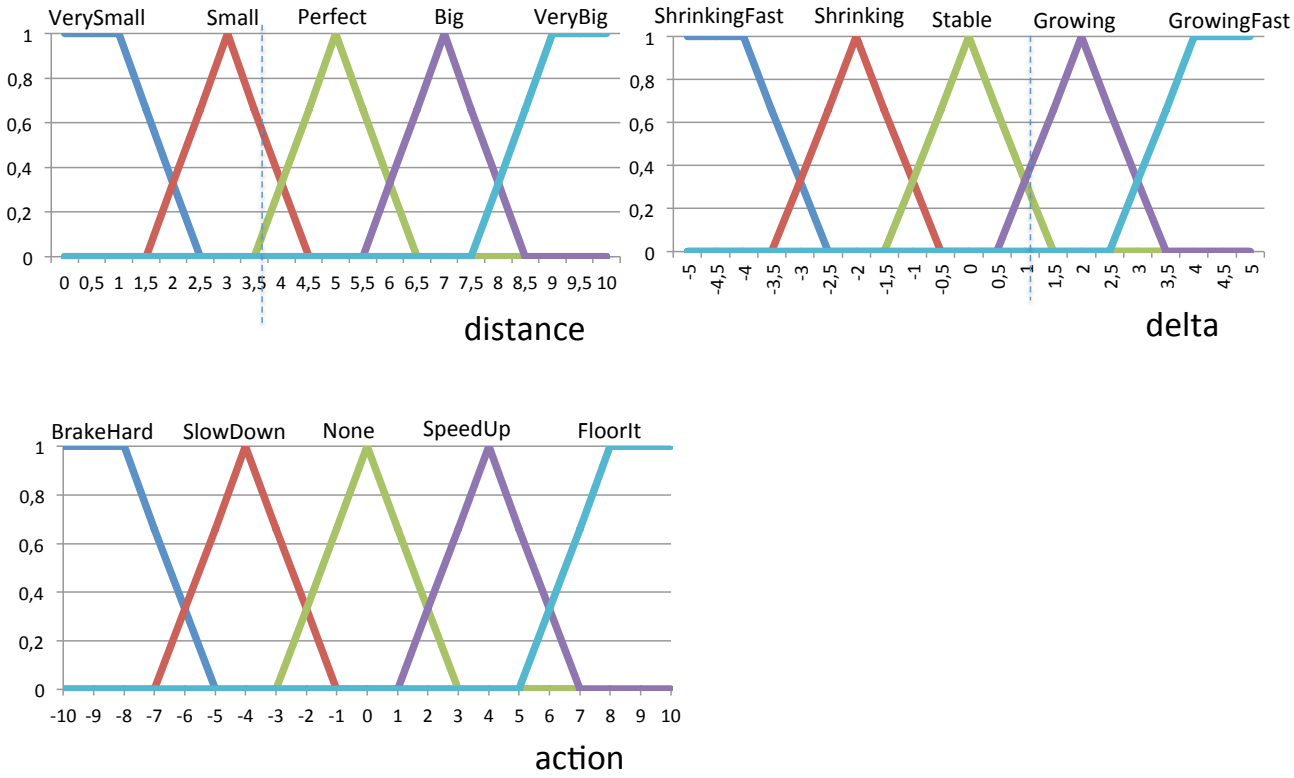
# Øving 3 Kognitive arkitekturer : Fuzzy-resonnering

