

# INF2080 – Logikk og beregninger

## Forelesning 14: Ikke beregnbart



UiO : **Institutt for informatikk**

Sist oppdatert: 2012-02-29 10:20

## 14.1 Ikke beregnbart

# Beverfunksjonen

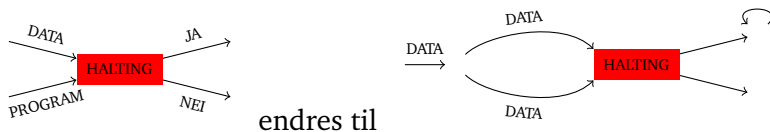
- Beverfunksjonen er ikke beregnbar
- Enhver beregnbar funksjon vokser saktere enn beverfunksjonen
- Kan ikke avgjøre om turing maskiner stopper eller ikke
- Ellers vil vi kunne beregne beverfunksjonen
- Nedenfor skal vi gi et generelt argument for dette
- Må skille mellom ekstensjonale og intensjonale egenskaper



- Beregninger har et fast antall tilstander
- Ingen begrensninger i tid eller rom

# Stoppeproblemet

- Anta at det fins maskin HALTING som løser stoppeproblemet
- Inn: PROGRAM + DATA
- Ut: JA / NEI — alltid et svar på input



- Kaller den nye maskinen  $\mathcal{T}$
- $\mathcal{T}$  anvendt på  $\mathcal{T}$  stopper
- $\Leftrightarrow$  HALTING stopper i øverste utgang
- $\Leftrightarrow \mathcal{T}$  anvendt på  $\mathcal{T}$  stopper ikke
- HALTING finnes ikke

# Totale og partielle maskiner

**Partiell** ikke krav at den stopper

**Total** maskin som alltid stopper

- Beregnbart/avgjørbart — maskinen er total og svarer JA/NEI
- Enkelt — partiell maskin som svarer JA om PROGRAM stopper
- Umulig — total maskin som svarer JA om program stopper
- Umulig — partiell maskin som svarer NEI om program stopper ikke

# Motsigelses bevis

- Start: Vi vet at det ikke fins maskin som avgjør STOPPEPROBLEMET
- Start: Anta at det fins maskin  $\mathcal{M}$  som avgjør problem  $\mathcal{P}$
- Ved å bruke kirurgi lager vi ny maskin  $\mathcal{N}$  fra  $\mathcal{M}$
- Det viser seg at  $\mathcal{N}$  løser stoppeproblemet
- Dette er umulig
- Det kan ikke finnes noen slik maskin  $\mathcal{M}$

Dette er en ganske avansert tenkemåte — motsigelses bevis. Prøv å tenke etter hvordan en kan argumentere med og konstruere maskiner som ikke finnes.

# Predikat om maskiner


**Ikke-triviell:** Av og til sant, av og til galt

**Ekstensjonalt:** Om input/output av maskin

**Intensjonalt:** Om koden til en maskin

**Avgjørbart:** Kan avgjøres av total maskin



- Ekstensjonal egenskap : Egenskap ved input / output
- Intensjonal egenskap : Egenskap ved transisjonene  utfører

# Gapet mellom ekstensjonalt og intensjonalt — 1

## Teorem 14.1

*Fins ikke noe ikke-trivielt avgjørbart ekstensjonalt predikat*

- Anta at  $\mathcal{P}$  er et slikt predikat
- La UNDEF være maskinen som aldri stopper
- Enten tilfredsstiller UNDEF  $\mathcal{P}$  eller ikke.
- Anta først at UNDEF tilfredsstiller  $\mathcal{P}$  og la  $Q$  tilfredsstillere  $\neg\mathcal{P}$
- Da kan vi løse STOPPEPROBLEMET ved å bruke  $\mathcal{P}$
- Samme argument om UNDEF tilfredsstillere  $\neg\mathcal{P}$
- Motsigelse —  $\mathcal{P}$  er ikke avgjørbart



## Gapet mellom ekstensjonalt og intensjonalt — 2

Gitt — UNDEF tilfredsstillter  $\mathcal{P}$  og  $Q$  tilfredsstillter  $\neg\mathcal{P}$ . La  $R$  være et vilkårlig program. Lag nytt program  $S$  ved

1. Lagre input til  $Q$  og input til  $R$
  2. Start  $R$  på dets input
  3. Om  $R$  stopper, så fortsett med  $Q$  på dets lagrete input
  4. Om  $R$  ikke stopper, så bare fortsett
- Om  $R$  stopper, så er  $S$  ekstensjonalt lik  $Q$
  - Om  $R$  ikke stopper, så er  $S$  ekstensjonalt lik UNDEF
  - Ved å bruke  $\mathcal{P}$  på  $S$  kan vi avgjøre om  $R$  stopper eller ikke
  - Motsigelse — vi kan ikke avgjøre  $\mathcal{P}$

Tilsvarende om UNDEF tilfredsstillter  $\neg\mathcal{P}$ . I begge tilfeller får vi at  $\mathcal{P}$  er ikke avgjørbart.

# Gapet mellom ekstensjonalt og intensjonalt — 3

- Kan ikke forutsi input/output fra kode
- Kaos — enkle program kan ha uventede konsekvenser
- STOPP er ekstensjonal egenskap
- STOPP er ikke triviell — fins en maskin som stopper og en annen maskin som ikke stopper
- STOPP kan ikke avgjøres
- Fins bare partiell maskin for STOPP — uinteressant maskin
- Tilsvarende argumenter for andre ekstensjonale egenskaper