UPPSALA UNIVERSITET

Föreläsningsanteckningar VT22

Logik och Bevisteknik

Rami Abou Zahra

Contents

1. Föreläsning - Introduktion till Satslogik 2
1.1. Historia 2
1.2. Vad behövs för ett matematiskt bevis? 2
1.3. Exempel 2
1.4. Satslogik 3
1.5. Predikatlogik (1:a ordningens logik) 3
2. Satslogik (Propositional calculus) 4
2.1. Språk (LP) 4
2.2. Exempel 4
3. Föreläsning - Naturlig deduktion (syntax) 5
3.1. Parantesers roll 5
3.2. Syntax 5
3.3. Naturlig deduktion 6
4. Förtydligande 7
4.1. Sanningstabeller 8
5. Föreläsning - Naturlig deduktion forts. 9
5.1. Botten ($\bot$) 9
6. Semantik för satslogik 10
7. Logisk ekvivalens och konsekvens 12
7.1. Logisk ekvivalens 12
7.2. Exempel 12
7.3. Några viktiga ekvivalenser 12
7.4. Logisk konsekvens 13
7.5. Exempel 13
7.6. Exempel 13
7.7. Algoritm för att avgöra $\Gamma \vDash \varphi$ 13
7.8. Exempel 13
7.9. Exempel 14
7.10. Kommentar 14
8. Substitution 15
8.1. Exempel 15
8.2. Normalformer 16
9. Funktionellt kompletta mängder av konnektiv 18
9.1. Användning av sundhetssatsen 19
10. Dugga 2021-04-13 21
11. Sammanfattning - Kapitel 2-5 22
11.1. Kapitel 2 22
11.2. Kapitel 3 - Semantik 23
12. Predikatlogik - Första ordningens logik 24
12.1. Relationer 24
12.2. Funktioner 25
12.3. Strukturer 25
12.4. Språk för predikatlogik (1:a ordningens logik) 25
13. Predikatlogik - Forts. 27
13.1. Fria och bundna förekomster av variabler 27
13.2. Variabelsubstitution 27

1. Föreläsning - Introduktion till Satslogik

1.1. **Historia.** Vad är ett matematiskt bevis?
Vad får användas i bevis?
Är matematiken motsägelsefri (Konsistent = motsägelsefritt)?

1.2. **Vad behövs för ett matematiskt bevis?**

- Ett påstående (även kallad utsaga) (ex.vis $\sqrt{2}$ är irrationellt), dessa har sanningsvärde *sant* eller *falskt*
- (Giltigt) Argument (resonemang)

1.3. **Exempel.**

- Påstående: *Varje kvadrat är en rektangel*
- Påstående: *Det finns en fyrhörning som inte är en rektangel*

Alltså finns en fyrhörning som inte är kvadrat

Detta är syftet med kursen, låt oss nu göra det mer abstrakt. (Låt x vara form, K(x) kvadrat, R(x) rektangel, F(x) fyrhörning) Då blir påståenden:

- $\forall x(K(x) \Rightarrow R(x))$
- $\exists x(F(x) \wedge \neg R(x))$

$\exists x(F(x) \wedge \neg(K(x)))$

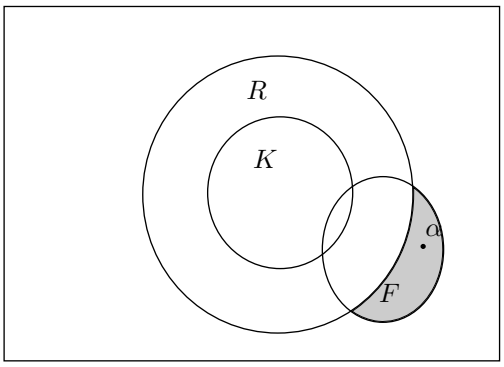

Figure 1. Grafisk tolkning

Sats 1.1: Logiskt giltighet (Predikat-logik = 1:a ordningens logik)

En följd av logiska steg kallas för *logiskt giltig* om det gäller $\forall$ tolkningar av K, R, F (även vovvisar)

1.4. **Satslogik.** Vi behöver:

- Ett skriftligt språk (mängd av teckensträngar som betyder utsagor/satser)
- Regler/Formella bevis (Hur får man dra slutsatser av nya teckensträngar), *syntax*
- Tolka teckensträngarna i en ”verklighet” (sant eller falskt)

1.5. **Predikatlogik (1:a ordningens logik).** Skiljer sig från satslogik i och med att vi inte hanterar satser utan predikaten.

- Språk behövs, liknande med teckenmängd men vi kan även hantera elementen
- Formella bevis består av teckensträngsmanipulation, relation mellan teckensträngar
  (Exvis $A_1, A_2, A_3 \vdash B$ (A bevisar B))
- Vad betyder det att något är sant eller falskt? Om det går att visa B utan något så är det sant.

Sats 1.2: Definiera sanning i struktur

$\vdash B =$ Sant om det inte krävs något för att visa B:s sanningsvärde.

$A_1, A_2, A_3 \vDash B$ det vill säga om alla A krav är uppfyllda så gäller B

- Samband mellan $\vdash$ (formell bevisbarhet) och $\vDash$ (hur man tolkar något som sant eller falskt)

Sats 1.3: Sundhetssatsen

Detta säger att bevissystemet är sunt, allt man visar är sunt

$A_1, A_2, A_3 \vdash B \Rightarrow A_1, A_2, A_3 \vDash B$

Sats 1.4: Adekvathet

Om det är så att B är sann så fort alla A är sanna så finns det ett bevis, vi kommer kunna påstå att det finns ett formellt bevis för samma sak.

$A_1, A_2, A_3 \vDash B \Rightarrow A_1, A_2, A_3 \vdash B$

Sats 1.5: Fullständighet

Slår vi ihop dessa 2 (Sundhetssatsen och Adekvathet) får vi fullständighet

$A_1, A_2, A_3 \vdash B \Leftrightarrow A_1, A_2, A_3 \vDash B$

2. Satslogik (Propositional calculus)

2.1. **Språk (LP).**

- Satssymboler: $\sigma = \{p_0, p_1 \cdots, p_n\}$ (en *satslogisk signatur*, även kallas *språkets signatur*)

  LP(σ) kallas för en *dialekt* av LP
- Alfabet i LP(σ):
  - Alla symboler i σ
  - Konnektiver: $\Leftrightarrow, \Leftarrow, \Rightarrow, \vee, \wedge, \neg, \bot$
  - Paranteser: (,)

Viktigt att notera, $\bot$ är konnektiv men även sats, så den skapar inga nya formler.

Sats 2.1: Mängden av formler i LP(σ)

Mängden definieras rekursivt (likt naturliga talen).

- Basfall:
  - Alla tecken i σ är en formel
  - $\bot$ är en formel
  - Dessa kallas för *Atomära* formler
- Induktion:
  - Om φ är en formel, så är $(\neg\varphi)$ en formel
- Låt $\square$ vara konnektiv:
  - Induktion $\square$:
    - Om φ_1 och φ_2 är formler, så är $(\varphi_1 \square \varphi_2)$ en formel

2.2. **Exempel.** Några exempel på formler i LP(σ) där $\sigma = \{p, q, r\}$:

- $\bot, r, (p \Rightarrow (\neg r)), (p \vee (\neg p))$

Några som *inte* är formler:

- $(p\wedge, p \vee q \wedge r$ (inga paranteser!)

3. Föreläsning - Naturlig deduktion (syntax)

3.1. Parantesers roll.

Exempel: $p \to q \wedge r$. Hur skall vi placera ut paranteser, och behåller det formelns sanningsvärde? Vad händer om vi skriver $(q \wedge r)$ istället så att $(p \to (q \wedge r))$, eller motsatsen, $(p \to q)$ så att $((p \to q) \wedge r)$.

Det finns en konvention som hjälper oss att hålla koll på var och när och hur många paranteser som behövs.

- Skriv inte ut yttersta paranteser
- $\neg$ binder starkare än $\wedge, \vee, \to, \leftrightarrow$
  - $\neg p \wedge q$ betyder $(\neg p) \wedge q$
- $\wedge, \vee$ binder starkare än pilarna $\to, \leftrightarrow$
  - $p \wedge q \to r \vee s \Leftrightarrow (p \wedge q) \to (r \vee s)$
- $\wedge, \vee$ binder lika hårt.
- $\to, \leftrightarrow$ binder lika hårt.
  - Ex: $p \vee q \wedge r$ ej klart vad som menas, här måste paranteser plaserar ut

Man kan formulera detta genom *parsingträd* där subnoderna *inte* får kommutera, detta kallas för att trädet är *ordnat*.

