SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Beskrivningslogiken ALC

av

Arash Valipour

2019 - No K41

Beskrivningslogiken ALC

Arash Valipour

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Erik Palmgren

2019

Sammanfattning

Rapporten börjar med att gå igenom och förklara webben (WWW) och den semantiska webben och anledningen till varför logik bör tillämpas i detta sammanhang. För detta ändamål går vi igenom boolesk algebra, satslogik, predikatlogik och beskrivningslogik (**eng**. Description logic). Slutligen undersöker vi några satser och diskuterar varför beskrivningslogiken är mer anpassad än predikatlogiken för semantiska webben.

Abstract

The report begins by reviewing and explaining the web (WWW) and the semantic web and the reason why logic should be applied in this context. For this purpose we go through boolean algebra, propositional logic, first order logic and description logic. Finally we go through some theorems and discuss why description logic is the preferred choice for semantic web.

Innehåll

1 Inledning **2**
1.1 Syfte och frågeställning 3
1.2 Semantisk webb 3

2 Teoretisk bakgrund **5**
2.1 Boolesk algebra 5
2.2 Satslogik 6
2.2.1 Analytiska tablåer för satslogik 8
2.3 Predikatlogik 13
2.3.1 Analytiska tablåer för predikatlogik 18

3 Beskrivningslogik (en: Description logic) **21**
3.1 Kunskapbas(KB) 27
3.1.1 $\mathcal{ALC}$ TBox 27
3.1.2 $\mathcal{ALC}$ ABox 29
3.2 Tablå-algoritm för $\mathcal{ALC}$ 32

4 Diskussion **41**

5 Slutsats **49**

6 Referenser **52**

A Ordlista **54**

Acknowledgment

Ett stort tack till min handledare Erik Palmgren som har bidragit med råd, engagemang och hans tålamod under arbetet, och ett stort tack också till Reza Chegini för hans stöd under hela utbildningen.
Detta är ett examensarbete som motsvarar 15 högskolepoäng i matematik som är skriven på matematiska institutionen på Stockholms universitet.

1 Inledning

Redan i slutet av 60-talet fanns ARPANET (**eng**. Advanced Research project Agency), som var världens första fungerande paketväxlande nätverk och en föregångare till dagens globala internet. Internet fanns redan innan webbsidor.

Sack [1] beskriver att förr i tiden fanns ingen webb (**eng**. word wide web) och man var tvungen att öppna terminalen för att kunna få tillgång till information.

- Man slog in adressen till den dator som vi ville ha kontakt med.
- Sedan kopplades man till en fjärrdator.
- Data hämtades från fjärrdatorn.
- Filer innehållande data laddades till den lokala datorn.
- Slutligen kunde man läsa den önskade filen på den lokala datorn [1].

Nackdelar med detta system bestod av det faktum att man först var tvungen att kunna kommandon och att man inte hade någon aning om innehållet i det som laddades ner.

Sack [1] säger att webben föddes i Europa först 1990. Detta var betydlig lättare än internet. Man öppnade webbläsaren och kunde direkt ladda ner det önskade dokumentet. De största fördelarna med detta var att:

- Inga professionella kunskaper krävdes.
- Lättåtkomlig information.
- Information kunde inhämtas via flera sökmotorer.

Nackdelarna med denna datasökning var enligt följande:

- Svårigheter att urskilja mellan signifikant kontra icke signifikant data?
- Källans pålitlighet.
- Att sammankoppla spretig data?

Man har nu både kunskap och erfarenhet att kunna lösa dessa problem. Webben som vi använder idag är baserad på språket HTML (**eng**. Hyper Text Markup Language). HTML är ett designspråk som visualiserar data men kan inte återge dess betydelse [1].

1.1 Syfte och frågeställning

Syftet med denna rapport är att närmare undersöka varför man använder logiken i semantiska webben (**eng**. Semantic Web) och varför bland satslogik (**eng**. Proposition logic), predikatlogik (**eng**. Predicate logic) och beskrivningslogik (**eng**. Description logic), beskrivningslogiken är mest anpassad för detta ändamål, och i slutet av rapporten berättas några för- och nackdelar med dem.

1.2 Semantisk webb

Vargie Beal [10] som är redaktör för Webopedia beskriver semantiska webben som en utvidgning (**eng**. extension) av den nuvarande webben som gör det enklare att hitta, dela, återanvända och kombinera data. Den är baserad på maskinläsbara data och byggs på XML-teknik med kapacitet att definiera ett anpassad märkningsschema och RDFs (**eng**. Resource Description Framework Schema) som erbjuder ett flexibelt sätt att presentera data.

Sack [1] beskriver att den semantiska webben definierar förhållandet mellan separata data vilket görs med hjälp av RDF som är baserad på XML (**eng**. Extensible Markup Language):
• XML är syntax som kan definiera enskilda objekt. För att definiera relationen mellan två eller flera objekt krävs RDF.
RDF tillåter oss att göra uttalanden om resurser. Formatet av dessa uttalanden är enkelt. Ett uttalande har alltid följande struktur:

$$< subjekt >< predikat >< objekt >$$

Sack [1] förklarar att ett RDF-uttalande uttrycker en relation mellan två resurser. Subjektet och objektet representerar de två resurserna som är relaterade, predikatet representerar karaktären av deras förhållande. Relationen formuleras i riktningsläge (från subjekt till objekt) och kallas i RDF en egenskap (**eng**. property) och kan åskådliggöras med en graf på följande sätt:

$$\boxed{Anders} \xrightarrow{isBornIn} \boxed{Stockholm}$$

$$\boxed{theMonaLisa} \xrightarrow{wasCreatedBy} \boxed{LeonardodaVinci}$$

Nedan exemplifieras ett enkelt beskrivning av filmen Avatar i RDF med XML-syntax:
• Exempel:
<Description id=”Avatar” ><title>Avatar <title>
<director about="http://www.rdfmovies.com/directors.rdf#James_ Cameron"/ >
<Description/>

Men var går gränsen och hur mycket kan RDF beskriva? För att tydliggöra detta tar rapporten upp ett exempel:

- *Alla män är människor.*
- *Alla kvinnor är människor.*

Sack [1] beskriver att RDF kan inte beskriva att Människor består av män och kvinnor, med andra ord en disjunktion mellan män och kvinna saknas. Ett annat problem med RDF är oförmågan till att hantera kardinalitet (**eng**. cardinality) vilket betyder att inte kunna räkna antalet objekt, därför behövs ett mer avancerad redskap i form av OWL (**eng**. web ontology language).

OWL är en påbyggnad av RDF som ger möjligheten till att beskriva klasser, egenskaper och objekt. OWL innehåller RDF-syntax utan några begränsningar i användning [1].

Sack [1] förklarar att ontologi (**eng**. Ontology) är en filosofisk term som betyder läran om verkligheten. Inom datavetenskap representerar den läran om informationens betydelse som görs tydligt genom formell och standardiserad kunskapsrepresentation. Ontologi måste tillämpas på ett formellt sätt som blir begripligt för datorer. Detta möjliggörs av logiken. Den enklaste typen av logik är satslogik [1].

2 Teoretisk bakgrund

I detta avsnitt förklaras boolesk algebra (**eng**. boolean algebra), satslogik och predikatlogik. Detta i ett led för att slutligen kunna förklara beskrivningslogiken.

2.1 Boolesk algebra

För att kunna kalkylera med logiken introducerade den brittiske matematikern och filosofen George Boole (1815-64) Boolesk algebra. Han noterade att algebra kan användas för att formulera och lösa logiska problem.
Exempel:
l:= lång, k:=kort, s:= svarthår
och man kan säga att lk = 0 vilket betyder att det inte finns någon människa som är kort och lång samtidigt.
Med hjälp av symbolerna skriver vi : "En lång kille som har svart hår, men inte är kort".
vilket blir: l(s-sk) = ls-lsk = ls-(lk)s = ls-0=ls
På detta sätt kan ett komplicerat uttryck förenklas algebraiskt. Boolesk algebra är en gren av algebra där värdet av variabler är sanningsvärdet som betecknas med 1 respektive 0, istället för elementär algebra där värdet av variablerna är tal.