Leveringsfrist : 13/10

En robotbil følger automatisk etter et kjørende objekt. Kritisk for dette er å holde en fornuftig avstand gjennom å styre akselerasjonen i forhold til objektets bevegelse. For å få en god, fleksibel og kontinuerlig kjøring, gjøres styringen av robotbilen ved hjelp av Fuzzy-resonnering. Det er to variable i roboten som avgjør responsen: **distance,** distansen til bilen foran, og **delta** , som er endringen i distanse pr. tidsenhet.. Den lingvistiske variablen **distance** har settene: VerySmall, Small, Perfect, Big, VeryBig. Variabelen **delta** opererer på settene: ShrinkingFast, Shrinking, Stable, Growing, GrowingFast, mens ut-variabelen for akselerasjon, **action,** har settene: BrakeHard, SlowDown, None, SpeedUp, FloorIt . Anta for enkelthets skyld at aksjonen bestemmes gjennom fem regler som vist under settene , se nedenfor:

# Fuzzy-sett og regler:




IF distance is Small AND delta is Growing THEN action is None

IF distance is Small AND delta is Stable THEN action is SlowDown

IF distance is Perfect AND delta is Growing THEN action is SpeedUp

IF distance is VeryBig AND (delta is NOT Growing OR delta is NOT GrowingFast) THEN action is FloorIt

IF distance is VerySmall THEN action is BrakeHard

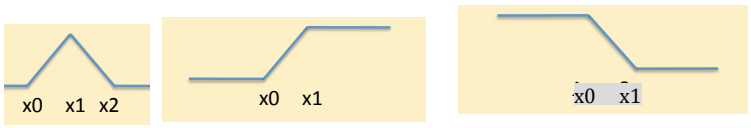
a) Gitt verdier for **distance** og **delta** som vist med de vertikale stiplete linjene i figuren (3,7 og 1,2), dvs hva slag aksjon gjør roboten med Mamdani resonnering?

Vis trinnene på hvordan du kom fram til svaret.

b) Skriv nå Mamdani-reasoner, hvor trinnene går klart fram, i et valgfritt programmeringsspråk. Du kan, for enkelthets skyld, hard-kode reglene som matematiske uttrykk (fuzzy-and, -or , -not). Du trenger også bla funksjoner/metoder for de ulike fuzzy-settene. Her er en versjon, i Java syntaks, av noen vanlige sett som du står fritt til å bruke:

```
/*
   Metodene triangle, grade og reverse_grade implementerer medlemskapsfunskjonene oppgaven viser til.
   Parameteren clip setter en øvre grense for verdien funksjonen kan returnere.
    */
```

triangle: grade: reverse grade:




```
public static double triangle(double position, double x0, double x1, double x2, double clip) {
    double value = 0.0;
    if (position >= x0 && position <= x1) value = (position - x0) / (x1 - x0);
    else if (position >= x1 && position <= x2) value = (x2 - position) / (x1 - x0);
    if (value > clip) value = clip;
    return value;
}

public static double grade(double position, double x0, double x1, double clip) {
    double value = 0.0;
    if (position >= x1) value = 1.0;
    else if (position <= x0) value = 0.0;
    else value = (position - x0) / (x1 - x0);
    if (value > clip) value = clip;
    return value;
}

public static double reverse_grade(double position, double x0, double x1, double clip) {
    double value = 0.0;
    if (position <= x0) value = 1.0;
    else if (position >= x1) value = 0.0;
    else value = (x1 - position) / (x1 - x0);
    if (value > clip) value = clip;
    return value;
}
```

Kjør situasjonen i a) og sjekk at du får riktig aksjon. Eksperimenter andre input-verdier og se hva roboten gjør.

Du vil oppdage at for enkelte verdier av **distance** og **delta** vil du få udefinerte/ugyldige output-verdier. Dette skyldes at du med vårt lille **ufullstendige** eksempelsett av regler, **ikke** dekker **alle kombinasjoner** av de to input-variablene og output. Dermed kan du risikere å få et aggregert sett hvor alt er klippet til 0, fordi alle reglene våre sender over null til høyresiden.

Når det gjelder implementasjonen, så ber vi **ikke** om å lage en animasjon med grafikk, men bare å programmere kjernen i Mamdani resonneringen.

**Lever svarene på a) og b), inklusiv koden, i en tekstfil.**