Ex: $(p \wedge (\neg q))$

Ex: $(p_1 \wedge p_2) \to (\neg p_2 \to (p_1 \to \bot))$. Detta är en *pilformel* och då kallas det för ett *huvudkonnektiv*.

Sats 3.1

Det sist tillagda konnektivet i formeln (motsvarar högsta noden i trädet) kallas för *huvudkonnektiv*.

Varje formel har ett entydigt träd. Men också tvärtom, givet ett träd så kan vi "bygga upp" en entydig formel.

Sats 3.2: Parsingträd

Låt T stå för ett träd. $T : \mathrm{LP}(\sigma) \to \{\text{parsingträd}\}$.

Vi definierar det induktivt, där basen är en p atom: $T(p) = \bullet p$. Sedan påbörjar induktionen:

- Om φ formel med träd $T(\varphi)$, så $T((\neg\varphi)) =$ fig.
- Om φ och ϕ formler med träd $T(\varphi)$ resp $T(\phi)$, så $T((\varphi \square \phi)) =$ fig.

Sats 3.3: Delformel

En *delformel* till en formel φ är en teckensträng från φ som själv är en formel, då är det en delformel. Den triviala delformeln är φ själv.

Exempel: $\varphi = (p_1 \wedge \neg p_2) \to ((p_2 \to p_3) \wedge p_4)$. Då är exempelvis $(p_2 \to p_3)$ en delformel eller $\neg p_2$. Däremot så är $p_3) \wedge p_4$ *inte* en delformel. I parsingträdet så motsvarar varje nod en delformel.

3.2. Syntax.

Hur man kan dra slutsatser på ett syntaktiskt sett:

Om Γ är en mängd av formler och φ är en formel vill vi studera relationen mellan dessa. Följer φ av formlerna i Γ?

Vi kommer studera denna frågan på 2 sätt, syntaktiskt och semantiskt.

- Syntax:
  - Formella bevisregler, tex. $\dfrac{AB}{(a \wedge B)}$

  - $\Gamma \vdash \varphi$, Γ bevisar φ, "det finns ett bevis i naturlig deduktion som har slutsatsen φ vars premisser/antaganden kommer från Γ"
  - Teckensträngsmanipulation
  - Naturlig deduktion
- Semantik:
  - Är φ sann om alla formler i Γ är sanna?
  - $\Gamma \vDash \varphi$

3.3. **Naturlig deduktion.**

För varje konnektiv som vi har kommer vi introducera 2 regler, en för att introducera konnektivet och en för att ta bort.

OBS! Reglerna i naturlig deduktion är *syntax*, det vill säga teckensträngsmanipulation. Alltså det spelar ingen roll vad som representeras, utan vilka regler som man får använda sig på dessa tecken. Eg. $A \wedge B \neq B \wedge A$ eftersom den första har tecknet A på första platsen men det har inte den andra.

- $\wedge$ intro ($\wedge I$). Om vi har teckensträng A och B så kan vi $\dfrac{AB}{(A \wedge B)}$
- $\wedge$-elimination. $\dfrac{(A \wedge B)}{A}$ och $\dfrac{(A \wedge B)}{B}$
- $\rightarrow$-intro. $A \cdots B$, jag börjar med A och jobbar mig mot B S.T $\dfrac{A \therefore B}{(A \rightarrow B)}$. Efter B får man dra vilken slutsats $A \rightarrow B$ där A vilken formel som helst. Om A är en premiss ovanför B så får A strykas.
- $\rightarrow$-elimination. Om A gäller och $A \rightarrow B$ så vet vi att B gäller (Modus pomens). Vi får dra B som slutsats.
- $\vee$-intro. Om jag vet A, då vet jag A eller B: $\dfrac{A}{A \vee B}$
- $\vee$-elimination. Om jag antar A skall jag kunna bevisa exakt teckensträng som om jag antar B.
- $\neg$-intro. Antag A så att man kommer fram till formen $\perp$ (botten kan tolkas som alltid falsk/motsägelse). Då får man dra slutsatsen $\neg A$. Här får man stryka premiss.
- $\neg$-elimination. Om man har visat A och $\neg A$ så får man dra slutsatsen $\neg$ (ingen strykning).
- $\leftrightarrow$-intro. Om $A \rightarrow B$ och $B \rightarrow A$ då kan vi skriva $A \leftrightarrow B$.
- $\leftrightarrow$-elimination. Från $A \leftrightarrow B$ får vi "2 fall", $A \rightarrow B$ och $B \rightarrow A$

Exempel: Vi vill göra ett bevisträd som har följande slutsats, $(A \rightarrow (B \rightarrow A \wedge B))$. I detta träd kommer alla antaganden vara längst upp och slutsatser längst ner. Vi tittar på slutsatsen och märker att huvudkonnektivet är en pil, så vi kommer behöva använda pilintro. Då skall jag antag A och försöka härleda $B \rightarrow (A \wedge B)$, men då måste jag visa att $B \rightarrow (A \wedge B)$ så vi måste anta B för att visa $A \wedge B$ och nu kan jag visa det jag ville visa. Vi använder $\wedge$-intro. Per vårat antagande gäller A, B och därmed $A \wedge B$. Nu kan vi använda pil-elimination för att få bort pilen $B \rightarrow (A \wedge B)$

Sats 3.4: Premiss

En *premiss* är en formel som ej är struken och förekommer högst upp i beviskedet.

Sats 3.5: Slutsats

Formeln som står längst ner i beviskedet.

Sats 4.1: Disjunktion

A eller B. Uttrycker att minst en av A och B är fallet. Betecknas $A \vee B$ där A, B kallas *disjunktionsled* eller *disjunkter*.

Det finns 2 typer av disjunktion, *uteslutande* och *icke-uteslutande*. Uteslutande disjunktion är helt enkelt A eller B (men inte båda) och icke-uteslutande är motsatt.

Disjunktionen är så kallad *inklusiv*, det vill säga det är helt okej att både A och B är sann. Det finns en så kallad *exklusiv* disjunktion vilket kommer lite senare (kanske $A \wedge B$?).

Sats 4.2: Implikation

Om A så B uttrycker att B är fallet givet att A är det. Detta betecknas $A \rightarrow B$. Här är A *antecendenten* eller även *förledet* och B är *konsekventen* eller *efterledet*. En implikation kallas också ivland en *materiell implikation*, *konditionalsats*, *villkorssats*.

Notera här att det kan bli lite klurigt med den näst sista raden i tabellen men! Antag att A är ”jag bajsar” och B är ”jag sitter ner”. Om jag inte bajsar så implicerar det fortfarande att jag *kan* sitta ner, varpå implikationen fortfarande gäller.

Sats 4.3: Ekvivalens

A omm B yttrycker konjunktionen av två implikationer, det vill säga om A så B oh om B så A. Skrivs $A \leftrightarrow B$ och kallas även för *materiella ekvivalenser* eller *bikonditionalsatser*.

Sats 4.4: Molekyler och Atomär

En sats är *molekylär* om den är uppbyggd av en eller två andra satser med hjälp av ett konnektiv. I motsatt fall är satsen *atomär*. En molekylär sats innehåller minst ett konnektiv.

Sats 4.5: n-ställig satsoperator

En *n-ställig satsoperator* är en operator som ”tar in” n variabler. Exempelvis är negationen $\neg$ en 1-ställig operator, medan $\wedge$ är en 2-ställig operator.

Sats 4.6: Huvudoperator

En *huvudoperator* är den operator som har tillämpats sist i uppbyggnaden av satsen. Exempelvis, i $(A \rightarrow (B \vee \neg A))$ är $\rightarrow$ huvudoperatorn. Däremot är $\vee$ huvudoperatorn i delformen $(B \vee \neg A)$.

4.1. **Sanningstabeller.**

För negation ¬:

¬A är sann ⇔ A är falsk ⇔ A inte är sann.

A	¬A
S	F
F	S

För Konjunktion:

A∧B är sann ⇔ A är sann och B är sann.

A	B	A ∧ B
S	S	S
S	F	F
F	S	F
F	F	F

För disjunktion:

A∨B är sann ⇔ A är sann eller B är sann ⇔ minst en av A och B är sann.

A	B	A∨B
S	S	S
S	F	S
F	S	S
F	F	F

För implikation:

A→B är sann ⇔ om A är sann så B är sann ⇔ A är falsk eller B är sann. (Tips, vad är sista sanningsvärdet, dvs sanningsvärdet på B?)

A	B	A→B
S	S	S
S	F	F
F	S	S
F	F	S

För ekvivalens:

A↔B är sann ⇔ om A är sann så är B saan och om B är sann så är A sann. ⇔ A→B är sann och B→A är sann ⇔ A och B har samma sanningsvärde.