Symboler som används i boolesk algebra är:
$\bigwedge$ vilket betyder och: konjunktion (**eng**. conjunction) som är en tvåställig (**eng**. binary) operation.
$\bigvee$ vilket betyder eller: disjunktion (**eng**. disjunction) som är en tvåställig operation.
$\neg$ vilket betyder komplement (**eng**. complement) och är en enställig (**eng**. unary) operation.
$\rightarrow$ vilket kallas implikation (**eng**. implication) och är en tvåställig operation och (a $\rightarrow$ b) betyder: b är minst lika sann som a.
$\leftrightarrow$ vilket kallas biimplikation (**eng**. equivalence) och (a $\leftrightarrow$ b) betyder: a och b är lika sanna.
Man kan tolka dem som [1]:

$$a \wedge b = a.b$$

[1] Addition **"+"** och multiplikation **"."** är binara operationer svarande $\vee$ och $\wedge$.

$$a \vee b = (a + b) - a.b$$

$$\neg c \equiv 1 - c$$

$$a \rightarrow b \equiv \neg a \vee b$$

$$a \leftrightarrow b \equiv (a \rightarrow b) \wedge (b \rightarrow a).$$

Syntaktiska prioriteringsregler för symboler i boolesk algebra är:
$\wedge$ binder hårdare än $\vee$, $\neg$ binder hårdare än $\wedge$, $\rightarrow$ binder svagare än $\wedge$ och $\vee$, $\leftrightarrow$ som binder svagare än $\rightarrow$ [2, Kapitel 1 & 2].
I boolesk algebra kan alla uttryck skrivas på konjunktiv eller disjunktiv normalform enligt följande exempel [2, Kapitel 1, definition 1.5.5 och 1.5.6]:
Disjunktiv normalform: ($\neg$x$\wedge$y$\wedge$z)$\vee$(y$\wedge$z)$\vee$x.
Konjunktiv normalform:($\neg$x$\vee$y$\vee$z)$\wedge$(x$\vee$z)$\wedge$x.

2.2 Satslogik

Formell logik kallas just det för att de logiska reglerna inte ska påverkas av påståendets betydelse. Endast formen ska vara relevant.
I satslogik kallar vi $P_1, P_2, ..., P_n$ påståendevariabler (**eng**. propositional variables). De kombineras som $P_1 \wedge P_2$, $P_1 \vee P_2$ eller $P_1 \rightarrow P_2$, Betydelsen beror på hur man tolkar P_1 och P_2 [2, Kapitel 4].
Noteringsformeln ges genom följande regler :
• Alla påståendevariabler är en formel.
• Om φ är en formel så är $\neg\varphi$ en formel.
• Om φ_1,φ_2 är formler är $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ och $\varphi_1 \rightarrow \varphi_2$ också formler [3, Kapitel 1].

I den här delen står p,q,r,s för påståendevariabler
A,B,C,X,Y,Z betecknar mängden av påståendevariabler eller formel(formler).
Atomär formel (**eng**. atomic formula): är en formel utan djupare påstående struktur, det vill säga en formel som inte innehåller delformler [3, Kapitel 1].

$\vee$, $\wedge$ och $\rightarrow$ kallas binära konnektiv. Ett sant påstående betecknas $\top$ och ett falsk påstående betecknas $\perp$.
Semantik i satslogik betyder att: påståendevariabler P_1, P_2... kan tolkas på

många olika sätt, till exempel:
Himlen är blå, oavsett falskt eller sant.
En tolkning ger sanningsvärde till symbolerna i ett formellt språk. Ett formel är alltid ett formel men dess tolkning är situationsberoende och betecknas med $\mathcal{A}$. Formler $P_1, P_2, ..., P_n$ tolkas som $P_1^{\mathcal{A}}, P_2^{\mathcal{A}}, ..., P_n^{\mathcal{A}}$. $\wedge$ läses som *och*, $\vee$ som *eller*, $\rightarrow$ som *inducerar*, $\top$ som *sant* och $\bot$ som *falsk*.
Sanningsvärde i boolesk algebra betecknas med 1 och falskvärde med 0.

Nedan beskrivs några viktiga definitioner:
Tolkning (**eng**. interpretation): En sanningsvärde av en formel är ett element i boolesk algebra som bestäms av tolkningen $\mathcal{A}$ genom:

$$[\![P_i]\!] = \begin{cases} 1, & \text{om } P_i^{\mathcal{A}} \text{ är sant} \\ 0, & \text{om } P_i^{\mathcal{A}} \text{ är falsk} \end{cases}$$

$$[\![\top]\!] = 1$$

$$[\![\bot]\!] = 0$$

$$[\![\varphi \wedge \psi]\!] = [\![\varphi]\!] \wedge [\![\psi]\!]$$

$$[\![\varphi \vee \psi]\!] = [\![\varphi]\!] \vee [\![\psi]\!]$$

$$[\![\varphi \rightarrow \psi]\!] = [\![\varphi]\!] \rightarrow [\![\psi]\!]$$

φ och ψ är godtyckliga formler [2, Kapitel 4].

Tautologi: En formel som är sann i alla tolkningar är en tautologi (**eng**. tautology).
Om ψ, φ är två formler så betyder $\psi \equiv \varphi$ alternativ $[\![\psi]\!] = [\![\varphi]\!]$ är sant i alla tolkningar.
• Med $\mathcal{A} \models \varphi$ menas $[\![\varphi]\!]^{\mathcal{A}} = 1$ och i så fall säger man att φ är sann i $\mathcal{A}$.
• Om Γ är en mängd av formler och $\mathcal{A}$ är en model av Γ betyder att $\mathcal{A}$ är en model av alla formler i Γ.
• Med $\varphi_1, \varphi_2, ..., \varphi_n \models \varphi$ menas att φ är sant i alla tolkningar i vilka $\varphi_1, \varphi_2, ..., \varphi_n$ är sanna, man kan säga att φ är en logisk konsekvens av $\varphi_1, \varphi_2, ..., \varphi_n$.
• Det finns 2 olika uttryck i satslogik som ser liknande ut men har olika betydelse:

$$\varphi_1, \varphi_2, ..., \varphi_n \models \varphi$$

$$\varphi_1, \varphi_2, ..., \varphi_n \vdash \varphi$$

Första uttrycket betyder att formeln φ utvärderar till sant om alla element i $\varphi_1, \varphi_2, ..., \varphi_n$ utvärdera till sant, och andra uttrycket betyder att formeln φ kan härledas från antaganden $\varphi_1, \varphi_2, ..., \varphi_n$ med hjälp av vissa regler, med andra ord beskriver de förhållandet mellan uttalanden som är sanna när ett uttalande logiskt följer av en eller flera uttalanden [2, Kapitel 4, 6 & 8].

Nu kommer vi till två av de viktigaste satserna i satslogik: sundhetssatsen (**eng**. soundness theorem) och fullständighetssatsen (**eng**. completeness theorem):
Informellt uttrycker en sundhetssats om ett deduktivt system att alla bevisbara satser är sanna ($\Gamma \vdash \varphi \Rightarrow \Gamma \models \varphi$).
Fullständighet säger att alla sanna satser är bevisliga ($\Gamma \models \varphi \Rightarrow \Gamma \vdash \varphi$).
Ett begrepp som måste förklaras närmare här är:

Deduktion (**eng**. deduction) vilket är identiskt med härledning av slutsatser från givna premisser (antaganden) (**eng**. premises).
Till exempel:
$\frac{\varphi \rightarrow \psi \quad \varphi}{\psi}$: tar två premisser, första är "φ implicerar ψ "och andra är "φ "och returnerar slutsatsen "ψ "[2, Kapitel 4, 6 & 8].

Men hur vet man om formeln X är sann? Hur ska man göra för att bevisa eller motbevisa den? Exempel: $X = [p \vee (q \wedge r)] \rightarrow [(p \vee q) \wedge (p \vee r)]$
Det finns ett antal metoder att resonera kring satslogik som analytiska tablåer (**eng**. analytic tableaux), resolution (**eng**. resolution) och naturlig deduktion (**eng**. natural deduction) men i den här rapporten väljer vi analytiska tablå metoden eftersom enligt Smyllyan [3, Kapitel 2] analytiska tablåer är ett elegant och effektivt bevisförfarande för satslogik som därefter sträcker sig till predikatlogik och beskrivningslogik.

