Grunder i matematik och logik (2021)

Logik och mängder

Marco Kuhlmann och Victor Lagerkvist

Satslogik

Satser och konnektiv

1.01 En **sats** (eng. *proposition*) är ett påstående som är antingen sant eller falskt. Exempel på satser är ”2 är ett jämnt tal” (sant) och ”2 är ett udda tal” (falskt). Satser betecknas ofta med bokstäver såsom p, q och r.

1.02 De två sanningsvärdena sant och falskt kan skrivas på olika sätt. Exempel: sant, S, T (eng. *true*), $\top$, 1; falskt, F (eng. *false*), $\bot$, 0. I den här kursen använder vi 1 och 0.

1.03 En sats är antingen *atomär* eller *sammansatt*. En atomär sats består av antingen värdet 1 (sant), värdet 0 (falskt) eller en satsvariabel. En sammansatt sats är konstruerad av en eller flera delsatser med hjälp av ihopkopplande ord som kallas **konnektiv**. Ett exempel på en sammansatt sats är ”Jag är student *och* jag gillar logik”.

1.04 De flesta programspråk innehåller syntax för sanningsvärden och konnektiv. I Python betecknas sanningsvärdena med de reserverade orden `True` och `False`, och det finns tre operatorer som uttrycker konnektiv: `not`, `and` och `or`.

1.05 **Negationen** av en sats p är sann om p är falsk; annars är den falsk. Negationen av en sats p skrivs $\neg p$ vilket utläses ”inte p”. I Python uttrycker man negation med hjälp av operatorn `not`. Negation är en unär operator, den har alltså endast ett invärde.

1.06 **Konjunktionen** av två satser p och q är sann om både p och q är sanna; annars är den falsk. Konjunktion av två satser p och q skrivs $p \wedge q$ vilket utläses ”p och q”. I Python uttrycker man konjunktion med hjälp av operatorn `and`.

1.07 **Disjunktionen** av två satser p och q är sann om minst en av p och q är sann; annars är den falsk. Disjunktionen av två satser p och q skrivs $p \vee q$ vilket utläses ”p eller q”.[1] I Python uttrycker man konjunktion med hjälp av operatorn `or`.

[1] Notera att detta är samma ”eller” som i ”mjölk eller socker”, inte som i ”död eller levande”.

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

(a) konjunktion

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

(b) disjunktion

Figur 1: Sanningsvärdestabeller för konjunktion och disjunktion.

Sanningsvärdestabeller

1.08 För att analysera en sats kan man sätta upp en **sanningsvärdestabell** för den. I en sådan tabell tilldelar man de satsvariabler som förekommer i satsen alla möjliga kombinationer av värden och räknar sedan stegvis ut sanningsvärdet för hela satsen.

1.09 Sanningsvärdestabellerna för konjunktion och disjunktion visas i figur 1. Sanningsvärdestabeller för två nya operatorer visas i figur 2.

1.10 En **implikation** $p \to q$ (utläses "p implicerar q") är falsk endast om p är sann och q är falsk; annars är den sann. Sanningsvärdestabellen för implikation visas i figur 2a. Implikation har ingen motsvarande operator i Python.

1.11 En **ekvivalens** $p \leftrightarrow q$ (utläses "p ekvivalent q") är sann om p och q har samma sanningsvärde; annars är den falsk. Sanningsvärdestabellen för ekvivalens visas i figur 2b. I Python kan ekvivalens uttryckas med hjälp av likhetstestet (`==`).

1.12 Figur 3 visar sanningsvärdestabellen för den sammansatta satsen

$$\big(p \vee (\neg q)\big) \to \big((\neg r) \wedge q\big)\,.$$

Satsen innehåller tre olika satsvariabler och fem konnektiv; detta ger sammanlagt åtta kolumner i tabellen. Att variablerna är tre ger också att antalet rader i tabellen blir åtta. (Exemplet är taget från Eriksson och Gavel 2005, s. 188.)

p	q	$p \to q$
0	0	1
0	1	1
1	0	0
1	1	1

(a) implikation

p	q	$p \leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

(b) ekvivalens

Figur 2: Sanningsvärdestabeller för implikation och ekvivalens.

p	q	r	$\neg q$	$p \vee (\neg q)$	$\neg r$	$(\neg r) \wedge q$	$(p \vee (\neg q)) \rightarrow ((\neg r) \wedge q)$
0	0	0	1	1	1	0	0
0	0	1	1	1	0	0	0
0	1	0	0	0	1	1	1
0	1	1	0	0	0	0	1
1	0	0	1	1	1	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	1	1	1
1	1	1	0	1	0	0	0

Figur 3: Sanningsvärdestabellen för en sammansatt sats.