A	B	A↔B
S	S	S
S	F	F
F	S	F
F	F	S

Sats 4.7: Satsparametrar

Antalet satsparametrar är antalet ”variabler” i vår utsaga. Exvis, i $A \to (B \wedge C \leftrightarrow \neg A)$ har 3st satsparametrar.

5. Föreläsning - Naturlig deduktion forts.

Exempel: Visa att för alla heltal n gäller att $n^2 + n$ är jämn.

Bevis 5.1: Exempel

Ett heltal är antingen jämn *eller* udda, så vi gör en falluppdelning:

- Fall 1 (jämn):
  - $n = 2k \Rightarrow (2k)^2 + 2k = 4k^2 + 2k = 2(2k^2 + k)$ alltså jämn
- Fall 2 (udda):
  - $n = 2k+1 \Rightarrow (2k+1)^2 + 2k + 1 = 4k^2 + 4k + 1 + 2k + 1 = 2(2k^2 + 3k + 1)$ alltså jämn.

□

Exempel: $A \vee B \vdash B \vee A$ (sanningstabellen för dessa är likadana). Vi skall producera ett beviskträd som har en premiss $A \vee B$ där slutsatsen är $B \vee A$

5.1. Botten ($\perp$).

Reglerna här är inte riktigt "intro/elimination" utan det har med motsägelser osv att göra.

Antag att vi vill visa A, då har vi premissen $\neg A$ så att vi kommer fram till $\perp$. Då kan vi dra slutsatsen A. Detta kallas för RAA = Reductio ad absurdum. Då får vi stryka $\neg A$

Bevis 5.2: Exempel: Det finns oändligt många primtal

Vi antar motsatsen (att det finns ändligt många primtal) och försöker visa motsägelse.

Låt A = det finns oändligt många primtal. Då antar vi $\neg A$ och visar $\perp$, och sen drar slutsatsen A. □

Beviskträd är inte entydiga.

6. Semantik för satslogik

Sats 6.1

En mängd av formler Γ kallas *konsistent* om $\Gamma \nvdash \bot$. Det finns inget bevisträd som visar φ från premisser i Γ. Kallas *inkonsistent* om $\Gamma \vdash \bot$

Exempel: $\{p_0, \neg p_0\}$ är inkonsistent, eftersom vi kan visa botten genom en $\neg$-elimination.

Exempel: $\{A \wedge B, \neg A \vee \neg B\}$ är inkonsistent.

Exempel: $\{A, B, C\}$ är konsistent (kan ej visa botten). Detta visas senare i kursen.

Vi har fortfarande vår signatur σ. Vi vill ge ett värde till varje formel i LP(σ). Detta värde är ett sant eller falskt värde (sanningsvärde). I predikatlogik har värdet "mer värde", det är lite rikare och beskriver mer vad det betyder att något är sant eller falskt gentemot i satslogik där det är binärt.

Sats 6.2

En σ-struktur är en funktion $A : \sigma \to \{0, 1\}$ som tilldelar sanningsvärde till vare satssymbol där 0 betyder falsk och 1 sann.

Exempel: $\sigma = \{p, q\}$. Här finns det 4 olika σ-strukturer eftersom p kan antingen vara falsk eller sann, och samma för q alltså $2 \cdot 2 = 4$:

A_1	1	1
A_2	1	0
A_3	0	1
A_4	0	0

Men vi vill ge sanningsvärden till *varje* formel i LP(σ).

Antag att A är en σ-struktur. Vi definierar en funktion $A^* : \{\text{formler i LP}(\sigma)\} \to \{0, 1\}$. Vi definierade formlerna i LP(σ) genom induktion, vi får göra liknande här:

- Bas:
  - $A^*(\bot) = 0$
  - $A^*(p) = A(p)$ om $p \in \sigma$
- Induktion ($\neg$):
  - $A^*((\neg\varphi)) = 1 - A^*(\varphi)$
- Induktion ($\wedge$):
  - $A^*((\varphi \wedge \psi)) = 1 \Leftrightarrow A^*(\varphi) = A^*(\psi) = 1$
- Induktion ($\vee$):
  - $A^*((\varphi \vee \psi)) = 1 \Leftrightarrow$ minst en av $A^*(\varphi)$ och $A^*(\psi) = 1$
- Induktion ($\to$):
  - $A^*((\varphi \to \psi)) = 0 \Leftrightarrow A^*(\varphi) = 1$ och $A^*(\psi) = 0$
- Induktion ($\leftrightarrow$):
  - $A^*((\varphi \leftrightarrow \psi)) = 1 \Leftrightarrow A^*(\varphi) = A^*(\psi)$

OBS: $A^*(\varphi)$ beroe endast på $A(p)$ för de satssymboler p som ingår i φ.

OBS: Olika värden på A^* fås från sanningsvärdestabeller.

Exempel: $\sigma = \{p, q\}$, σ-strukturerna A_1, A_2, A_3, A_4 från tabell. Vi låter $\varphi = (\neg p) \wedge (p \to q)$. Nu vill vi ta reda på vad φ får för värden på olika A^*:

	p	q	$(\neg p)$	$\wedge$	$(p \to q)$
A_1	1	1	0	0	1
A_2	1	0	0	0	0
A_3	0	1	1	1	1
A_4	0	0	1	1	1

Då blir sanningsvärdestabellen för $A_n^*(\varphi)$ samma som kolonnen under $\wedge$

Sats 6.3: Modell av en formell

σ-strukturen A kallas *modell* för φ om $A^*(\varphi) = 1$, alltså en modell för när φ är sann i A. Notation:

$A \vDash \varphi$

Satisfierbar är en egenskap som en formel kan ha, medan σ-strukturen kan vara en modell om den är sann.

Exempel: Från föregående exempel blev det sant för A_3 och A_4, alltså $A_3 \vDash \varphi$ och $A_4 \vDash \varphi$. Däremot, såg vi att $A_1 \nvDash \varphi$ och $A_2 \nvDash \varphi$ (φ inte sann i A_2)

Sats 6.4: Tautologi

φ kallas för *tautologi* om $A \vDash \varphi$ för varje σ-struktur A. Alltså om sanningsvärdestabellen har värdet 1 i varje rad. Notation:

$\vDash \varphi$

Sats 6.5: Satisfierbar

φ kallas *satisfierbar* om $A \vDash \varphi$ för minst en struktur A. I föregående exempel är φ satisfierbar.

Sats 6.6: Falsifierbar

φ kallas *falsifierbar* om det är möjligt att göra den falsk. $A \nvDash \varphi$ för minst en struktur falsk. I föregående exempel är φ falsifierbar

Sats 6.7: Osatisfierbar

φ kallas *osatisfierbar* om formen är falsk i alla strukturer, det vill säga $A \nvDash \varphi$ i alla strukturer.

7. Logisk ekvivalens och konsekvens

7.1. **Logisk ekvivalens.**

Sats 7.1: Logisk ekvivalens

En relation som 2 formler har till varandra, de kan kallas för *logiskt ekvivalenta.*

Låt σ vara en struktur, φ och ψ i LP(σ). φ och ψ *kallas logiskt ekvivalenta* om $\vDash \varphi \leftrightarrow \psi$

Notation: φ eq ψ, alltså samma sanningsvärde.

7.2. **Exempel.**

$p \to q$ eq $\neg p \vee q$. Här får man rita upp sanningstabellen:

p	q	$p \to q$	$\neg p \vee q$
1	1	1	1
1	0	0	0
0	1	1	1
0	0	1	1

Sats 7.2: Eq

Eq är en ekvivalensrelation

Bevis 7.1: Eq

- Reflexiv (φ eq φ)
- Symetrisk (φ eq $\psi \Leftrightarrow \psi$ eq φ)
- Transitiv (φ eq ψ och ψ eq $\xi \Rightarrow \varphi$ eq ξ)

□

7.3. **Några viktiga ekvivalenser.**

$$\left.\begin{array}{l} p \vee (q \wedge r) \text{ eq } (p \vee q) \wedge (p \vee r) \\ p \wedge (q \vee r) \text{ eq } (p \wedge q) \vee (p \wedge r) \end{array}\right\} = \text{Distributiva lagar}$$

$$\left.\begin{array}{l} p \vee (q \vee r) \text{ eq } (p \vee q) \vee r \\ p \wedge (q \wedge r) \text{ eq } (p \wedge q) \wedge r \end{array}\right\} = \text{Associativa lagar}$$

$$\left.\begin{array}{l} p \vee q \text{ eq } q \vee p \\ p \wedge q \text{ eq } q \wedge p \end{array}\right\} = \text{Kommutativa lagar}$$

$$\left.\begin{array}{l} \neg(p \wedge q) \text{ eq } \neg p \vee \neg q \\ \neg(p \vee q) \text{ eq } \neg(p) \wedge \neg(q) \end{array}\right\} = \text{de Morgans lagar}$$

$$\left.\begin{array}{l} p \vee p \text{ eq } p \\ p \wedge p \text{ eq } p \end{array}\right\} = \text{Idempotenslagar}$$

$$\neg\neg p \text{ eq } p\} = \text{Lagen om dubbel-negation}$$

7.4. **Logisk konsekvens.**

Sats 7.3: Logisk konsekvens

Låt Γ vara en mängd av formler i LP(σ) och låt φ en formel i Γ. φ är en *logisk konsekvens* av Γ, skrivet $\Gamma \vDash \varphi$, om φ är sann i varje modell för Γ.

