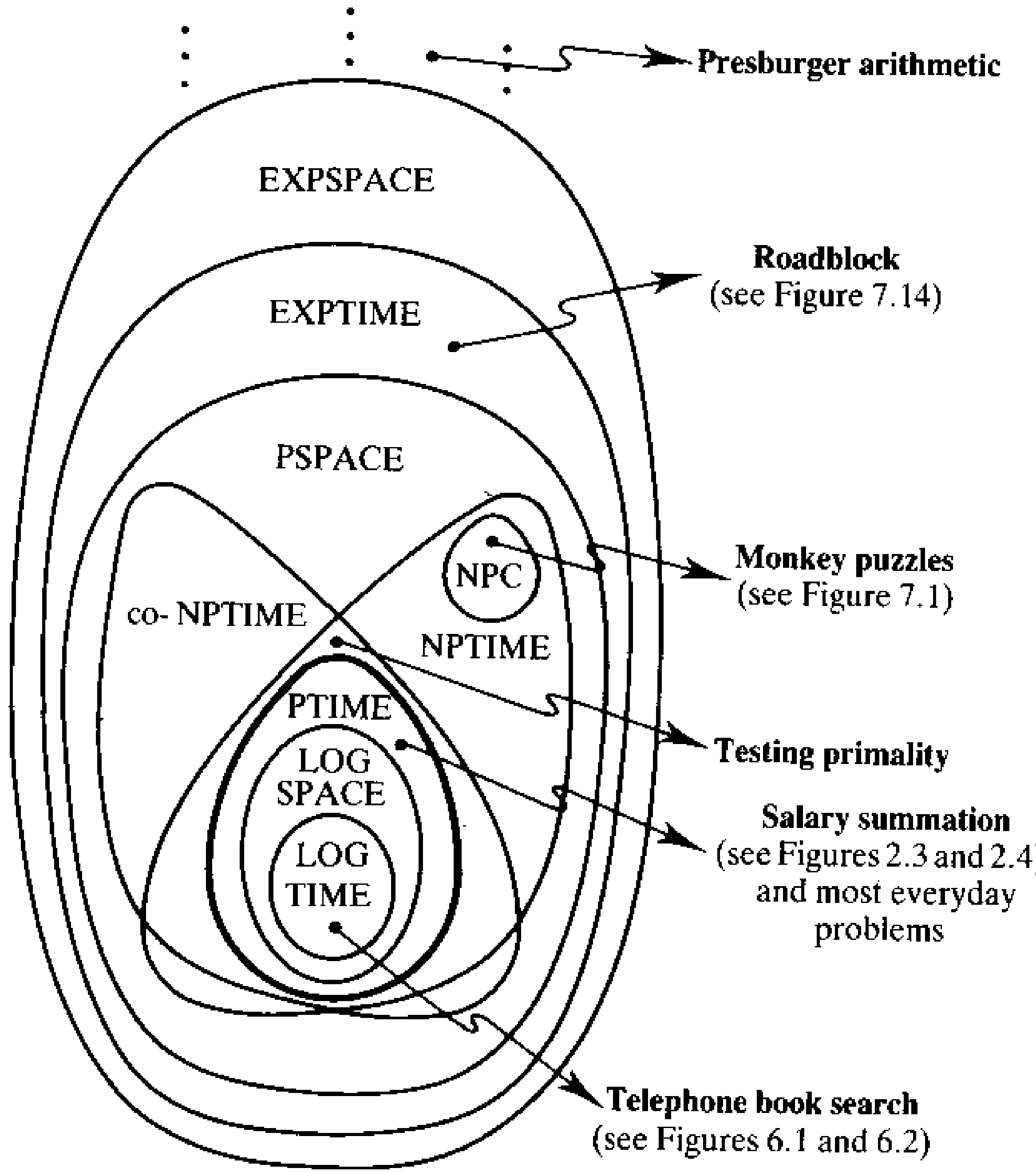
- NP-*equivalent* definitsioon

  Ülesanne A on NP-*equivalent*, kui ta on samaaegselt NP-*hard* ja NP-*easy.*

NP-*hard* ülesannete lahendamine

- Lähendav lahendamine
  - Ei anna “õiget” optimaalset vastust
  - Võib olla väga kiire
  - Võib teha iteratiivse lähendamise, mis annab parema vastuse, kui arvutamiseks on rohkem aega
- Heuristiline lahendamine
  - Halvima juhu keerukus on halb
  - Keskmise juhu keerukus võib olla küllalt hea
  - Praktikas tihti kasutatav vajalike sisendi suuruste jaoks

Klassid



Presburger arithmetic
EXPSPACE
Roadblock
(see Figure 7.14)
EXPTIME
PSPACE
Monkey puzzles
(see Figure 7.1)
NPC
co- NPTIME
NPTIME
PTIME
Testing primality
LOG
SPACE
Salary summation
(see Figures 2.3 and 2.4)
and most everyday
problems
LOG
TIME
Telephone book search
(see Figures 6.1 and 6.2)

Lahendamatud ülesanded

- Tahame näidata, et probleem A on algoritmiliselt mittelahenduv
- Meil on üks näidisprobleem B, mille mittelahenduvust oleme suutelised tõestama
  - näiteks algoritmi peatuvuse probleem
- Meil on teisendus oma probleemist A näidisprobleemi B
  - st probleemi A algoritmilise lahenduse abil saaks lahendada ka probleemi B
  - teisendus peab olema algoritmiline
  - teisendus ei pea olema polünomiaalne või mõne muu keerukuse piiranguga

Peatuvuse probleem

```
while(x != 1)
  if(even(x)) x = x/2;
  else        x = 3*x+1;
```

- keegi pole suutnud tõestada kas see algoritm peatub iga *x* korral, kuigi ta näib seda tegevat
  - on võimalik järgi proovida, et peatub iga *x* korral, kui $0 < x < 10^{10}$

Peatuvuse probleem

halting problem

- programmile Q ette antakse programm P lähtekood ja potensiaalne sisend sellele programmile P.
- Q otsustab lõpliku aja jooksul kas programm P peatub talle antud sisendi korral
- tõestame vastuväiteliselt, et sellist programmi Q ei ole olemas

Tõestus

- Loome uue 1 argumendiga programmi S(W)

```
if(Q(W,W) == "yes")
  while(true);          // infinite loop
else
  exit(0);              // terminates immediately
```

  - Q(P,I) on programm, mis otsustab lõpliku aja jooksul, kas P peatub sisendi I korral
  - Kui Q on legaalne programm, siis on seda ka S
  - Kui Q(S,S) on *yes* (S(S) peatub), siis S(S) ei peatu
  - Kui Q(S,S) on *no* (S(S) ei peatu), siis S(S) ei peatub

  $\Rightarrow$ Ei ole olemas sellist peatuvat programmi Q, mis suudaks lõplikus ajas otsustada teise programmi peatumist.