Varför är logik viktig för datavetare?

1. Datavetenskap handlar ofta om att automatisera processer som tidigare styrts av människor.
   - Intuition, intelligens och mänskliga resonemang ersätts av beräkningar.
   - Metoder att resonera exakt och på ett ”mekaniskt“ sätt krävs.
   - Men vad är ett *korrekt resonemang* och hur kan det implementeras? (Kom ihåg beräkningsbarhet och komplexitet!)
2. Några specifika tillämpningar:
   - Specifikation av programspråkssemantik
   - Korrekthet av datorsystem, program och protokoll
   - Beskrivning av hårdvarudesign (”logic design“)
   - Artificiell intelligens

En logik består av semantik och syntax

- Semantik

  Klassen av alla möjliga situationer man vill kunna resonera om. Eftersom deras antal normalt är mycket stort (ev. oändligt) är det inte möjligt att ge dem var sitt namn.

- Syntax

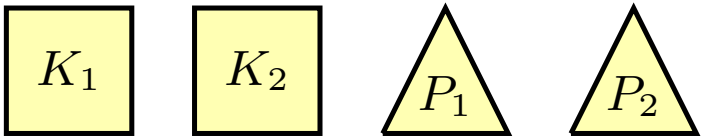
  Ett språk för att beskriva egenskaper av möjliga situationer, tillsammans med en uppsättning "mekaniska" regler som beskriver tillåtna resonemang.

Utan att förstå den här skillnaden är det omöjligt att förstå hur den formella logiken fungerar.

Syntax är något datorer kan handskas med medan semantiken är det vi egentligen är ute efter. Varje logik (det finns många!) består av båda komponenter – samt exakta regler som kopplar dem samman.

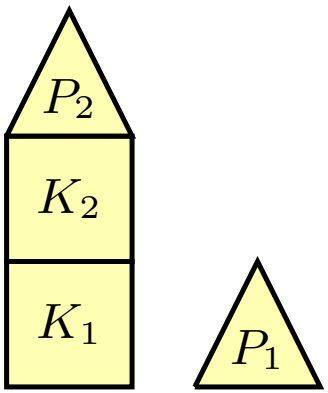
Blockvärldar – ett exempel

Vi börjar med semantiken: Vi vill kunna prata om blockvärldar. Varje sådan är ett arrangemang av två kuber och två pyramider på ett bord.

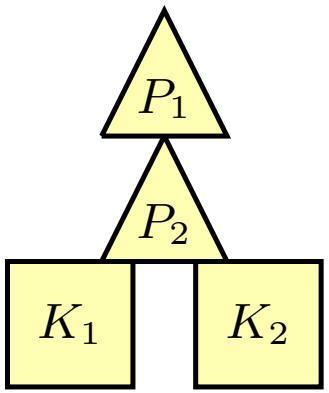


Villkor

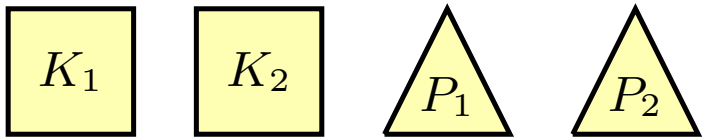
- Kuberna får staplas på varandra; en pyramid får placeras på en kub.
- Objekt får endast placeras mitt på varandra.