2.2.1 Analytiska tablåer för satslogik

Först bör några begrepp definieras:
•*Boolesk evaluering*: Ett värde v av E (mängd av formler av satslogik) kallas en boolesk evaluering (**eng.** Boolean valuation) om för varje x, y i E följande villkor gäller:
a: Formeln $\neg X$ tilldelas värdet t om X tilldelas värdet f och f om X tilldelas

värdet t.
b: Formeln $X \wedge Y$ tilldelas värdet t om X och Y båda tilldelas värdet t annars $X \wedge Y$ tilldelas värdet f.
c: Formeln $X \vee Y$ tilldelas värdet t om minst en av X och Y tilldelas värdet t annars $X \vee Y$ tilldelas värdet f.
d: Formeln $X \rightarrow Y$ tilldelas värdet f om X,Y tilldelas respektive värden t,f, annars mottar $X \rightarrow Y$ värdet t.
• Med *tolkning* av en formel X menas en tilldelning av sanningsvärden av alla variabler som finns i X.
• Om det finns n variabler i X, då finns det 2^n tolkningar i X.
• En formel X är satisfierad (**eng**. satisfied) omm X är sann minst en boolesk evaluering, en mängd S av formler är satisfierad omm det finns minst en boolesk evaluering där alla element i S är sanna.
• X sanningsfunktionellt (**eng**. : truth-functionally) implicerar Y omm $X \rightarrow Y$ är tautologi och X är sanningsfunktionellt ekvivalent med Y omm formeln $X \leftrightarrow Y$ är tautologi [3, Kapitel 2].
•*Sanningsmängd* : låt v vara en boolesk evaluering, låt S vara en mängd av alla formler som är sanna under v, Det är omedelbart från definitionen av en boolesk evaluering att S uppfyller följande villkor:

1. Bara en formel ur paret $(X, \neg X)$ tillhör S, det vill säga: $\neg X \in S$ omm $X \notin S$.
2. $X \wedge Y$ i S omm $X \in S$ och $Y \in S$.
3. $X \vee Y$ i S omm $X \in S$ eller $Y \in S$.
4. $X \rightarrow Y$ i S omm $X \notin S$ eller $Y \in S$.

En mängd som följer de ovan nämnda villkoren kommer att kallas mättad (**eng.**: saturated) eller sanningsmängd (**eng.**: truth set) [3, Kapitel 2].

Analytiska tablå metoden är en variant av semantiska tablåer. Principen bakom semantiska tablåerna är väldigt enkel: man söker efter en modell (satisfierande tolkning) genom att sönderdela formlerna i mängder av atomer (det vill säga påståendevariabler p, q, r....) och negationer av atomer. En mängd av atomer (och dess negation) är satisfierad omm denna mängd inte innehåller en atom p och dess negation $\neg p$. En formel är satisfierad omm en av dessa mängden är satisfierad [3, Kapitel 2].

Under alla tolkningar ska följande åtta fakta gälla:

1. (a) Om $\neg X$ är sann då X är falsk.
   (b) Om X är sant då $\neg X$ är falsk.
2. (a) Om en konjunktion $X \wedge Y$ är sann, då X,Y är båda sanna.
   (b) Om en konjunktion $X \wedge Y$ är falsk, då antingen X eller Y är falsk.
3. (a) Om disjunktion $X \vee Y$ är sann, då antingen X eller Y är sann.
   (b) Om disjunktion $X \vee Y$ är falsk, då X,Y båda är falska.
4. (a) Om $X \rightarrow Y$ är sann, då X är falsk eller Y är sann.
   (b) Om $X \rightarrow Y$ är falsk, då X är sann och Y är falsk.

I analytiska tablåer används T som sann (**eng**. true) och F som falsk (**eng**. false) och under alla tolkningar en signerad [2] formel TX är sann om X är sann och falsk om X är falsk. FX är sann om X är falsk och falsk om X är sann. Med konjugat av en signerad formel menas T byts mot F och vice versa. Ovannämnda regler kallas dekompositionsregler och symboliseras som [3, Kapitel 2]:

1. $\dfrac{T\neg X}{FX}$, $\dfrac{F\neg X}{TX}$
2. $\dfrac{T(X \wedge Y)}{\begin{array}{c}TX\\TY\end{array}}$, $\dfrac{F(X \wedge Y)}{FX|FY}$ [3]
3. $\dfrac{T(X \vee Y)}{TX|TY}$, $\dfrac{F(X \vee Y)}{\begin{array}{c}FX\\FY\end{array}}$
4. $\dfrac{T(X \rightarrow Y)}{FX|TY}$, $\dfrac{F(X \rightarrow Y)}{\begin{array}{c}TX\\FY\end{array}}$

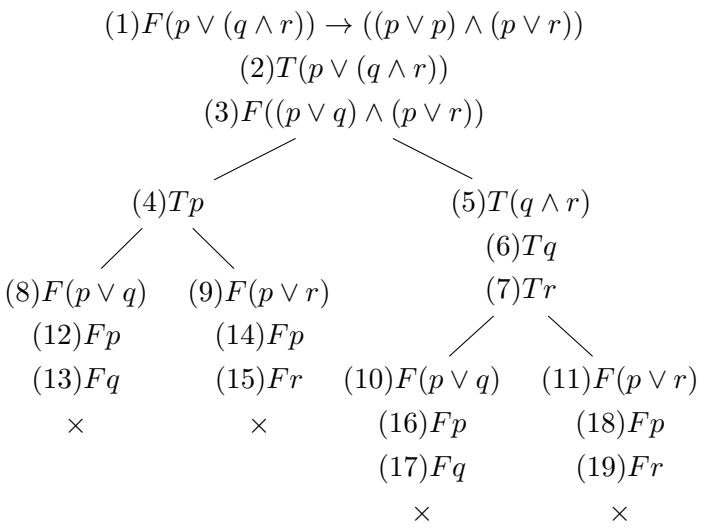
[2] En signerad formel är ett uttryck som ”TX” eller ”FX” där X är osignerad.

[3] T(X $\wedge$ Y) ger direkt både X och Y medan F(X $\vee$ Y) förgrenas in i X och Y, regel 3 och 4 kan förstås analogt.

Signerade formler är av två olika typer:

- De som har direkta konsekvenser: $[F\neg X, T\neg X, T(X\wedge Y), F(X\vee Y), F(X \rightarrow Y)]$
- De som har grenar: $[F(X \wedge Y), T(X \vee Y), T(X \rightarrow Y)]$

Nu kommer vi att undersöka huruvida formeln X är en tautologi eller inte. Låt X = $(p \vee (q \wedge r)) \rightarrow ((p \vee p) \wedge (p \vee r))$

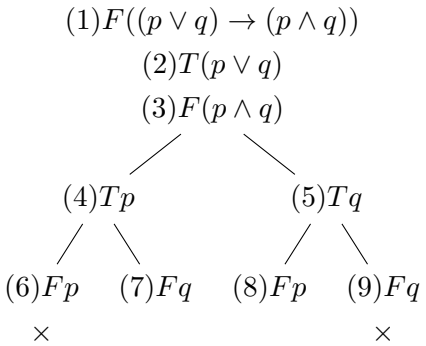


- Som vi ser i exemplet ovan rad 2 och 3 är direkta konsekvenser av rad 1.
- Vi har inga direkta konsekvenser av rad 2, för att antingen Tp eller $T(q \wedge r)$, så 2 grenar in i rad 4 och 5.
- Men rad 5 har direkta konsekvenser vilka är rad 6 och 7.
- Nu kollar vi på rad 3, som vi ser här finns inga direkta konsekvenser, antingen $F(p \vee q)$ eller $F(p \vee r)$.
- Och vi vet att antingen rad 4 eller 5 kommer att hållas, så kommer båda att grena ut i två möjligheter.
- Mer specifikt 4 kommer att grena ut i rad 8 och 9, och rad 5 kommer att grena ut i rad 10 och 11.
- Rad 12 och 13 är direkta konsekvenser av rad 8.

- Rad 14, 15 är direkta konsekvenser av rad 9.
- Rad 16, 17 är direkta konsekvenser av rad 10.
- Rad 18, 19 är direkta konsekvenser av rad 11.
- Rad 12 är direkt motsägelse av rad 4.
- Rad 14 är direkt motsägelse av rad 4.
- Rad 17 är direkt motsägelse av rad 6.
- Rad 19 är direkt motsägelse av rad 7.

Om osignerade formler används i tablåer, X betyder TX och ¬X betyder FX. Och att stänga en gren innebär naturligtvis att grenen avslutas med ett kors så snart två formler uppträder, varav den ena är den negation av den andra. En tablå kallas *sluten* (**eng**. completed) om alla grenar är *stängda* (**eng**. closed). Om vi vill bevisa att en formel X är tautologi, utformar vi inte en tablå för X men för ¬X, som vi gjorde i exemplet ovan med andra ord vi vill se om vi kan härleda en motsägelse från antagandet att formeln X är falsk. [3, Kapitel 2].

Nu kollar vi på exemplet nedan:



Ovanstående tablå finns det två öppna grenar 7 och 8.
Först tas hänsyn till 7, enligt den här grenen Tp och Fq och med Tp och Fq man kan se att (1),(2) och (3) är sanna.
Den andra grenen som är öppen är 8, och enligt den har vi Tq och Fp och vi kan lätt se att (1), (2) och (3) är sanna, detta betyder att $F[(p \vee q) \rightarrow (p \wedge q)]$ är sant och i sin tur bevisar att $[(p \vee q) \rightarrow (p \wedge q)]$ är falsk [3, Kapitel 2].