Dvs, för varje σ-struktur gäller: Om $A \vDash \gamma$ för varje $\gamma \in \Gamma$, så kräver vi att φ (den logiska konsekvensen) också ska vara sann. Här betyder $\vDash$ "när hälst varje struktur i Γ".

7.5. **Exempel.**
Visa att $\{p_1 \to p_2, p_1\} \vDash p_2$ (visa att p_2 är en logisk konsekvens av mängden).

Vi visar detta genom att låta A vara en modell för $p_1 \to p_2$ och p_1, alltså A är en struktur där de två är sanna. Detta betyder att A är en σ-struktur, dvs $A^*(p_1 \to p_2) = 1$ och $A^*(p_1) = 1$. Det vi måste visa är att $A^*(p_2) = 1$. Detta kan vi göra genom att anta att $A^*(p_2) = 0$, vi vill få en motsägelse. Då blir $A^*(p_1 \to p_2) = 0$, men detta mostäger antagandet att $A^*(p_1 \to p_2) = 1$, alltså $A^*(p_2) = 1$

Om inte $\Gamma \vDash \varphi$ gäller, så skriver man $\Gamma \nvDash \varphi$, dvs "φ är inte en logisk konsekvens av Γ" $\Leftrightarrow \neg(\varphi$ sann i varje modell för $\Gamma) \Leftrightarrow$ det finns någon (minst 1) modell för Γ i vilken φ är falsk.

7.6. **Exempel.**
$\{p_1 \to p_2, p_2\} \nvDash p_1$. Vi måste ge en motexempelstruktur A, så att $A^*(p_1 \to p_2) = A^*(p_2) = 1$ och $A^*(p_1) = 0$. Kom ihåg att struktur betyder tilldeling av sanningsvärde till satssymbolerna.

Ta tex följande:

- $A(p_1) = 0$
- $A(p_2) = 1$

Nu har vi hittat en struktur som gör VL sann och HL falsk.

7.7. **Algoritm för att avgöra $\Gamma \vDash \varphi$.**

Använd sanningsvärdestabeller för alla formler i Γ och för φ. För de rader där alla i Γ är sanna, kolla om φ är sann.

7.8. **Exempel.**

Avgör om följande gäller $\neg A \vee \neg B$, $B \vee C \vDash (A \wedge C) \to B$:

A	B	C	$\neg A \vee \neg B$	$B \vee C$	$A \wedge C \to B$
0	0	0	1	0	1
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	1	1

Rad 6 säger ju att VL = 1 men HL = 0, alltså finns modell för Γ så att $(A \wedge C) \to B$ är falsk, alltså $\Gamma \nvDash (A \wedge C) \to B$

7.9. **Exempel.**
Avgör om följande gäller: $\neg(A \to B) \vDash A \wedge \neg B$. OBS, ekvivalent omformulering: $\neg(A \to B)$ eq $\neg(\neg A \vee B)$

Vi kan använda de Morgans lagar och arbeta in icke-tecknet:

$$\neg(\neg A \vee B) \text{ eq } (\neg\neg A \wedge \neg B) \text{ eq } A \wedge \neg B$$

Vi skulle dock visa om den var logiskt konsekvent men vi visade att den var logiskt ekvivalent, då är den alltid sann! Dvs, om den är logiskt ekvivalent så är den automatiskt logiskt konsekvent.

Vi kan resonera på följande sätt: Anta att S är en σ-struktur där $\neg(A \to B)$ är sann.
Dvs, $S^*(\neg(A \to B)) = 1$. Då följer att $S^*(A \to B) = 0$. Stjärna av en pil är bara falsk om förledet är sant men efterledet är falskt, dvs $S^*(A) = 1$ och $S^*(B) = 0$. Nu vet vi vad atomerna måste ha för värde i S.
Alltså $S^*(\neg(B)) = 1$ och sedan $S^*(A \wedge \neg B) = 1$, då har vi visat att $\neg(A \to B) \vDash A \wedge \neg B$

7.10. **Kommentar.**
Det är inte alltid självklart att hela formeln kan substitueras om det är eq.

8. Substitution

Sats 8.1: Substitution

Syftet är att byta ut en delformel i en formel, med en annan formel. Vi börjar med att byta ut satssymboler dock.

Låt φ vara en formel, p en satssymbol, och ψ en formel. Notation: $\varphi\,[\psi/p]$. Utläses "φ med ψ för p", eler alt "i formeln φ, byt ut ψ mot p". Definieras med induktion över hur formeln φ är uppbyggd ty vi vet att φ är konstruerad mha induktion.

För att definiera med induktion:

- Bas:
  - φ atom: $\varphi = q$:
    - $\varphi[\psi/p] = q[\psi/p]$:
      - ψ om $q = p$
      - Annars q
  - φ atom: $\varphi = \bot$:
    - $\varphi[\psi/p] = \bot$ $[\psi/p] = \bot$
- Induktion $\neg$:
  - φ är $(\neg\varphi_1)$:
    - $\varphi[\psi/p] = (\neg\varphi_1)[\psi/p] = \neg(\varphi_1[\psi/p])$
- Induktion $\square$:
  - φ är $(\varphi_1 \square \varphi_2)$, $\varphi[\psi/p] = (\varphi_1 \square \varphi_2)[\psi/p] = (\varphi_1[\psi/p] \square \varphi_2[\psi/p])$

8.1. **Exempel.**

$(p_q \to p_2)[p_7 \wedge p_8/p_2] = (p_1 \to (p_7 \wedge p_8))$

Här behöver vi 2 satser:

Sats 8.2

Anta φ_1 eq φ_2. Då gäller $\varphi_1[\psi/p]$ eq $\varphi_2[\psi/p]$

Exempel på sats 8.2, tag $\varphi_1 = p_1 \to p_2$ och $\varphi_2 = \neg p_1 \vee p_2$. Vi vet att $p_1 \to p_2$ eq $\neg p_1 \vee p_2$ från tidigare. Sats 8.2 ger $(p_1 \to p_2)[p_7 \wedge p_8/p_2]$ eq $(\neg p_1 \vee p_2)[p_7 \wedge p_8/p_2]$, det vill säga
$p_1 \to (p_7 \wedge p_8)$ eq $\neg p_1 \vee (p_7 \wedge p_8)$

Sats 8.3

ntag ψ_1 eq ψ_2 och φ en formel, p en satssymbol. Då gäller: $\varphi[\psi_1/p]$ eller om vi gör $\varphi[\psi_2/p]$ så är de eq. Det vill säga $\varphi[\psi_1/p]$ eq $\varphi[\psi_2/p]$

Exempel på sats 8.3, visa att $\neg(A \to B)$ eq $\neg(\neg A \vee B)$. Låt $\psi_1 = A \to B$ och $\psi_2 = \neg A \vee B$. Här har vi ψ_1 eq ψ_2.
Låt $\varphi = \neg p$, då ger sats 8.3 att $\varphi[\psi_1/p]$ eq $\varphi[\psi_2/p]$, det vill säga $(\neg p)[\psi_1/p]$ eq $(\neg p)[\psi_2/p]$. Nu byter vi ut p mot ψ_1, dvs $\neg\psi_1$ eq $\neg\psi_2$, dvs $\neg(A \to B)$ eq $\neg(\neg A \vee B)$

8.2. **Normalformer.**

Sats 8.4: DNF

En formel är på *disjunktiv normalform (DNF)* om den är på formen:

$$\varphi_1 \vee \varphi_2 \vee \cdots \vee \varphi_n$$

Där varje φ_i är en konjunktion av atomer och/eller negerade atomer.