Satisfierbarhet och ekvivalens

1.13 Varje rad i sanningsvärdestabellen för en sats p representerar en **tilldelning** av sanningsvärden till de satsvariabler som förekommer i p. Givet en sådan tilldelning kan man läsa av värdet hos p från den sista kolumnen.

1.14 En sats kallas **satisfierbar** om det finns minst en tilldelning av sanningsvärden till satsvariablerna (alltså en rad i sanningsvärdestabellen) med vilken satsen får värdet 1 (sant). Ett exempel på en satisfierbar sats är $p \wedge q$, som får värdet 1 med tilldelningen $p = 1$, $q = 1$. Även satsen i figur 3 är satisfierbar, som sanningstabellen visar.

1.15 För att avgöra om en sats p är satisfierbar kan man sätta upp sanningsvärdestabellen för den och testa om den sista kolumnen innehåller värdet 1. Detta kan ta lång tid om antalet variabler är stort, men man känner idag inte till någon mer effektiv algoritm (förutom för vissa specialfall).[2]

1.16 Hur många tilldelningar (rader i sanningsvärdestabellen) behöver man i värsta fall gå igenom för att testa om en sats med n stycken satsvariabler är satisfierbar?

Svar: 2^n många tilldelningar

1.17 Två satser p och q är **logiskt ekvivalenta** om de med samma tilldelning alltid ger samma värde. Ett exempel på två ekvivalenta satser är $p \rightarrow q$ och $\neg p \vee q$.

1.18 För att avgöra om två satser p och q är logiskt ekvivalenta kan man jämföra de sista kolumnerna i deras respektive sanningstabeller; dessa bör vara identiska.

Tautologier och kontradiktioner

1.19 En **tautologi** är en sats vars värde alltid är 1, oavsett tilldelning; en **kontradiktion** är en sats vars värde alltid är 0, oavsett tilldelning. Ett exempel på en tautologi är $p \vee \neg p$; ett exempel på en kontradiktion är $p \wedge \neg p$.

[2] Problemet att avgöra om en sats är satisfierbar är ett så kallat NP-hårt problem.

1.20 För att avgöra om en sats p är en tautologi eller en kontradiktion kan man på samma sätt som när man testar om den är satisfierbar sätta upp en sanningsvärdestabell för den. Ifall p är en tautologi kommer då hela sista kolumnen innehålla ettor; ifall p är en kontradiktion kommer den endast innehålla nollor.

Predikatlogik

1.21 Ett **predikat** är en sats om ett objekt som betecknas med en variabel. I motsats till en sats är sanningsvärdet för ett predikat beroende på vilket konkret objekt som sätts in för variabeln. Predikatet "$x \leq 2$" till exempel är sant för $x = 1$ men falskt för $x = 3$.

1.22 I programspråk förekommer predikat t.ex. i test och i form av funktioner som returnerar ett sanningsvärde. Följande exempel definierar ett predikat `is_even` som representerar påståendet "x är ett jämnt tal".

```
def is_even(x):
    return x % 2 == 0
```

1.23 I samband med predikat används två **kvantorer**: allkvantorn $\forall$, som står för "för alla", "för varje" samt existenskvantorn $\exists$ som står för "det finns", "det existerar". Exempel på satser formulerade med dessa kvantorer:

$$\forall x(x \text{ är student} \wedge x \text{ gillar logik}) \qquad \exists x(x \text{ är student} \wedge x \text{ gillar logik})$$

1.24 I Python kan man uttrycka allkvantorn och existenskvantorn med hjälp av de inbyggda funktionerna `all` och `any`, som testar om ett predikat ger värdet 1 respektive 0 för alla element i en samling:

```
all(is_even(x) for x in range(100))    # False
any(is_even(x) for x in range(100))    # True
```

Mängdlära

Mängder och element

1.25 En **mängd** (eng. *set*) är en samling objekt. Två standardexempel är mängden av naturliga tal ($\mathbb{N}$) och mängden av heltal ($\mathbb{Z}$); objekten i en mängd behöver dock inte vara tal. De objekt som ingår i en mängd kallas för mängdens **element**.