2.3 Predikatlogik

Hur kan man säga att ” 2 är primtal ” ?
Det bästa sättet man kan göra det på är att använda påståendevariabler som ” 3 är primtal ”, ” 5 är primtal ” och så vidare. Men det borde vara bättre att ha tecken för 2, 3 och 4 som direkt symbolisera predikatet ” är primtal ” [2, Kapitel 9].
När vi talar om predikatlogik använder vi följande symboler:

1. Symboler från satslogik.
2. (a) $\forall$ och läses: för alla.
   (b) $\exists$ och läses: det existerar.
3. (a) En uppräknelig lista med symboler som heter individuella variabler som betecknas med x, y, z, ...
   (b) En uppräknelig lista med symboler som heter parametrar som betecknas med a,b,c, ...
4. För alla positiva heltal n, en uppräkningsbar lista med symboler P,Q,R kallas n-ställiga predikat eller predikat av grad n.

$\forall$ och $\exists$ kallas universella och existentiella kvantifierare (**eng**. universal and existential quantifier) och om man vill beskriva dem med ord:
$\forall$x P(x): innebär att alla x har egenskapen P.
$\exists$x P(x): innebär att något x har egenskapen P [3, Kapitel 4].

För att hänvisa till matematiska objekt använder man termer, vilka är byggda av variabler och funktioner: Alla individuella variabler $(x_1, x_2, ...)$ och parametrar(konstanter)$(a_1, a_2, ...)$ är termer samt om f är en n-ställig funktionssymbol och $t_1, t_2...$ är termer så är det f$(t_1, t_2, ...)$ och när man säger en variabel förekomst (**eng**. occurrence) i en term menas:
x_i förekommer i x_j om $i = j$ och
x_i förekommer i $f_j(t_1, ..., t_n)$ om x_i förekommer i någon argument.
Bra att veta att x_i förekommer aldrig i konstanta symboler [3, Kapitel 9].

En stor skillnad mellan satslogik och predikatlogik är att man kan dra slutsatser som t.ex:
Alex är en matematiker

Anna är en datalog
Alex är kär i Anna
Då drar man slutsatsen att någon matematiker är kär i någon datalog.
Alex är objekt a, Anna är objekt b, matematiker är egenskap P, datalog är egenskap Q och R är förhållandet:
a är Anna
b är Alex
a är R till b
Slutsats: något P är R till något Q.
Vi kan skriva om det hela med hjälp av predikatlogiken:
P(a)
Q(b)
R(a,b)
Slutsats:$\exists$a[P(a)$\wedge\exists$b(Q(b)$\wedge$R(a,b))]
Och tolkas som: det finns ett a som är matematiker och ett b som är datalog och a är kär i b.
Till anslutning till exemplet ovan beskrivs begreppet substitution (**eng**. substitution) : Om A är en formel och x är en variabel och a en parameter då definieras formeln A_a^x med följande induktiva schema [3, Kapitel 4]:

- Om A är en atom då A_a^x är en resultat av substituera a för alla fria förekomster av x i A.
- $[A \wedge B]_a^x = A_a^x \wedge B_a^x$.
- $[A \vee B]_a^x = A_a^x \vee B_a^x$.
- $[A \rightarrow B]_a^x = A_a^x \rightarrow B_a^x$.
- $[(\forall x)A]_a^x = (\forall x)A$.
- $[(\exists x)A]_a^x = (\exists x)A$.

Men om det finns en variabel y skild från x då omvandlade två sista till:

$$[(\forall x)A]_a^y = (\forall x)A_a^y$$

$$[(\exists x)A]_a^y = (\exists x)A_a^y$$

Detta tydliggörs med ett exempel:
Anta formeln A = $(\forall x)Px \vee \neg(\exists y)Q(xy)$ då

$$A_a^x = (\forall x)Px \vee \neg(\exists y)Q(a, y)$$

Detta hänvisas till A_a^x som ett resultat av substituera a för alla fria förekomster av x i A. Med en sluten formel menas en formel A som för varje variabel x och parameter a, $A_a^x = A$ [4, Kapitel 4].

En variabel x är bunden (**eng**. bound) i en formel A som antingen är inom en omfattning (**eng**. scope) av någon förekomst av $(\forall x)$ eller $(\exists x)$ eller det är direkt föregås av $\forall$ och $\exists$. Om x är inte bunden i en formel då är den fri, och x är fri i formel A om en av förekomsterna av x i A är fri.

Exempel:

1. $((\forall x)P(x)) \rightarrow [(\forall x)Q(xy) \vee R(x)]$
2. $(\forall x)[P(x) \rightarrow [(\forall x)Q(xy) \vee R(x)]]$
3. $(\forall x)[P(x) \rightarrow (\forall x)(Q(xy) \wedge R(x))]$

I exemplet ovan kallar vi universella kvantifierare längst till vänster som F1 och det till höger F2.
Omfattning av F1 i (1) är P_x.
Omfattning av F1 i (3) är $P_x \rightarrow (\forall x)Q_{xy} \vee R_x$.
Omfattning av F2 i båda (1) och (2) är Q_{xy}
Omfattning av F2 i (3) är $(Q_{xy} \vee R_x)$.
x har ingen fri förekomst antingen i (2) eller (3) och har bara en fri förekomst i (1) och alla förekomster av y in (1), (2) och (3) är fria.

En **atomär formel** i predikatlogik betyder en n+1 tupel $P(c_1, ..., c_n)$ var P är vilket predikat som helst av grad n och $c_1, c_2, ..., c_n$ kan vara godtyckliga termer.
Grad för en formel A d(A) betyder förekomster av konnektiv i formler och en atomär formel har grad 0 och en formel som har inga parametrar kallas ren(en: pure) formel. En **formel** byggs med hjälp av regler från satslogik som rapporten har förklarat i del 2.2 samt:
Om A är en formel och x är en variabel då båda $(\forall x)$A och $(\exists x)$A är formler

som kallas universell kvantifierare med hänsyn till x respektive existentiell kvantifierare med hänsyn till x [3, Kapitel 4].

Första ordningens evaluering och modeller, innan man beskriver första ordningens evaluering och tolkning, bör man veta att en evaluering i satslogik är en tilldelning av sanningsvärde till en påståendevariabel (**eng.** Propositional variable), med motsvarande tilldelning av sanningsvärdet till alla påstående formler med de variabler, men i predikatlogik (första ordningens logik) och högre ordningslogik, en struktur, (tolkningen) och motsvarande tilldelning av ett sanningsvärde till varje sats i språket för den strukturen [9].

Smullyan [3, Kapitel 4] skriver att: låt U vara en icke-tom mängd som kallas universum av individer eller domän. Vi vill definiera en notation av formler i U med konstanter (U-formler). Med atomär U-formel menar vi $P_{\varsigma_1}, ..., \varsigma_n$ där p är en n-ställig predikat och ς_i är antingen variabel eller element av U(inte parameter), nu kan vi definiera en mängd av alla U-formler med hjälp av regler som beskrevs i början av avsnittet.

En U-formel är som en formel med variabel utan att innehålla någon parameter (ren formel (**eng**. pure formula)). Om F är en U-formel så är det $F_k^x, k \in U$. U-formel definieras precis som F_a^x var a är parameter. Nu låter vi E^U vara en mängd av alla stängda U-formler, med *första ordning evaluering* v av E^U menas en tilldelning av sanningsvärde till alla element av E^U så att för alla A in E^U och alla variabler x följande villkor är uppfyllda:

1. v är en boolesk evaluering av E^U.
2. (a) $\forall xA$ är sant under v omm för varje $k \in U$, A_k^x är sant under v.
   (b) $\exists xA$ är sant under v omm för åtminstone ett element $k \in U$, A_k^x är sant under v.

Med första ordning sanningsmängd (med hänsyn till universum U) menas en delmängd $\mathcal{S}$ av E^U vilken satisfierar en sanningsmängd som är definierad i satslogiken (avsnitt 2.2.1), plus villkoren nedan:
a: $(\forall xA)$ tillhör $\mathcal{S}$ omm för varje $k \in U$, A_k^x tillhör $\mathcal{S}$.
b: $(\exists xA)$ tillhör $\mathcal{S}$ omm för något $k \in U$, A_k^x tillhör $\mathcal{S}$.

Vidare definieras *atomär evaluering* av E^U igenom en tilldelning av sanningsvärden till alla atomära element av E^U, om två första ordning värde gäller för alla atomära element av E^U då måste de gälla för alla element A av E^U [3, Kapitel 4].