Exempel, $(p_1 \wedge p_2 \wedge \neg p_3) \vee (p_8 \wedge p_7) \vee (\neg p_8)$ är på DNF. Samma sak med $p_1 \wedge p_2 \wedge \neg p_3$ (här är $n = 1$). Ett exempel på något som inte är på DNF är $\neg(p_1 \wedge p_2)$ eftersom negationen sker inte framför en atom. Däremot är $\neg p_1 \vee \neg p_2$ som är på DNF, och enligt de Morgans lag är de eq.

Exempel, skriv $p \leftrightarrow q$ på DNF. Då vill vi helt enkelt hitta en eq formel som vi kan substituera och som är på DNF.
Vi får $p \leftrightarrow q$ eq $(p \to q) \wedge (q \to p)$ eq $(\neg p \vee q) \wedge (\neg q \vee p)$. Här kan vi använda distributiva lagar så vi får att det är eq $((\neg p \vee q) \wedge \neg q) \vee ((\neg p \vee q) \wedge p)$. Men här har vi tyvärr ”eller”, vi vill ha och! Vi kör distributiva lagar så vi får
eq $((\neg p \wedge \neg q) \vee (q \wedge \neg q)) \vee ((\neg p \wedge p) \vee (q \wedge p))$ eq $(\neg p \wedge \neg q) \vee (q \wedge p)$ är på DNF.

Sats 8.5: Varje formel kan skrivas på DNF

Låt φ vara en formel. Då finns formel ψ på DNF så att φ eq ψ. Detta gäller även för KNF. Generellt, varje formel φ i LP(σ) har formel φ_1 och φ_2 så att:

- φ eq φ_1 eq φ_2
- φ_1 på DNF
- φ_2 på KNF

Detta visas med ett exempel som ger en algoritm.

Bevis 8.1: Varje formel kan skrivas på DNF

$\varphi = ((p \leftrightarrow q) \to r) \to q$. Skriv på DNF. Skriv sanningstabellen:

p	q	r	$(p \leftrightarrow q)$	$\to r$	$\to$	q
1	1	1			1	
1	1	0			1	
1	0	1			0	
1	0	0			0	
0	1	1			1	
0	1	0			1	
0	0	1			0	
0	0	0			1	

Markera raderna där huvudkonnektivet är 1. (Fyll i tabellen)

Vi ser av tabellen att φ eq $(p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (\neg p \wedge q \wedge r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge \neg r)$ där varje parantes motsvarar varje rad där huvudkonnektivet är 1 enligt tabellen.

Notering, $\perp$ är på DNF $\square$

Sats 8.6: KNF

Konjunktiv normalform. Då är det helt enkelt som DNF men med och istället:

$$\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n$$

Där φ_i = disjunktion av atom/$\neg$atom

Lemma 8.1: Följdsats av KNF

Låt φ vara en formel. Då finns ψ på KNF så att φ eq ψ

Kan vi använda en liknande algoritm som i beviset för DNF? Om vi vänder på algoritmen, det vill säga vi markerar raderna med 0 och för varje rad med noll tar vi tvärtom så där det är 1 skriver vi exempelvis $\neg p$. Detta funkar eftersom det vi har gjort är använt de Morgans lagar och vänt på lite saker.

9. Funktionellt kompletta mängder av konnektiv

Vi har observerat att varje formel kan skrivas på DNF. Vi har samtidigt visat att varje formel har en annan formel som endast skrivs med $\wedge, \vee, \neg$. Detta kallas för en *funktionellt komplett mängd av konnektiv.*

Sats 9.1: Funktionellt komplett mängd av konnektiv

En mängd av konnektiver kallas *funktionellt komplett* om varje formel i LP(σ) kan skrivas ekvivalent med en formel som endast innehåller konnektiver från mängden. ”De räcker till för att säga allt som går att säga”.

Exempel:
Mängden $\{\vee, \wedge, \neg\}$ är funktionellt komplett.
Vi påstår även att $\{\wedge, \neg\}$ är också funktionellt komplett. För att visa detta påstående behöver vi att varje formel i LP(σ) är ekvivalent med att bara använda dessa konnektiv.

Bevis 9.1: Föregående mängd är funk. komp

φ i LP(σ). Låt φ_1 eq φ och φ_1 på KNF (eller DNF, spelar ej roll). Men i KNF kanske det finns eller tecken, men vi vill inte ha det i vårat bevis. Vi måste göra något för att få bort eventuella eller tecken. OBS, varje gång vi använder ”eller” så kan vi använda de Morgans lag och skriva om den med ”och” och ”icke”:

$$A \vee B \text{ eq } \neg(\neg A \wedge \neg B)$$

Använder vi andra varianten av de Morgans lag så kan vi göra samma sak med DNF och därmed visa att mängden $\{\vee, \neg\}$ är funktionellt komplett. □

Exempel på en mängd som *inte* är funktionellt komplett:
Mängden $\{\vee, \wedge\}$ är *inte* funktionellt komplett. Det räcker med att visa att en formel inte går att skriva med hjälp av dessa konnektiv. Vi låter mängden A vara mängden av alla formler som går att skriva med konnektiven $\vee, \wedge$. Vi vill definiera den konkret, så vi härmar definitionen av formler:

- Bas
  - $p \in A$ om p satssymbol ($p \in \sigma$)
- Induktion
  - Om φ_1 och φ_2 tillhör A, så $(\varphi_1 \wedge \varphi_2) \in A$ och $(\varphi_1 \vee \varphi_2) \in A$

Betrakta följande σ-strukturen A: a(p) = 1$\forall p \in \sigma$

Om $\varphi \in A$, så gäller A$^*(\varphi) = 1$. (Beviset ges av induktion på A). Men då kan vi betrakta $\psi = \perp \in$ LP(σ). Per definition så är A* $\perp = 0$, men vi har ju definierat σ-strukturen att alltid vara sann, och då kan vi inte skriva alla formler med hjälp av enbart $\wedge, \vee$. Alltså är mängden inte funktionellt komplett.

Exempel: Konnektivet | (sheffers streck) har följande sanningstabell:

p	q	$p\|q$
1	1	0
1	0	1
0	1	1
0	0	1

Vi påstår att mängden $\{|\}$ är funktionellt komplett. Idé, skriv något vi vet är funktionellt komplett med hjälp av strecket och använd *eq*.

Vi vill knyta ihop detta med semantiken och syntax i satslogiken:

Syntax	Semantik
$\Gamma \vdash \varphi$ (det finns ett bevisträd i naturlig deduktion som visar φ med premisser i Γ)	$\Gamma \vDash \varphi$ (φ sann i varje modell för Γ)
$\vdash \varphi$	$\vDash \varphi$ dvs φ tautologi
Γ konsistent	Γ satisfierbar

Sats 9.2: Sundhetssatsen

Om $\Gamma \vdash \varphi$ så $\Gamma \vDash \varphi$.
Specialfall, $\vdash \varphi \Rightarrow \vDash \varphi$

Man kan även vända på pilen, dvs ekvivalent med $\Gamma \nvDash \varphi \Rightarrow \Gamma \nvdash \varphi$

Beviset nämns kort:

Bevis 9.2: Sundhetssatsen

Detta görs egentligen av induktion på alla bevisträd. Det innebär att i basfallet har man korta bevis som bara har en formel. Bevisträdet φ bevisar att $\varphi \vdash \varphi$. Om vi försöker baka ihop till ett induktionsbevis får vi φ som bas och induktionssteg som $\wedge$-intro. Kortfattat, bevisreglerna "bevarar sanning". □

9.1. **Användning av sundhetssatsen.**

När vi pratade om logisk konsekvens så fanns det en formel vi kunde använda för att avgöra om saker var sant eller ej genom att skriva upp sanningsvärdestabeller. En anvädning av denna sats är för att visa att något inte är bevisbart i naturlig deduktion.

Exempel: Visa att $\nvdash \underbrace{(p \to q) \to (p \vee \neg q)}_{\varphi}$. Det hela går ut på att negera sundhetssatsen i någon mening, om vi kan visa att något inte är logiskt konsekvent så är det inte heller bevisbart, dvs om $\Gamma \nvDash \varphi$ så $\Gamma \nvdash \varphi$. Vi visar $\nvDash \varphi$, dvs $\nvDash (p \to q) \to (p \vee \neg q)$, alltså inte en tautologi. Om $p = 0, q = 1$ så är påståendet falskt, dvs vi har hittat en struktur så att den inte är en tautologi, alltså kan vi mha sundhetssatsen dra slutsatsen att $\nvdash \varphi$ eftersom om det inte är en tautologi så är $\vDash \varphi$ men det är den inte.

Påminnelse, en mängd kallas *konsistent* om $\Gamma \nvdash \bot$. Nu har vi en metod för detta!