1.26 Det är vanligt att använda sig av stora bokstäver för att beteckna mängder, och av små bokstäver för att beteckna element. (Vi kommer dock in på mängder som har mängder som element ganska snart.)

1.27 Man skriver $a \in A$ som en förkortning för "objektet a tillhör mängden A" och $a \notin A$ som en förkortning för "objektet a tillhör *inte* mängden A". Till exempel gäller att $2 \in \mathbb{N}$ (2 är ett naturligt tal) och $\sqrt{2} \notin \mathbb{N}$ (kvadratroten ur 2 är inte ett naturligt tal).

1.28 I Python finns en inbyggd datatyp för mängder som heter `set`. Elementsymbolen $\in$ har sin motsvarighet i det reserverade ordet `in`.

```
a = set([1, 2, 3])
1 in a    # True
5 in a    # False
```

Hur man beskriver mängder

1.29 Man kan beskriva en mängd genom att räkna upp dess element mellan klammerparenteser (”måsvingar”). Här kommer, som exempel, mängden av sanningsvärden, mängden av alla naturliga tal och mängden av alla medlemmar i *Beatles* år 1964.

$$\mathbb{B} = \{\text{sant}, \text{falskt}\} \qquad \mathbb{N} = \{0, 1, 2, \dots\} \qquad B = \{\text{John}, \text{Paul}, \text{George}, \text{Ringo}\}$$

1.30 Man kan också beskriva en mängd genom att specificera vilken egenskap som utmärker de element som tillhör mängden. Exempel:

$$B = \{\, x \mid x \text{ var en medlem i } \textit{Beatles} \text{ år } 1964 \,\}$$

Detta ska läsas som: ”Mängden B är mängden av alla x sådana att x var en medlem i *Beatles* år 1964.” Denna notation kallas för **mängdbyggare**.

Likhet mellan mängder

1.31 Två mängder A och B räknas som lika ($A = B$) om och endast om de innehåller exakt samma element. Detta innebär att vi inte tar hänsyn till hur vi beskriver en mängd; det enda som är viktigt är vilka element som ingår i mängden.

1.32 Ersätt symbolen $\bigcirc$ med antingen $=$ eller $\neq$ så att påståendena blir sanna:

a) $\{1, 2, 3\} \bigcirc \{3, 2, 1\}$

b) $\{1, 1\} \bigcirc \{1\}$

c) $\{\, 2n \mid n \in \mathbb{N} \,\} \bigcirc \mathbb{N}$

d) $\{\, |n| \mid n \in \mathbb{Z} \,\} \bigcirc \mathbb{N}$

Svar:

a) $\{1, 2, 3\} = \{3, 2, 1\}$

b) $\{1, 1\} = \{1\}$

c) $\{\, 2n \mid n \in \mathbb{N} \,\} \neq \mathbb{N}$

d) $\{\, |n| \mid n \in \mathbb{Z} \,\} = \mathbb{N}$

Grundmängd

1.33 Den mängd som innehåller alla element som är relevanta i ett visst sammanhang kallas **grundmängd**. Om vi t.ex. arbetar med naturliga tal, så är det lämpligt att välja $\mathbb{N}$ som grundmängd. Grundmängden betecknas ofta med bokstaven U (”universum”).

1.34 Använder man en mängdbyggare så kan man skriva grundmängden till vänster om det lodrätta strecket när man definierar nya mängder. Exempel:

$$\mathbb{J} \;=\; \text{mängden av alla jämna naturliga tal} \;=\; \{\, n \in \mathbb{N} \mid \; n \text{ är jämt} \,\}$$

Kardinalitet

1.35 Storleken hos en mängd A kallas för A:s **kardinalitet** och skrivs $|A|$. För ändligt stora mängder anger $|A|$ antalet element i A. Om vi som exempel tar $A = \{1, 3, 5\}$, så innehåller denna mängd tre element och vi skriver $|A| = 3$.