Smullyan [3, Kapitel 4] beskriver att en tolkning är en tilldelning av en sats till symbolerna för ett formellt språk. Många formella språk som används i matematik, logik och teoretisk datavetenskap definieras i enbart syntaktiska termer och har sålunda ingen mening förrän de ges någon tolkning. Den allmänna studien av tolkningar av formella språk kallas formell semantik. Termen *Tolkning* användes i satslogik för att tolka en atomär evaluering men i predikatlogik den betyder följande:
Låt E vara en mängd av alla rena (**eng**. pure), slutna (**eng**. closed) formler av kvantifiering sats, med en tolkning I av E i ett universum U menas a funktion som tilldelar till varje n-ställigt predikat P en n-ställig relation P^* av elementen av U.

En atomär U-sats $P\varsigma_1, ..., \varsigma_n$ kallas sann under I om n-tupeln $\sigma_1.....\sigma_n$ står i relation P^*. På detta sätt inducerar tolkningen I en unik atomär evaluering v_0 och man kan säga att ett godtycklig element av E_U, som kan vara icke atomelement, är sant under tolkningen I om den är sant under inducerade atomära evaluering v_0 [4, Kapitel 4].

Det är bra att veta att tolkningen måste vara homomorfism[4] medan evaluering är helt enkelt en funktion.

Låt U vara en domän av naturliga tal och betrakta formeln $(\forall x)(\exists y)P(x, y)$. Det går inte att bedöma om detta är sant eller falsk innan man har gett den en tolkning vilket visar vad P^* är. Om vi tolkar P_{xy} som "$x < y$" då är formeln sann, för att tolkningen blir för alla naturliga tal x, finns ett naturligt tal y som är större än x, vilket stämmer bra då är formeln sann, men om vi tolkar $P(x, y)$ som "$x > y$" då betyder det att för alla naturliga tal x det finns ett naturlig tal y som är mindre, vilket inte stämmer och formeln är falsk [3, Kapitel 4].

[4]I algebra är en homomorfism (**eng**. homomorphism) en strukturbevarande avbildning mellan två algebraiska strukturer av samma typ, t.ex: två grupper, två ringar eller två vektorrum

När man har en mängd S och en tolkning I så att alla element i mängden S under I är sanna då kallas I en *modell* för denna mängd och en ren (**eng**: pure) formel A kallas *giltig* (**eng**. valid) om för varje universum U, A är sann under varje första ordning evaluering, medan A kallas *satisfierad* om för åtminstone ett universum U det finns åtminstone en första ordning evaluering där A är sann [3, Kapitel 4].

En *Boolesk atom* är en sats vilken inte är en boolesk kombination av andra satser(man kan säga att boolesk atom motsvarar påståendevariabel i satslogik), med andra ord en sats är en boolesk atom omm det är antingen en atomär sats $P_{a_1,..,a_n}$ eller är en av formerna ∀xA eller ∃xA.

Betrakta universum V där alla element är parametrar. Nu kan vi prata om både första ordningens evaluering och boolesk evaluering men de två är inte detsamma, alla första ordningens evalueringar är booleska evalueringar av E^v men en boolesk evaluering kanske inte satisfierar kraven för att vara en första ordning evaluering [3, Kapitel 4].

2.3.1 Analytiska tablåer för predikatlogik

Smullyan [4, Kapitel 5] beskriver analytiska tablåer som följande:
Låt X vara en formel, den analytiska tablån för formeln X är ett träd där varje punkt är formler, och resten av tablån är konstruerad med följande 6 regler:

1. $\dfrac{\neg\neg X}{X}$

2. $(a)\dfrac{(X \wedge Y)}{\begin{array}{c}X\\Y\end{array}}$, $(b)\dfrac{\neg(X \wedge Y)}{\neg X | \neg Y}$

3. $(a)\dfrac{(X \vee Y)}{X|Y}, (b)\dfrac{\neg(X \vee Y)}{\begin{array}{c}\neg X \\ \neg Y\end{array}}$

4. (a)$\dfrac{(X \to Y)}{\neg X|Y}, (b)\dfrac{\neg(X \to Y)}{\begin{array}{c}X \\ \neg Y\end{array}}$

5. $(a)\dfrac{(\forall x)Px}{Pa}$(a är en **valfri** parameter), (b)$\dfrac{\neg(\forall x)Px}{\neg Pa}$(a är en **ny** parameter)

6. (a)$\dfrac{(\exists x)Px}{Pa}$(a är en **ny** parameter), (b)$\dfrac{\neg(\exists x)Px}{\neg Pa}$(a är en **valfri** parameter)

Första 4 regler har vi sett redan i avsnitt 2.2.1 men här ser vi dem som osignerade formler, så att vi tog bort T och ersätt ¬ för F.
Men regler 5 och 6 är nya och tillhör predikatlogikdelen.
Vi definierar en tablå som:
• En gren av en tablå är stängd när denna gren innehåller både formeln och dess negation.
• En tablå är sluten om alla dess grenar är stängda.
• En sluten tablå för ¬X utgör ett bevis av X.
Nedan följer ett exempel:

$$(1)\neg((\exists x)(Px \wedge Qx) \to ((\exists x)Px \wedge (\exists x)Qx)))$$
$$(2)((\exists x)(Px \wedge Qx))$$
$$(3)\neg((\exists x)Px \wedge (\exists x)Qx)$$
$$(4)Pa \wedge Qa$$
$$(5)Pa$$
$$(6)Qa$$

$$(7)\neg(\exists x)(Px) \qquad (9)\neg(\exists x)(Qx)$$
$$(8)\neg Pa \qquad\qquad (10)\neg Qa$$
$$\times \qquad\qquad \times$$

rad 2 fick vi från rad 1 med hjälp av regel 4b,
rad 3 fick vi från rad 1 med hjälp av regel 4b,

rad 4 fick vi från rad 2 med hjälp av regel 6a,
rad 5 fick vi från rad 4 med hjälp av regel 2a,
rad 6 fick vi från rad 4 med hjälp av regel 2a,
rad 7 fick vi från rad 3 med hjälp av regel 2b,
rad 8 fick vi från rad 7 med hjälp av regel 6b,
rad 9 fick vi från rad 3 med hjälp av regel 2b,
rad 10 fick vi från rad 9 med hjälp av regel 6b.

Och på slutet ser man att rad 8 och 5, samt 10 och 6 motsäger varann och beviset är klart. Vi kan applicera reglerna i vilken ordning som helst och nå samma slutsats. Den enda skillnaden är att vi kan få ett större träd [3, Kapitel 5].

3 Beskrivningslogik (en: Description logic)

Baader et al. [4, Kapitel 1.1] beskriver beskrivningslogiken (**eng**. Description logic (DL)) som en familj av kunskapsrepresentationsspråk (**eng**. Knowledge representation (KR)) som används för att representera kunskap om en applikationsdomän på ett strukturerat och välförstått sätt. Mest DL är fragment av predikatlogik men är skriven i särskiljande syntax. I DL finns det satser som kommer att vara sanna eller falska som predikatlogik men till skillnad från predikatlogik begreppet behöver inte vara en atom och kan ha struktur (översatt direkt från [4, Kapitel 1.1]).

DL särskiljer vanligen domänkunskap i två komponenter, en terminologisk del kallas TBox (terminologiska rutan (**eng**. terminological box)) och en påstående del som heter ABox (påstående rutan (**eng**. assertional box)) och kombinationen av de två kallas kunskapsbas (**eng.** knowledge base (KB)).

I DL markeras en skillnad mellan den så kallade TBox och ABox. I allmänhet innehåller TBoxen meningar som beskriver begreppshierarkier (dvs relationer mellan begrepp) medan ABox innehåller grundmeningar som anger var i hierarkin individer tillhör (dvs relationer mellan individer och begrepp) [4, Kapitel 1.1].
Till exempel uttalandet:
Alla anställda är en *person*.
Den hör till TBox.
Men
Anna är en anställd.
Hör till ABox.

TBox representerar domänens struktur (som en databasschema) medan ABox representerar kunskap om en konkret situation (som en databas instans). Vi tar upp ett exempel för att tydliggöra detta:
Ett TBox påstående fångar kunskaper om en universitetsdomän och kan innehålla:
En *lärare* är en *person* som undervisar en *kurs*.
En *elev* är en *person* som deltar i en *kurs*.
studenter *undervisar* inte.