Exempel: Visa att $\Gamma = \{p_1, p_2, p_3\}$ är konsistent. Vi vill visa att vi inte kan använda dessa premisser för att komma fram till botten. Vi börjar med att visa på semantik sidan att Γ inte är en logisk konsekvens av $\bot$ dvs $\Gamma \nvDash \bot$. Låt A vara en σ-struktur så att $A(p_1) = A(p_2) = A(p_3) = 1$, då bör $A^* \vDash \Gamma$ men $A^* \nvDash \bot$. Sundhetssatsen säger då att $\Gamma \nvdash \bot$.

Sundhetssatsen kan man säga är "pilen från syntax till semantik". Pilen från andra hållet kallas för adekvathetssatsen:

Sats 9.3: Adekvathetssatsen

Om man vet att något är en logisk konsekvens så får man dra slutsatsen i naturlig deduktion, mer formellt, om $\Gamma \vDash \varphi$ så $\Gamma \vdash \varphi$.

Man kan även vända på pilen och få ekvivalent med $\Gamma \nvdash \varphi \Rightarrow \Gamma \nvDash \varphi$

Vi får slutgiltigen att Sundhetssatsen + Adekvathetssatsen = Fullständighetssatsen När vi pratade om logisk konsekvens så fanns det en formel vi kunde använda för att avgöra om saker var sant eller ej genom att skriva upp sanningsvärdestabeller. En anvädning av denna sats är för att visa att något inte är bevisbart i naturlig deduktion.

Fullständighetssatsen = $\underbrace{\Gamma \vdash \varphi}_{\text{Sytax}} \Leftrightarrow \underbrace{\Gamma \vDash \varphi}_{\text{Semantik}}$

Γ konsistent $\Rightarrow$ Γ har modell/Γ är satisfierbar.

Påstående: Varje konsistent mängd av formler i LP(σ) har en modell. Här ger "konsistent" en lettråd om att den befinner sig på semantik-sidan. Om man vill visa adekvathetssatsen så vill vi visa påståendet och att det är ekvivalent med adekvathetssatsen.

Sats 9.4

Γ är konsistent $\Leftrightarrow$ Γ är satisfierbar

Bevis 9.3: Föregående sats

Vi har redan visat $\Rightarrow$. Vi skall visa $\Leftarrow$:
Antag Γ satisfierbar, säg att A är en modell för Γ.
Vi vill visa att Γ är konsistent, dvs $\Gamma \nvdash \bot$
Antag att $\Gamma \vdash \bot$, enligt sundhetssatsen ger det att $\Gamma \vDash \bot$
Eftersom A är en modell för Γ så följer att $\bot$ sann i A, men detta är en motsägelse, ty $\bot$ är falsk i alla strukturer
Från detta följer det att $\Gamma \nvdash \bot$, dvs Γ är konsistent. $\square$

10. Dugga 2021-04-13

I hela duggan har vi $\sigma = \{p, q, r\}$

(1) Gör härledningar i naturlig deduktion som visar att följande sekventer är korrekta där $\varphi, \psi, x \in \text{LP}(\sigma)$

(a) Tänk lite semantiskt först. Antag $\neg\varphi$ och visa x är målet. Vi har en eller formel i VL, så antag $\varphi \wedge \psi$. Då är φ sann och motsäger $\neg\varphi$ och vi får x. I den andra delen av eller tecknet antar vi x och kommit fram till x. Nu kan vi ta $\vee$-elm så vi får x

(b) Här vill vi återigen visa en implikation. vad vi vill göra är:

- Antag φ, försök visa $\neg\psi$. Vi får fram $\neg\psi$ genom att få botten från att ha antagit ψ, så vi antar ψ. Detta är givet från första premissen $(\varphi \wedge \psi)$
- Men första premissen motsäger andra premissen, vi får $\perp$ och vi får därmed $\neg\psi$

(2) Börja med att göra sanningsvärdestabell. Detta kommer även hjälpa oss i fråga 3:

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

För **KNF**, läs från 0-raderna. Vi har bara en 0-rad, vi får $\neg p \vee q \vee \neg r$. Notera att den även är på DNF!

(3) (a) Ja. Det finns minst en 1-rad

(b) Valid betyder tautologi, så nej ty vi har en 0-rad

(c) Frågan är om $\psi \vDash \varphi$ där ψ är $\neg((p \rightarrow q) \vee r)$ och φ är formeln ovan. Men detta betyder att φ är sann i varje struktur av ψ, så vi skriver upp sanningsvärdestabellen för ψ och jämför raderna:

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

Är φ sann då ψ är sann? Svar ja, alltså gäller $\psi \vDash \varphi$

(d) Per definition betyder det att $\varphi \Leftrightarrow \psi$ är en tautologi, dvs $\vDash \varphi \leftrightarrow \psi$. Detta kan vi få genom att jämföra tabellerna och vi ser att de *inte* är samma, alltså inte eq.

(Sekvent = Påstående)

(a) Frågan är om det finns ett bevisträd i naturlig deduktion som har den slutsatsen med de premisserna. Strategin här går ut på att kolla om det snabbt går att göra ett träd i naturligt deduktion. I detta fall är det lite klurigt, så vi kan istället kika på den semantiska sidanoch använda oss av adekvathetssatsen för att "översätta" den tillbaks till den syntaxiska världen. Detta gör man genom att rita upp sanningsvärdestabell för båda sidorna av $\vdash$ tecknet och jämför. Vi vill veta om den är logisk konsekvens och vi ser att det inte är det, vi hittar alltså en motexempelstruktur.

11. Sammanfattning - Kapitel 2-5

Sammanfattning inför Dugga. Vi har hitils gått igenom Semantik och Syntax, vilket motsvarar Kapitel 2 till Kapitel 5 i boken "Grundläggande Logik". Nedan följer sammanfattning och viktiga begrepp från varje kapitel:

11.1. **Kapitel 2.**

Satslogik är en del av *predikatlogik*. I Satslogik behandlas *påståendesatser*, det vill säga satser med ett *sanningsvärde*, exempelvis:

- 7+5=12
- Klocka är tjugo över fem

Istället för att skriva ut hela satser, använder vi istället *satssymboler* såsom exempelvis φ, p

Mängden satssymboler kallas för *satslogisk signatur* och betecknas oftast med σ En byggsten för att koppla samman logik i satser är *konnektiv*, som består av $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \leftarrow$ och kallas även för *logiska symboler*. Dessa konnektiv har alltid samma tolkning, oavsett vilken satslogisk signatur som används.

Konnektiv och satser kan kombineras för att bygga *formler*, detta görs induktivt på följande sätt:

- Basfallet: alla symboler i σ är en formel
- Induktionssteg 1: Om φ är en formel, så är $\neg\varphi$ en formel
- Induktionssteg 2: Låt $\square$ vara ett konnektiv och φ_1, φ_2 formler, då är även $\varphi_1 \square \varphi_2$ en formel.
- Före start av en ny formel bör en startparantes (tillsättas, och i slutet av formeln en slutparantes) tillsättas

Mängden formler givet den satslogiska signaturen σ kallas för *LP(σ)*.
Det är inte alla kombinationer av satser och konnektiv som ger en formel, exempelvis kanske man har fel antal paranteser. Då är det ett *uttryck* och inte en formel. Varje formel är ett uttryck, men inte alla uttryck är formler.

Notera att vi har Induktionssteg 1 och 2, detta eftersom $\neg$ kan ses som enenvariabelfunktion, medan i steg 2 har vi konnektiv som är 2variabelsfunktioner. Mer formellt kallas dessa för *1:ställig resp. 2:ställig konnektiv*.

Det går att visualisera bygget av en formel genom ett så kallat *parsingträd* eller även kallat *konstruktionsträd*. Varje nod i ett sådant träd är en godtycklig formel. Slutnoden/roten kommer från det sista konnektivet som används. Detta konnektiv kallas för *huvudkonnektivet*.

Varje formel som innehåller minst ett konnektiv kallas för *molekylär* formel, men finns denna molekylära formel med som nod i konstruktionsträdet kallas den för *delformel*.