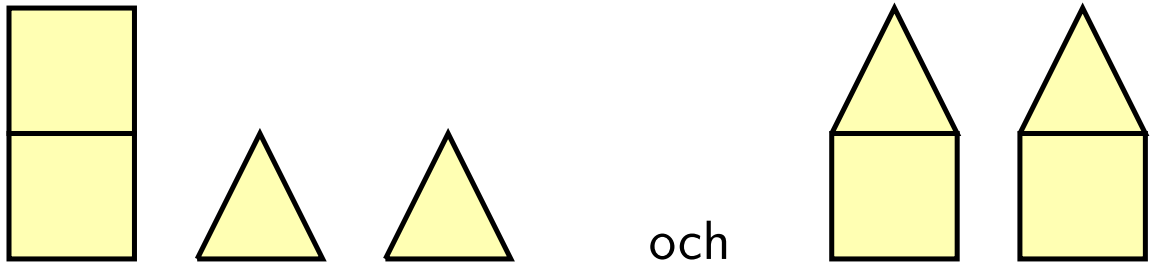
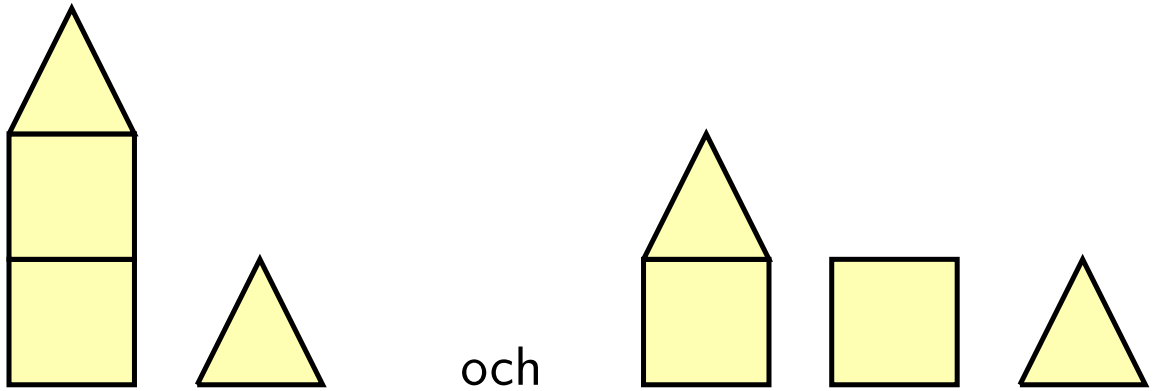


är okej men inte



Vi intresseras endast av att veta vilket objekt som ligger på vilket, dvs den exakta placeringen spelar ingen roll. Det ger 13 möjliga världar:

- Världen 

- 2×2 världar  och
- 2×4 världar  och

Syntaxen vi använder för att beskriva egenskaper av blockvärldar:

Atomära satser (där $x, y \in \{K_1, K_2, P_1, P_2\}$)	Koppling till semantiken
On_table(x)	objekt x ligger (direkt) på bordet
On(x, y)	objekt x ligger (direkt) på objekt y
Is_cube(x)	objekt x är en kub
Is_pyramid(x)	objekt x är en pyramid

Exempel: Is_cube(P_1) och On(K_2, P_1) är falska i alla möjliga världar, Is_cube(K_1) är sant i alla och On(P_1, K_1) samt On_table(P_2) är sanna i vissa.

Syntaxen utökas med logiska konnektiv, symboler som får användas för att kombinera satser:

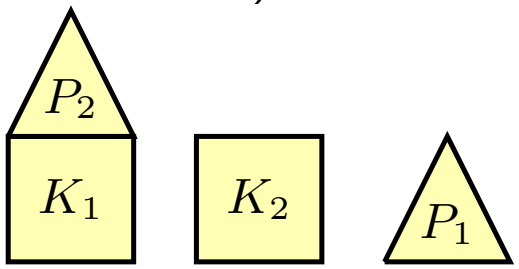
Symbol	Rank	Koppling till semantiken
$\wedge$	2	logisk konjunktion (”och“)
$\vee$	2	logisk disjunktion (”eller“)
$\neg$	1	logisk negation (”icke“)
$\rightarrow$	2	logisk implikation (”implicerar“)
$\leftrightarrow$, $\equiv$	2	logisk ekvivalens (”omm“, ”om och endast om“)

Precedensregler och parenteser används för att undvika flertydighet. Korrekt byggda satser kallas välformade formler (eng. well-formed formula, kort wff).