Den tomma mängden

1.36 Den **tomma mängden** är mängden som inte innehåller några element och skrivs $\emptyset$. Dess kardinalitet är 0: $|\emptyset| = 0$.

1.37 Vilka av följande mängder är den tomma mängden?

a) $A = \{\, n \mid \; n \in \mathbb{N}, n + 2 = 0 \,\}$

b) $B = \{\, n \mid \; n \in \mathbb{Z}, n + 2 = 0 \,\}$

c) $C = \{\, n \mid \; n \in \mathbb{Q}, n^2 = 5 \,\}$

d) $D = \{\, n \mid \; n \in \mathbb{R}, n^2 = 5 \,\}$

Svar:

a) $A = \emptyset$

b) $B = \{-2\}$

c) $C = \emptyset$

d) $B = \{-\sqrt{5}, +\sqrt{5}\}$

1.38 Ett vanligt misstag är att blanda ihop $\emptyset$ (den tomma mängden) och $\{\emptyset\}$ (mängden som innehåller den tomma mängden som sitt enda element). Observera:

$$\emptyset \in \emptyset \text{ är falskt} \qquad \emptyset \in \{\emptyset\} \text{ är sant}$$

Delmängdsrelationen

1.39 En mängd A är en **delmängd** till en mängd B om och endast om varje element i A är ett element i B. Som ett exempel kan vi ta $A = \{1, 3\}$ och $B = \{1, 3, 5\}$. Då är A delmängd till B eftersom båda elementen i A även finns i B; däremot är B inte delmängd till A eftersom 5 ingår i B men inte i A.

1.40 Vi skriver $A \subseteq B$ för att uttrycka att A är en delmängd till B och $A \nsubseteq B$ för att uttrycka att A *inte* är en delmängd till B. Med exempelmängderna $A = \{1, 3\}$ och $B = \{1, 3, 5\}$ gäller alltså $A \subseteq B$ och $B \nsubseteq A$.

1.41 Ange alla delmängder till $A = \{\text{svart}, \text{vit}\}$.

Svar: Vi måste hitta alla möjliga mängder där varje element också ingår i mängden A. Det finns tre sådana mängder som innehåller minst ett element: $\{\text{svart}\}$, $\{\text{vit}\}$ och $\{\text{svart}, \text{vit}\}$. Dessutom så är den tomma mängden $\emptyset$ en delmängd till A. Därmed finns det alltså totalt fyra delmängder till A.

1.42 **Potensmängden** till en mängd A är mängden som består av alla delmängder till A. Potensmängden till A skrivs ofta $\mathcal{P}(A)$.

1.43 Vad är potensmängden till den tomma mängden?

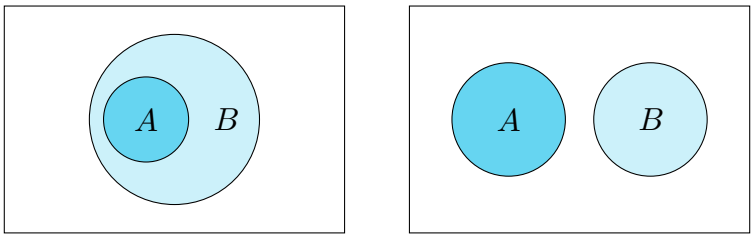
Svar: $\{\emptyset\}$

1.44 Om A är delmängd till B, men B innehåller även element som inte är element i A, så säger man att A är en **äkta delmängd** till B och skriver $A \subset B$. (Notera parallellerna mellan symbolparet $\subseteq, \subset$ för mängder och $\leq, <$ för tal.)

1.45 Delmängdsrelationen kan illustreras med hjälp av ett **venndiagram** (efter John Venn, 1834–1923); se figur 4a. I dessa diagram representeras mängder genom cirklar. Cirklarnas placering representerar relationerna mellan mängderna.