Medan Abox påstående från samma domän kan innehålla:
Anna är en *person.*
Matte är en *kurs.*
Anna undervisar *matte1.*

En viktig del av DLs är att uttalanden har en formell logik-baserad semantik och kan se till att ovannämnda satser skrivs i predikatlogik som :

- $\forall x$ (lärare(x) $\leftrightarrow$ person(x) $\wedge$ $\exists y$ (lär(x, y) $\wedge$ kurs(y)))
- $\forall x$ (student(x) $\leftrightarrow$ person(x) $\wedge$ $\exists y$ (deltar(x, y) $\wedge$ kurs(y)))
- $\forall x$ ($\exists y$ (undervisar (x, y))) $\rightarrow$ $\neg$student(x)
- *person*(*anna*)
- *kurs*(*matte1*)
- *lär*(*anna, matte1*)

Och samma satser i DLs skrivs som:

1. *lärare* $\equiv$ *person* $\sqcap$ $\exists$*lär*
2. *student* $\equiv$ *Person* $\sqcap$ $\exists$*deltar.kurs*
3. $\exists$*deltar.*$\top$ $\sqsubseteq$ $\neg$*student*
4. *anna:Person*
5. *matte1:Kurs*
6. *(anna,matte1):lär*

Nummer 1, 2 och 3 tillhör TBox och 4,5 och 6 tillhör ABox. Det som vi märker redan nu är att i DLs används inte variabler (x, y). KB som vi skrev ovan inducerar att *Anna* är en *Lärare* vilket kallas logik-baserad semantik av DL som i sin tur betyder att vi har en väldefinierad gemensam uppfattning när ett påstående framställs av KB [4, Kapitel 1.1].
Attributspråk (**eng**. Attributive language (AL)) är ett grundspråk och familjen av DL som möjliggör en atomär negation, konceptualisering (**eng.** conceptualization), begränsad universell kvantifiering ($\forall R$) och begränsad

existentiell kvantifiering ($\exists R$). En AL-baserad beskrivningslogik är attributet (begrepp) språk med komplement (**eng**. Attributive language with complements (ALC)), där, till skillnad från AL, komplementet till begrepp (**eng**. concept) är tillåtna, inte bara komplementet till atombegrepp.
ALC begreppsuttryck kan innehålla begreppen namn, skärning, förening, komplement, existentiella och universella kvantifierare och enskilda namn [4, Kapitel 2].

Här börjar vi med att gå igenom grundläggande begrepp av **DL** *ALC*, innan vi beskriver ABox och Tbox mer detaljerat. Anta att vi vill beskriva en abstraktion av någon domän av intresse som har många element, här använder vi 3 huvudblock för att förklara de elementen:

- **Begrepp(class):** begrepp (**eng**. concept) representera en mängd av element som kan ses som ett-ställig predikat. De bygger på begreppets namn och roller, en mängd som ett begrepp representerar kallas dess omfång, t.ex: män och kvinna är begrepps-namn och *Sten* är ett element av omfånget av män och *Anna* är ett element av omfånget av kvinna.
- **Roll:** rollnamnet står för binära relationer på element och kan ses som binära predikat, om rollen r hänför sig till ett element med ett annat element så kallas den för en r-utfyllnad av den tidigare, t.ex: om *anna* undervisar matte så kallas matte för en r-utfyllnad av *anna*.
- **Individual(konstant):** individual är namn som t.ex: *Anna*, *Sten*, *Sean*... [4, Kapitel 2.1].

Definition : syntax för ett ALC begrepp definieras på följande sätt:
Låt C vara en mängd av begrepps-namn och R en mängd av rollnamn. Minsta mängden av ALC begreppsuttryck är $\top$ och $\bot$ och varje begrepps-namn a $\in$ C är en ALC begreppsbeskrivning, och om C och D är ALC begreppsbeskrivningar och r $\in$ R, då C $\sqcap$ D (konjunktion), C $\sqcup$ D (disjunktion), $\neg$C (negation), $\forall r$C (universell begränsning (**eng**. value restriction)) och $\exists r$C (existentiell begränsning (**eng**. existential restriction)) är också ALC begreppsbeskrivningar.

$\top$ och $\bot$ kallas *atomär* begrepp och en **sammansättning** begrepp är ett begrepp som är konstruerad genom att använda minst en av tillgängliga

operatörer : $\sqcap$, $\sqcup$, $\forall$, $\exists$ och $\neg$ [4, Kapitel 2].

Vi kan redan nu se likheter mellan ALC, predikatlogik och satslogik, t.ex: $\neg$*Människa* läses , *inte Människa*, och betyder allt som inte är i omfånget (**eng**. extension) av *Människa*. *Människa* $\sqcap$ *Kvinna* läses , *Människa* och *Kvinna*, och betyder allt som är i omfånget av båda *Människa* och *Kvinna*. *Människa* $\sqcup$ *Kvinna* och läses ,*Människa* eller *Kvinna*, och betyder allt som är i omfånget av *Människa* eller *Kvinna* eller båda.

Och två följande exempel är likhet mellan ALC och predikatlogik:
$\forall$lär.Kurs läses alla lär-utfyllnad (**eng**. filler) är kurser, och betyder alla element som endast har element i kursen relaterat till dem genom undervisar.
$\forall$ har nära relation med universell kvantifiering i predikatlogik.
$\exists$lär.Kurs läses det finns en lär-utfyllnad som är en *Kurs*, och betyder alla element som har minst en lär-utfyllnad som är i *Kurs*, och det har nära relation till existentiell kvantifiering i predikatlogik [4, Kapitel 2].

Men en viktig fråga nu är vad betyder begrepp och roller?
ALC tolkningar kan definieras enligt följande:

En terminologisk ALC tolkning $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ över en signatur (C, R, I) består av en icke-tom mängd $\Delta^{\mathcal{I}}$ (domän) och en tolkningsfunktion $\cdot^{\mathcal{I}}$, som avbildar varje individ a till ett element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, varje begrepp till en delmängd av $\Delta^{\mathcal{I}}$ och varje rollnamn till en delmängd av $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, så att för alla ALC-begrepp C och D och alla rollnamn R :

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$,
- $\bot^{\mathcal{I}} = \emptyset$,
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$,
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$,

- $\neg C = \Delta^{\mathcal{I}} \setminus C$,

- $(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} |$ Det finns någon y $\in \Delta^{\mathcal{I}}$ med $< x, y > \in R^{\mathcal{I}}$ och $y \in C^{\mathcal{I}}\}$,

- $(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} |$ För alla $y \in \Delta^{\mathcal{I}}$, om $< x, y > \in R^{\mathcal{I}}$, då $y \in C^{\mathcal{I}}\}$.

Och vi kallar $C^{\mathcal{I}}$ en omfång av C i I och $b \in \Delta^{\mathcal{I}}$ en R-utfyllnad av a in $\mathcal{I}$ if $(a, b) \in R^{\mathcal{I}}$ [4, Kapitel 2].

T.ex: anta vi har en domän $\Delta^{\mathcal{I}}$ och tolkningen $\mathcal{I}$ som följande:

$\Delta^{\mathcal{I}}$	=	{Anna, Matte1, Matte2, Logik1 ,et}
Lärare $^{\mathcal{I}}$	=	{Anna}
Kurs $^{\mathcal{I}}$	=	{Matte1, Matte2, Logik1, et}
Matte $^{\mathcal{I}}$	=	{Matte1, Matte2}
Logik $^{\mathcal{I}}$	=	{Logik1}
Person $^{\mathcal{I}}$	=	{Anna, et}
lär $^{\mathcal{I}}$	=	{(Anna, Matte1), (Anna, Matte2), (Anna, Logik1), (et, et)}

Figur.1: Tolkningen $\mathcal{I}$.

Med hjälp av definitionen kan vi se att alla element är i omfånget av $\top$ och inget element är i omfånget av $\bot$, *Anna* och *et*[5] är i omfånget av *Person* och *et* är en lär-utfyllnad av sig själv.

Om vi utvidgar I till en sammansättning begrepp, vi kan se att Matte1 är en omfång av $Kurs \sqcap \neg Person$, och *Matte1* är en *Kurs* och inte en *Person*, så är även *Matte2* och *Logik1* en omfång av $Kurs \sqcap \neg Person$ och av samma anledning *Anna* är en omfång av $Person \sqcap Lärare$ och i vår tolkning *Anna* är den enda omfång av $Person \sqcap Lärare$ för att *Anna* är det enda element som är en *Person* och en *Lärare*.

[5]et är i detta exempel: Elektronik tekniker certifikat programmet ger flexibel, kompetens-baserad utbildning i grundläggande elektronik. Det har utvecklats för vuxna elever som bedriver grundläggande teknikernivå utbildning genom självständig studie, särskilt studenter som inte kan gå på högskola på heltid på grund av arbete eller familjeåtaganden.

När det gäller existentiell begränsning kan vi se att *Anna* och *et* är i omfånget av $\exists$*lär.Kurs*. Som berättades tidigare om en roll r relaterar ett element med ett annat element då kallar vi den ett r-utfyllnad av den tidigare. $\exists$*lär.Kurs* läses: det finns en r-utfyllnad som är en *Kurs*, vilken innebär alla element som har minst en r-utfyllnad som är i *Kurs*.