Exempel: $\mathsf{On\_table}(K_1) \wedge \neg \mathsf{On}(P_1, K_2)$ uttrycker att K_1 ligger på bordet och P_1 inte ligger på K_2.

Några observationer

- En blockvärld kan entydigt beskrivas av mängden av alla atomära satser (alltså satser utan logiska konnektiv) som är sanna i världen.

Till exempel beskrivs 

av

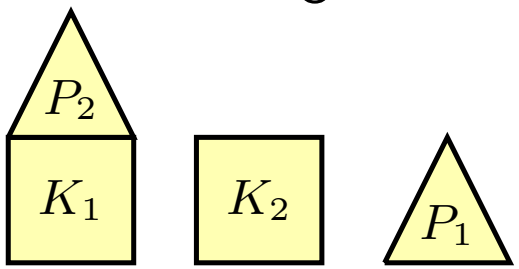
$$\{\ \mathsf{Is\_cube}(K_1), \mathsf{Is\_cube}(K_2), \mathsf{Is\_pyramid}(P_1), \mathsf{Is\_pyramid}(P_2),$$
$$\mathsf{On\_table}(K_1), \mathsf{On\_table}(K_2), \mathsf{On\_table}(P_1), \mathsf{On}(P_2, K_1)\ \},$$

som motsvarar wff:en

$$\mathsf{Is\_cube}(K_1) \wedge \mathsf{Is\_cube}(K_2) \wedge \mathsf{Is\_pyramid}(P_1) \wedge \mathsf{Is\_pyramid}(P_2) \wedge$$
$$\mathsf{On\_table}(K_1) \wedge \mathsf{On\_table}(K_2) \wedge \mathsf{On\_table}(P_1) \wedge \mathsf{On}(P_2, K_1).$$

- I exemplet ovan räcker även $\mathsf{On\_table}(K_1) \wedge \mathsf{On\_table}(P_1) \wedge \mathsf{On}(P_2, K_1)$.

- OBS! Det här beror på logikens utformning. Om vi t ex stryker de atomära formlerna On_table(x) kan fortfarande alla världar beskrivas mha wff:er men inte längre nödvändigtvis av mängder av atomära satser. T ex beskrivs



av $\mathsf{On}(P_2, K_1) \wedge \neg\mathsf{On}(P_1, K_2) \wedge \neg\mathsf{On}(K_1, K_2)$.

- I sådana fall måste alltså både de atomära satser som är sanna och de som är falska specificeras.

Att resonera över blockvärldar

Hittills har vi

- en mängd av möjliga världar och
- ett språk som tillåter att uttrycka egenskaper hos de här världarna.

Det som ännu fattas är möjligheten att resonera över världarnas egenskaper.
Kort sagt: Är ett påstående sant i alla världar eller inte?

Exempel: $\mathsf{On\_table}(K_1) \vee \mathsf{On}(K_1, K_2)$ (K_1 ligger på bordet eller på K_2).

- För att se om det här alltid är sant måste alla 13 möjliga världar prövas.
- I mindre triviala situationer är mängden av världar för stor. Dessutom är de möjliga världar ev. i sin tur beskrivna på ett logiskt sätt.
- Vad som behövs är en inferensmekanism som ger ett ja/nej-svar beroende på om satsen är sann eller inte.

I kursen kommer två logiktyper att presenteras:

- Satslogik
  - Formler består av konnektiv och oparametriserade atomära formler (som i blockvärldsexemplet).
  - Rätt enkelt att förstå, inga problem med oavgörbarhet.
  - Begränsad uttryckskraft, ibland onödigt krångliga formler.
- Första ordningens predikatlogik
  - Variabler och kvantifierare (för alla $\forall$, det finns $\exists$).
  - Ganska kraftfull och adekvat i många situationer.
  - Svårare att lära sig.
  - Oavgörbara problem (t ex inferens).