Paranteserna har tagits upp som en pelare i formler, men i vissa fall går det att strunta i dem. Reglerna som bestämmer dessa kallas för *paranteskonventionerna*. Vill man vara på den säkra sidan kan det vara värt att minnas att det är aldrig fel att skriva ut alla paranteser. Reglerna lyder:

- I en formel φ sådant att φ inte står som en delformel i en annan formel och φ har en 2-ställig operator som huvudkonnektiv kan de yttre paranteser slopas
- Led av konjunktion eller led av disjunktion kräver inga paranteser
- Paranteser kring en konjunktion eller en disjunktion som är för eller efterled i en implikation eller ekvivalens kan elimineras

Sammanfattning av sanningstabeller för konjektiven:

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \leftarrow B$	$A \leftrightarrow B$
S	S	F	S	S	S	S
S	F	F	F	S	F	F
F	S	S	F	S	S	F
F	F	S	F	F	S	S

11.2. **Kapitel 3 - Semantik.**

I syntaxen kollar vi på hur vi kan formulera formler genom att koppla samman konnektiv och satser. I semantiken är målet att kunna dra slutsatser från detta, vi vill på något sätt ge mening till att något implicerar något annat.

I semantiken finns det 2 grundprinciper:

- Varje atomär sats har exakt 1 sanningsvärde, sant eller falskt (betecknas S, F eller 1,0)
- De vanliga konnektiven $(\vee, \wedge, \neg, \rightarrow, \leftrightarrow, \leftarrow)$ går att översätta till språk (och, eller, inte) och kan tolkas sanningsfunktionellt

Det sista ordet, sanningsfunktionellt, kan vara lite svårt att förstå, men vad vi menar är att "inte" kan tolkas som en syntaktisk negation och som byter sanningsvärdet på en formel.

Notera ordet "funktion". En *n-ställig sanningsfunktion* är en funktion A som tillordnat ett sanningsvärde till varje satssymbol. Ja, satssymbol, för formler behöver vi lite mer definitioner.

En *σ-strutkur* är en funktion $A : \sigma \rightarrow \{S, F\}$ som tildelar sanningsvärde till varje satssymbol.

För att kunna utvidga denna stuktur så att den täcker mängden av formler ($\mathrm{LP}(\sigma)$) definierar vi funktionen A^* precis på samma sätt som vi utvidgade satser till formler, induktivt. Då utökas strukturen sådant att $A^* : \mathrm{LP}(\sigma) \rightarrow \{S, F\}$:

- Basfallet: $A^*(\bot) = 0$ och $A^*(\varphi) = A(\varphi)$ om $\varphi \in \sigma$
- Induktionssteg $\neg$: $A^*(\neg\varphi) = S \Leftrightarrow A^*(\varphi) = F$
- Induktionssteg $\wedge$: $A^*(\varphi \wedge \psi) = S \Leftrightarrow A^*(\varphi) = A^*(\psi) = S$
- Induktionssteg $\vee$: $A^*(\varphi \vee \psi) = S \Leftrightarrow A^*(\varphi) = S$ eller $A^*(\psi) = S$
- Induktionssteg $\rightarrow$: $A^*(\varphi \rightarrow \psi) = S \Leftrightarrow A^*(\varphi) = F$ eller $A^*(\psi) = S$
- Induktionssteg $\leftrightarrow$: $A^*(\varphi \leftrightarrow \psi) \Leftrightarrow A^*(\varphi) = A^*(\psi)$

12. Predikatlogik - Första ordningens logik

Predikat kan ses som en slags relation, *mindre än relation*, *delbarhet* osv. Ett annat ord för relation är predikat, så predikatlogik är alltså logik som har tillgång till relationer. Syftet med predikatlogik är att skapa bättre språk. Det satslogiska språket är logiskt "fattigt".

Exempel:

- Alla rationella tal är reella ($\forall x(x \in \mathbb{Q} \Rightarrow x \in \mathbb{R})$)
- $2/3$ är ett rationellt tal ($2/3 \in \mathbb{Q}$)
- Alltså följer att $2/3$ är ett reellt tal ($2/3 \in \mathbb{R}$)

Detta går inte att uttrycka i satslogik m.h.a konnektiven, alltså är uttrycket "Alla rationella tal är reella tal" en atomär sats. Men det är även "$2/3$ är ett rationellt tal", då blir det otydligt vilken relation satserna har till varandra.

Vi har 2 sorters tillåtna teckensträngar i språket (vi har inte kommit till språket än):

- *Termer*: Namn på element exvis $2/3$, x, y, $x+3$
- *Formler*: Har sanningsvärde

Bland termerna finns det variabler som kan dyka upp i formler på lite olika sätt. De förekommer ibland *bundet* och ibland *fritt*.
Tag exempelvis $\forall x(x > 3)$, här spelar det ingen roll vad variablen heter. Samma sak i $\int 2xdx$, här är x *bunden*.
Däremot är termen $x+3$ inte bunden, den är *fri* ty den ger en egenskap till just x. Det går att ha både och, exempelvis $\underbrace{\forall x(x > 3)}_{\text{bunden}} \wedge \underbrace{x}_{\text{fri}} = 7$

12.1. **Relationer.**

Det finns n-ställiga relationer (exmepelvis är ">" 2-ställig ty den tar 2 variabler/termer). En relation är alltid på en mängd och relationen ger en delmängd av ordnade par.

En två-ställig relation på mängden A är en delmängd AxA, dvs en mängd av ordnade par. $R \subseteq A$xA

Exempel:
D på $\mathbb{Z}$. $D = \{(a,b) \in \mathbb{Z}x\mathbb{Z} : a|b\}$, här är $(2,6) \in D$ men $(2,7) \notin D$. Det är vanligt att skriva symbolen mellan a och b, så istället för (a,b) skriver vi i detta fall $a|b$.

Ett-ställig relation på A. $R \subseteq A$

Exempel:
$P = \{x \in \mathbb{Z} : x$ är primtal$\}$. Här är $P(3)$ ty 3 primtal. Brukar skrivas $3 \in P$

En tre-ställig relation på $\mathbb{Z}$ är en mängd $R \subseteq \mathbb{Z}$x$\mathbb{Z}$x$\mathbb{Z}$.

Exempel:
$M(a,b,c) \Leftrightarrow b$ är strikt mellan a och c. $(1,2,4) \in M$ men $(1,0,4) \notin M$

12.2. **Funktioner.**

Relationer är altså något som vi vill ha i vårat språk. Ytterliggare något som vi vill ha i språket är *funktioner*.

Låt A vara en mängd. En n-ställig *funktion* (kommer också vara n-ställig) på mängden A är en funktion $F : A\text{x}A\text{x}\cdots\text{x}A \to A$ (n-stycken A).
Observera: F är definierad $\forall$ n-tupler $a_1, a_2, \cdots, a_n$.

Exempel: En 1-ställig funktion från $A \to A$ är $F : A \to A$ är en regel som till varje $a \in A$ tilldelar precis ett element, $F(a)$ i A.

Exempel på en 2-ställig funktion är $+$.

12.3. **Strukturer.**

Vi talar ofta om en struktur, men vad är det? Jo, det är en icke-tom mängd A som är uttrystad med konstanter, relationer, och funktioner:

- $\mathfrak{A} = < A, \{c_i : i \in I\}, F_n, R_n >$
  - där c_i är en konstant, dvs $c_i \in A$
  - där F_i är en funktion $F_i : A\text{x}A\text{x}\cdots\text{x}A \to A$ där ställigheten av funktionen betecknas s_i
  - där R_i är en relation av ställighet r_i. Dvs $R_i \subseteq A\text{x}A\text{x}\cdots\text{x}A$
  - Där A kallas strukturens *universum*.

Man kan ha 0st konstanter, funktioner, eller relationer.

Exempel:
$\mathfrak{N} = < \mathbb{N}, \{0\}, S, <>$
Där $S : \mathbb{N} \to \mathbb{N} \qquad n \mapsto n + 1$ och $<$ är den vanliga mindre-än relationen.
Notera att vi har mängklammrar runt 0. Det behöver vi inte för ändliga mängder som vi kan räkna upp. Se följande exempel.

Exempel:
$\mathfrak{R} = < \mathbb{R}, 0, 1, +, -, \cdot >$

Exempel:
$\mathfrak{Z} = < \mathbb{Z}, 0, + > \leftarrow$ här kommer vi inte kunna *namnge* talen eftersom de alla är 0.

Exempel:
$\mathfrak{Z}_5 =< \mathbb{Z}_5, 0, \oplus >$ (addition modulo 5).

12.4. **Språk för predikatlogik (1:a ordningens logik).**

Bakgrunden är att vi tänker oss att vi vill jobba i strukturen $\mathfrak{N}$. Vi vill hitta på ett språk som passar att resonera i strukturen. Vi vill ha en symbol i alfabetet för konstanter. För varje funktion lägga till en fuktionssymbol, för varje relation lägga till en relationssymbol. Vi börjar på följande sätt:

Låt $\mathfrak{A} = < A, \{c_i : i \in I\}, F_n, R_n >$ vara en struktur.