I vårt fall undervisar *Anna Matte1*, *Matte2* då *Matte1* och *Matte2* är en lär-utfyllnad av *Anna* och *et* är en lär-utfyllnad av sig själv, samt *Matte1*, *Matte2* och *et* finns i *Kurs*.

För universella begränsningar är det annorlunda, när vi skriver $\forall$*lär.Kurs* då menar vi alla element är i omfånget av den, det är självklarhet för *Anna* och *et*, men även för *Matte1, Matte2 och Fysik1*, för att de har inte någon *lär-utfyllnad* då alla *lär-utfyllnad* som vi förelägger kan satisfiera den. Kortfattat ett element är i omfånget av $\forall$r.C omm den inte har någon r-utfyllnad [4, Kapitel 2].

Nu tittar vi på $\forall$*lär.(Kurs* $\sqcap$ *Matte)*: enligt förklaringen läses, alla *lär-utfyllnad* är *(Kurs* $\sqcap$ *Matte)*, och betyder alla de elementen som har bara element i *(Kurs* $\sqcap$ *Matte)* relaterade med den via *lär*. Med tanke på definitionen kan man säga att *Anna* inte är i omfången av den för att *Anna* har *Logik1* som *lär-utfyllnad* vilken inte är i omfång av *Kurs* $\sqcap$ *Matte*. Men om istället $\forall$ skriver vi $\exists$*lär.(Kurs* $\sqcap$ *Matte)* då är *Anna* i omfånget av den.

Följande sats gör denna observation korrekt.
Sats 3.1: Låt $\mathcal{I}$ vara tolkningen och C, D begrepp och r vara en roll,
då gäller [4, Kapitel 2]:

$$
\begin{aligned}
\top^{\mathcal{I}} &= (C \sqcup \neg C)^{\mathcal{I}}, \\
\bot^{\mathcal{I}} &= (C \sqcap \neg C)^{\mathcal{I}}, \\
(\neg\neg C)^{\mathcal{I}} &= C^{\mathcal{I}}, \\
\neg(C \sqcap D)^{\mathcal{I}} &= (\neg C \sqcup \neg D)^{\mathcal{I}}, \\
\neg(C \sqcup D)^{\mathcal{I}} &= (\neg C \sqcap \neg D)^{\mathcal{I}}, \\
(\neg(\forall r.C))^{\mathcal{I}} &= (\exists r.\neg C)^{\mathcal{I}}, \\
(\neg(\exists r.C))^{\mathcal{I}} &= (\forall r.\neg C)^{\mathcal{I}}.
\end{aligned}
$$

3.1 Kunskapbas(KB)

Vi skrev i början av del 3 att KB är en kombination av en TBox och en ABox och vi kan nu lägga till att när vi jämför KB med DB(databas), en **TBox** är ett **schema** för att det uttrycker generella begränsningar på vad (vår abstraktion av) världen ser ut som och en **ABox** är som **data** eftersom det pratar om ett konkret element och dess egenskap och dess förhållande [4, Kapitel 2].

3.1.1 $\mathcal{ALC}$ TBox

Nu vi börjar med definitionen nedan som förklarar syntax och semantik av TBox:

Vi antar att C och D är två sammansatta begrepp, ett uttryck av formen C$\sqsubseteq$D kallas en ALC generell begreppsinkludering (**eng**. general concept inklusion (GCI)), och C$\equiv$D som kallas en förkortning av C$\sqsubseteq$D och D$\sqsubseteq$C.
En ändlig mängd av GCI kallas en ALC Tbox och tolkningen $\mathcal{I}$ tillfredställer C$\sqsubseteq$D om $C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$, och varje $\mathcal{I}$ som satisfierar GCI i en TBox $\mathcal{T}$ kallas en model av $\mathcal{T}$.

T.ex tolkningen $\mathcal{I}$ som vi såg i figur.1 tillfredställer GCI i:

$\mathcal{T}=$
{lärare $\sqsubseteq$ Person
matte $\sqsubseteq$ $\neg$ Person
lärare $\sqsubseteq$ $\exists$lär.Kurs
$\exists$lär.Kurs $\sqsubseteq$ Person}

Enligt definitionen:
lärare$^{\mathcal{I}}$ = {Anna}⊆{Anna, et} =Person$^{\mathcal{I}}$
Matte$^{\mathcal{I}}$ = {Matte1, Matte2}⊆ {Matte1, Matte2, Logik1}= (¬Person)$^{\mathcal{I}}$
Lärare$^{\mathcal{I}}$ = {Anna} ⊆ {Anna, et} = (∃lär.Kurs)$^{\mathcal{I}}$
(∃lär.Kurs)$^{\mathcal{I}}$ = {Anna, et}⊑{Anna, et} = Person$^{\mathcal{I}}$

I allmänhet tillåter en TBox oss att skilja mellan de tolkningar som är och de som inte är modeller av $\mathcal{T}$, det betyder att vi kan använda TBox för att begränsa vår uppmärksamhet till dem tolkningar som passar våra intuitioner om domänen, t.x: formeln **Kurs** ⊑ ¬**Person** måste vara in vår TBox om vi tycker att en kurs kan inte vara en människa och ∃**lär.kurs** ⊑**lärare** måste vara in vår TBox om vi tycker att bara lärare kan undervisa kurser [4, Kapitel 2].
Så om vi ska visa ett litet exempel av en TBox för undervisning i en högskola kan den se ut ungefär så här:

$\mathcal{T}_1$=	
{ Kurs ⊑ ¬Person	($\mathcal{T}_1$.1)
Matte ⊑ Kurs	($\mathcal{T}_1$.2)
Fysik ⊑ Kurs	($\mathcal{T}_1$.3)
Lärare ≡ Person ⊓ ∃lär.Kurs	($\mathcal{T}_1$.4)
∃lär.⊤⊑ Person	($\mathcal{T}_1$.5)
Student ≡ Person ⊓ ∃delta.Kurs	($\mathcal{T}_1$.6)
∃delta.⊤ ⊑ Person	($\mathcal{T}_1$.7)
}	

Figur.2: Ett exempel av en TBox $\mathcal{T}_1$.

$\mathcal{T}_1$.4 är en likhet och definierar att en *Lärare* är en *person* som undervisar kurser eller tvärtom en *Person* som undervisar en *Kurs* är en *Lärare*.
$\mathcal{T}_1$.5 och $\mathcal{T}_1$.7 säkerställer att bara en person kan undervisa en kurs och kan delta i en kurs.
Slutligen tittar vi på figur 1 och ser att i tolkningen $\mathcal{I}$ et är en *Kurs* och en *Person* och vet att den kan inte vara en modell av $\mathcal{T}_1$ då den bryter mot $\mathcal{T}_1$.1.

3.1.2 $\mathcal{ALC}$ ABox

Vi börjar med en formell definition av ABox:
Låt **I** vara en mängd av individuella namn som är disjunkta från **R** och **C**. For a,b∈**I** individ-namn, **C** ett sammansatt ALC begrepp och r ∈**R** ett roll-namn, en uttryck av formen
•a:C kallas ett ALC begreppspåstående (**eng**. concept assertion), och
•(a,b):r kallas ett ALC rollpåstående (**eng**. role assertion).
En ändlig mängd av ALC begrepps- och rollpåståenden kallas en ALC ABox.

En tolknings-funktion $.^{\mathcal{I}}$ krävs dessutom för att avbilda varje individuellt namn a∈**I** till ett element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.
En tolkning $\mathcal{I}$ satisfierar:
•En begreppspåstående a:C om $a^{\mathcal{I}} \in C^{\mathcal{I}}$ och
•En rollpåstående (a,b):r om $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$.
En tolkning som satisfierar varje begreppspåstående och varje rollpåstående i en ABox $\mathcal{A}$ kallas en model av $\mathcal{A}$.
Nedan i figur 3 visas en ABox $\mathcal{A}_1$ med begrepps- och rollpåstående:

$\mathcal{A}_1$=	
{ Anna: Person	($\mathcal{A}_1$.1)
Logik1: Kurs	($\mathcal{A}_1$.2)
Matte1: Kurs ⊓ Matte	($\mathcal{A}_1$.3)
Betty: Person	($\mathcal{A}_1$.4)
Emma: Person ⊓ Lärare	($\mathcal{A}_1$.5)
(Anna, Matte1): lär	($\mathcal{A}_1$.6)
(Betty, Logik1): lär	($\mathcal{A}_1$.7)
(Emma, Logik1): delta	($\mathcal{A}_1$.8)
(Betty, Matte1): delta	($\mathcal{A}_1$.9)
(Anna, Logik1): delta	($\mathcal{A}_1$.10)
}	

Figur.3: Ett exempel på en ABox $\mathcal{A}_1$.