Disjunkthetsrelationen

1.46 Två mängder A och B är **disjunkta** om de inte har några gemensamma element. Ibland ser man notationen $A \parallel B$ som förkortning för ”A och B är disjunkta”. Disjunkthetsrelationen illustreras med hjälp av venndiagrammet i figur 4b.



(a) $A \subseteq B$ (b) $A \parallel B$

Figur 4: Två relationer mellan mängder: ”är delmängd till” och ”är disjunkt till”.

Mängdoperationer

1.47 När vi räknar med mängder har vi operationerna union, snitt, differens och komplement. Dessa operationer illustreras genom venndiagrammen i figur 5.

1.48 **Unionen** mellan två mängder A och B är den mängd som består av alla element som ingår i A, B eller båda mängderna. Unionen mellan A och B betecknas $A \cup B$. Med hjälp av symbolen för logisk disjunktion kan vi skriva:

$$A \cup B \;=\; \{\, x \mid x \in A \vee x \in B \,\}$$

1.49 **Snittet** mellan två mängder A och B är den mängd som består av alla element som ingår i båda mängderna. Snittet mellan A och B betecknas $A \cap B$. Med hjälp av symbolen för logisk konjunktion kan vi skriva:

$$A \cap B \;=\; \{\, x \mid x \in A \wedge x \in B \,\}$$

1.50 **Differensen** mellan två mängder A och B är den mängd som består av alla element som ingår i A men inte i B. Differensen mellan A och B betecknas $A \backslash B$. Med hjälp av en mängdbyggare kan vi skriva:

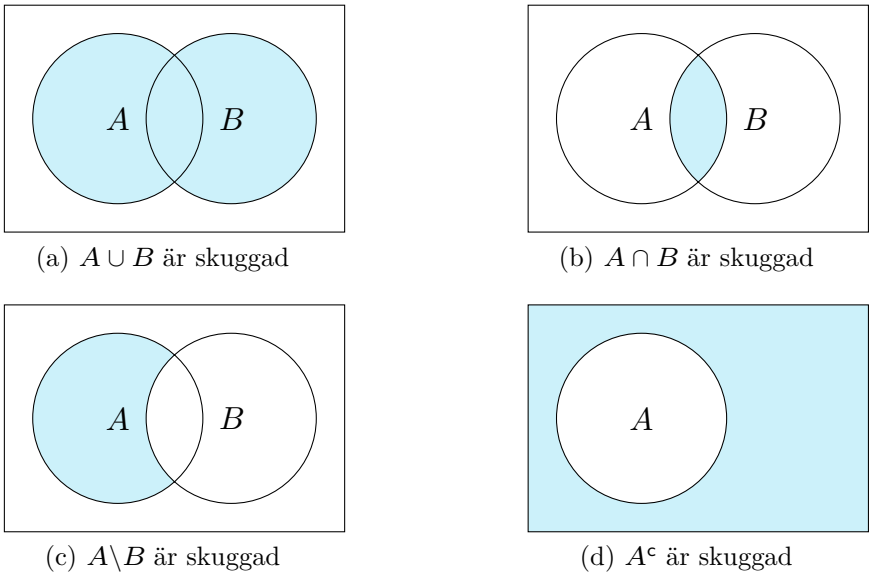
$$A \backslash B \;=\; \{\, x \mid x \in A \wedge x \notin B \,\}$$

1.51 **Komplementet** till en mängd A är den mängd som består av alla element som ingår i grundmängden $\mathcal{U}$ men inte i A. Komplementet till A betecknas A^{c}. Med hjälp av en mängdbyggare kan vi skriva:

$$A^{\mathsf{c}} \;=\; \{\, x \mid x \in \mathcal{U} \wedge x \notin A \,\} \;=\; \mathcal{U} \backslash A$$

1.52 I Python ser mängdoperationerna union, snitt och differens ut så här:

```
a | b    # union
a & b    # snitt
a - b    # differens
```



(a) $A \cup B$ är skuggad

(b) $A \cap B$ är skuggad

(c) $A \backslash B$ är skuggad

(d) A^{c} är skuggad

Figur 5: Mängdoperationer