Vi vill generalisera detta men vi vill inte prata om en strutkur, utan alla strukturer med samma typ (där typ är antalet funktioner, konstanter, relationer samt deras ställighet). Därför önskar vi att införa ett alfabet i språket:
Inför alfabetet $< \{\bar{c_i} : i \in I\}, \{\bar{F_n}\}, \{\bar{R_m}\} >$ där:

- $\bar{c_i}$ är en konstantsymbol
- $\bar{F_n}$ är en funktionssymbol av ställighet s_i
- $\bar{R_m}$ är en relationssymbol med ställighet r_i

Här är ett alfabet med frivilliga symboler (frivilliga ty vi kan välja vilka symboler som helst, till skillnad från konnektiv).

Säg att det är ändlig många konstanter kst. Vi vill konstruera tuppeln σ.
$\sigma =< \bar{c_1}, \cdots, \bar{c_k}, \bar{F_1}, \cdots, \bar{F_n}, \bar{R_1}, \cdots, \bar{R_m} >$ kallas för en *signatur*. För att få generaliteten behöver vi tala om att signaturen är av en viss *typ*.
Typen av σ är $< 0, 0, \cdots, 0; s_1, \cdot, s_n; r_1, \cdots, r_n >$ (dvs räkna upp ställigheterna):

- Konstanterna betraktas som 0-ställiga (varpå nollorna)

Nu kan vi generalisera språket till varje typ σ.

Språket betecknas *LR(σ)* (relationslogik istället för propositionell logik) för signaturen σ av given typ (ovan). Vad behöver vi?:

- Alfabet:
  - Alla symboler i σ och konnektiven och paranteser och kvantorerna $(\exists, \forall)$ samt en radda av uppräknerligt många variabler $x_1, x_2, \cdots$, (skrivs vanligtvis x, y, z), $\doteq$ (2 element är lika) och kommatecken.
  - Alla föregående förutom symbolerna i σ kallas för *logiska* symboler, medan de andra kallas för *icke-logiska*.

Vi skall definiera *termer* och *formler*:

Termer LR(σ) definieras induktivt och ska peka på element:

- Bas:
  - x variabel $\Rightarrow x$ är en term
  - $\bar{c_1}, \cdots, \bar{c_k}$ är termer
- Induktionssteg:
  - Om $\bar{F}$ är en k-ställig funktionssymbol, och $t_1, \cdots, t_k$ är termer, då är $\bar{F}(t_1, \cdots, t_k)$ en term.

Formler i LR(σ) definieras induktivt:

- Bas (atomära formler):
  - $\perp$ är en formel
  - Om t_1 och t_2 är termer, så är $t_1 \doteq t_2$ en formel
  - Om $\bar{R}$ är en k-ställig relationssymbol, och $t_1, \cdots, t_k$ är termer, så är $\bar{R}(t_1, \cdots, t_k)$ en formel
- Induktionssteg (genom konnektiv och kvantorer):
  - Om φ en formel, så är $(\neg\varphi)$ en formel
  - Om φ och ψ är formler, så är $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $(\varphi \rightarrow \psi)$ och $(\varphi \leftrightarrow \psi)$ formler
  - Om φ är en formler och x en variabel, så är $\exists x\varphi$ och $\forall x\varphi$ formler

Vad vi har gjort är inte så konstig,t vi har härmat definitionen av formler i satslogik men vi har ändrat bassteget och kör lite extra induktionssteg. De liknar varandra för att vi vill fortfarande ha tillgång till satslogiken i vår predikatlogik.

Exempel:
Vilka formler i LR(σ) för $\mathfrak{U} = < \bar{0}, \bar{s}, \bar{<} >$?
Några formler i språket är:

- $\perp \qquad \bar{S}(\bar{0}) \doteq x \qquad \bar{S}(\bar{0} \doteq \bar{0})$
- $\big(\bar{<}(\bar{S}(\bar{0}), x) \wedge \bar{0} \doteq \bar{S}(\bar{0})\big)$
- $\forall \bar{S}(\bar{0}) \doteq x \qquad \exists(\bar{S}(\bar{0}) \bar{<} \bar{0} \wedge x \doteq \bar{S}(\bar{S}(\bar{0}))) \qquad$ (notera paranteserna)

Observera: Om $\bar{R}$ är en 2-ställig relationssymbol, och t_1, t_2 är termer, så skriver vi oftast $t_1 \bar{R} t_2$ när vi menar $\bar{R}(t_1, t_2)$

13. Predikatlogik - Forts.

13.1. **Fria och bundna förekomster av variabler.**

Intuitivt skall en bunden variabel inlindas i kvantorer, vi definierar:

Sats 13.1

Alla förekomster av variabler i termer kallas *fria*

En förekomst av variabler x är *fri* (i formeln) om den inte förekommer inom en existens-kvantor med x efter $(\exists x)$ eller $\forall x$.
Annars kallas den för *bunden*

Exempel:
$\forall x \left(\bar{R}(x, \bar{c}) \vee y \doteq \bar{c}\right)$. Här ser vi att det finns 2 variabler som förekommer, x, y. x förekommer vid en all-kvantor och i en relation. x i relationen samt kvantorn är en s.k *bunden förekomst* av x. Vi tittar på y, finns det någon kvantor? Nej, så den är fri. Med bundna variabler skall vi kunna uttala formeln utan att säga variabelns namn, vi kan byta ut den mot "något".

En formel kallas för *sluten* om den saknar fria variabel-förekomster. Exempel i $\mathrm{LR}(\sigma)$ för $\mathbb{N}$:

$\bar{S}(\bar{0}) \doteq \bar{S}(\bar{S}(\bar{0}))$ $\quad$ $\forall x \left(\bar{0} \bar{<} \bar{S}(x)\right)$ (alla variabler förekommer bara i en kvantor) $\quad$ $\exists \left(x \doteq \wedge \bar{S}(x) \bar{<} x\right)$

En sluten formel kallas för *sats*.

Vi vill göra lite som vi gjorde i satslogiken, införa begreppet *semantik* och naturlig dedaktion (syntax).

13.2. **Variabelsubstitution.**

Vi vill definiera detta dels i termer och i formler. Vi börjar med att försöka definiera det intuitivt för termer:

Låt T, t vara termer och x variabler. Vi skriver $T[t/x]$, varje x vi ser ersätts med t i termen T. Definieras vidare med induktion av uppbyggnaden av T

- Bas:
  - Om T är en variabel eller konstantssymbol:
    - $\left.\begin{array}{l} t \text{ om } y = x \\ y \text{ om } y \neq x \end{array}\right\} y[t/x]$
- Induktion:
  - Om $T = \bar{F}(t_1, \cdots, t_k)$, så $T[t/x] = \bar{F}(t_1[t/x], \cdots, t_k[t/x])$

Variabelsubstitution i formler sker på samma sätt men istället för T har vi φ och vi vill fortfarande byta ut en variabel. Betecknas $\varphi[t/x]$ där φ är en formel, t är en term, och x är en variabel. Detta görs även induktivt:

- Bas:
  - $\bot\,[t/x] = \bot$
  - $t_1 = t_2[t/x] = t_1[t/x] \doteq t_2[t/x]$
  - $\bar{R}(t_1, \cdots, t_k)[t/k] = \bar{R}(t_1[t/x], \cdots, t_k[t/x])$
- Induktion:
  - $(\neg\psi)[t/x] = \neg(\psi[t/x])$
  - $(\psi_1 \square \psi_2)[t/x] = \psi_1[t/x] \square \psi_2[t/x]$ där $\square$ är godtyckligt konnektiv
  - $(\forall y \psi)[t/x] = \forall y(\psi[t/]) $ om $y \neq x$, annars $\forall x \psi$ (vi gör ingenting).
  - $(\exists y \psi)[t/x] = \exists y(\psi[t/x])$ om $y \neq x$, annars $\exists x \psi$

Anmärkning: Vi byter alltså **inte** bundna variabler

En variabel substitution kallas för *tillåten* om ingen tidigare fri förekomst av en variabler efter substitutionen blir bunden.

Exempel:
$\exists x(y \bar{<} x)$ är en formel som har en fri variabel förekomst (y). Om vi gör substitutionen $[x/y]$ får vi $\exists x(x \bar{<} x)$. Detta är en "korrekt" substitution, den följer reglerna, men vi vill inte tillåta detta ty om vi "översätter" så står det "det finns någonting som är mindre än sig självt" medan innan vi gjorde substitutionen sa vi "det finns någonting som är större än y". Vi har tappat betydelsen i substitutionen, vilket vi inte kan tillåta.
Alla substitutioner som vi gör framöver skall vara tillåtna.