Som vi skrev i början av del 3 beskriver en ABox hierarkin av individuella och konkreta kunskaper om en korrekt situation. *Anna* är en *Person* och tillhör begreppet *Person*, *Logik1* är en *Kurs* och tillhör begreppet *Kurs* samt berättar ABox för oss att *Anna* undervisar *Matte1* och *Betty* deltar i

Matte1 kursen, *Matte1* är tillhör begreppen *Kurs* och *Matte* (Baader, Horrocks, Lutz & Sattler [4, Kapitel 2]). Följande tolkning $\mathcal{I}$ är en model av $\mathcal{A}_1$:

$\Delta^{\mathcal{I}}$	=	{a,b,Matte1, Logik1}
Anna$^{\mathcal{I}}$	=	{a}
Emma$^{\mathcal{I}}$	=	Betty$^{\mathcal{I}}$ = {b}
Logik1$^{\mathcal{I}}$	=	{l1}
Matte1$^{\mathcal{I}}$	=	{m1}
Person$^{\mathcal{I}}$	=	{a,b,m1,l1}
Lärare$^{\mathcal{I}}$	=	{a, b}
Kurs$^{\mathcal{I}}$	=	{m1,l1}
Matte$^{\mathcal{I}}$	=	{m1}
Logik$^{\mathcal{I}}$	=	{l1}
Student$^{\mathcal{I}}$	=	$\emptyset$
lär$^{\mathcal{I}}$	=	{(a,m1), (b,l1)}
delta$^{\mathcal{I}}$	=	{(b,l1), (a,l1)}

Figur.4: Tolkningen $\mathcal{I}$ som är en modell av $\mathcal{A}_1$.

I tolkningen $\mathcal{I}$, figur 4, vi kan se att både *emma* och *betty* har tolkats som ett element b. Det är tillåtet i vår definition av semantik. Det finns två funktioner i beskrivningslogik som inte delas av de flesta andra databeskrivning formler, ena är att DL inte har antagande om unikt namn (**eng**. unique name assumption (UNA)) vilken betyder att samma namn är tillåtet att hänvisa till olika individer. Det andra är att DL inte har sluten världsantagande (**eng**. close world assumption (CWA)) eller med andra ord DL har öppet världsantagande (**eng**. open world assumption (OWA)) vilken innebär att brist på kunskap om det faktum inte omedelbart innebär kunskap om negation av ett faktum, och vi kan se att i $\mathcal{I}$ omfånget av lärare har mer element än vad $\mathcal{A}_1$ strängt kräver.

Dessutom $\mathcal{I}$ tolkar begreppet logik som är inte nämnt i $\mathcal{A}$. Vi kan se att i figur 4 är $\mathcal{I}$ inte en model av TBox $\mathcal{T}$ i figur 2, t.ex: b $\in$ (Person $\sqcap$ $\exists$delta.Kurs) men h $\notin$ Student$^{\mathcal{I}}$ således satisfierar $\mathcal{I}$ inte axiom $\mathcal{T}_1$.6.

Nu vet vi vad TBox och ABox är och kan vi då definiera KB:
En ALC kunskapsbas $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ består av en $\mathcal{ALC}$ TBox $\mathcal{T}$ och en $\mathcal{ALC}$ ABox $\mathcal{A}$. En tolkning som är en model av både $\mathcal{A}$ och $\mathcal{T}$ kallas en model av $\mathcal{K}$. T.ex: vi bygger en KB $\mathcal{K}_1$ som består av $\mathcal{A}_1$ från figur 2 och $\mathcal{A}_1$ från figur 3, och resultatet blir $\mathcal{K}_1$=($\mathcal{T}_1$,$\mathcal{A}_1$).

Som vi nämnde tidigare a och b deltar i kursen *Logik1* men är inte i omfånget av *student* i $\mathcal{I}$ och bryter axiom $\mathcal{T}_1$.6 och m1 och l1 är i omfång av *Person* och *Kurs* därmed bryter axiom $\mathcal{A}$ [4, Kapitel 2].
En tolkning $\mathcal{I}^*$ av $\mathcal{K}_1$ är:

$\Delta^{\mathcal{I}*}$={a,b,e,Matte1,Matte2,Logik1}
Anna$^{\mathcal{I}*}$= {a}
Betty$^{\mathcal{I}}$ = {b}
Emma$^{\mathcal{I}*}$= {e}
Logik1$^{\mathcal{I}*}$= {l1}
Matte1$^{\mathcal{I}*}$={m1}
Matte2$^{\mathcal{I}*}$={m2}
Person$^{\mathcal{I}*}$={a,b,e}
Lärare$^{\mathcal{I}*}$={a,b,e}
Kurs$^{\mathcal{I}*}$={m1,m2,l1}
Matte$^{\mathcal{I}*}$={m1,m2}
Logik$^{\mathcal{I}*}$={l1}
Student$^{\mathcal{I}*}$={a,b,e}
lär$^{\mathcal{I}*}$={(a,m1), (b,l1), (e,m2)}
delta$^{\mathcal{I}*}$={(b,l1), (a,l1), (b,m2)}

Figur.5: Tolkningen $\mathcal{I}^*$ som är en modell av $\mathcal{K}_1$.

$\mathcal{A}_1$ visar att *Emma* är en *Lärare* och vi vet från $\mathcal{T}_1$.1 att *Emma* måste åtminstone undervisa en *Kurs* men vi vet inte vilken, en sån ofullständig information är inget problem i DL. Vi vet att *Emma* står för ett element som är *lär-relaterad* men vi vet inte vilket element, d.v.s $\mathcal{K}_1$ har andra modeller i vilka *Emma* undervisar olika kurser [4, Kapitel 2].

3.2 Tablå-algoritm för $\mathcal{ALC}$

ALC består av logiska och icke-logiska symboler.

Logiska symboler:

- Skiljetecken: "(", ")", ".".
- Operatorer: $\neg, \exists, \forall, \sqcap, \sqcup$.
- Konnektiv: $\sqsubseteq, \equiv$.

Icke-logiska symboler:

- Begrepp: representerade av C, D, A.
- Roller: representerade av r, s.
- Individer: representerade av a, b.

Syntax är en kombination av symboler och vi avbildar syntax till verkliga världens objekt genom tolkning och hur syntaxen tolkas utgör semantik. Syntax beskriver hur du säger något och semantiskt beskriver innebörden bakom det som sades.

Begrepp	**Syntax**	**Semantik**	**Exempel**
Universellt begrepp	$\top$	$\Delta^{\mathcal{I}}$	Universum
Botten begrepp	$\bot$	$\emptyset$	Tom mängd
Negation	$\neg$C	$\Delta^{\mathcal{I}} \setminus \mathrm{C}^{\mathcal{I}}$	$\neg$ Människa
Universell begränsning	$\forall$R.C	$\{a \in \Delta^{\mathcal{I}} \mid \forall \mathrm{b.(a,b)} \in \mathrm{R}^{\mathcal{I}} \rightarrow \mathrm{b} \in \mathrm{C}^{\mathcal{I}} \}$	$\forall$harBarn.Kvinna
Existentiell begränsning	$\exists$R.C	$\{a \in \Delta^{\mathcal{I}} \mid \exists \mathrm{b.(a,b)} \in \mathrm{R}^{\mathcal{I}} \wedge \mathrm{b} \in \mathrm{C}^{\mathcal{I}} \}$	$\exists$harBarn.Kvinna
Konjunktion	C $\sqcap$ D	$\mathrm{C}^{\mathcal{I}} \cap \mathrm{D}^{\mathcal{I}}$	Människa $\sqcap$ Män
Disjunktion	C $\sqcup$ D	$\mathrm{C}^{\mathcal{I}} \cup \mathrm{D}^{\mathcal{I}}$	Män $\sqcup$ Kvinna

Figur.6: Relation mellan begrepp, syntax och semantik.

Nu rekapitulerar vi tablå-algoritmen kort:
Tablå-algoritmen är en bevisalgoritm för att kontrollera konsistensen i en logisk formel genom att dra slutsatsen att dess negation ger en motsägelse.

För detta ändamål konstruerar vi ett träd där varje nod är markerad med en logisk formel. En väg från roten till ett blad är en konjunktion av

alla formler som representeras i vägens knutpunkter. En förgrening av vägen representerar en disjunktion. Trädet är konstruerat genom successiv applikation av expansionsregler (figur.7) [4] [2].

$\sqcap$-regel: om 1. a:C$\sqcap$D $\in \mathcal{A}$ och
2. {a:C,a:D} $\not\subseteq \mathcal{A}$
då $\mathcal{A} \rightarrow \mathcal{A} \cup$ {a:C,a:D }
$\sqcup$-regel: om 1. a:C$\sqcup$D $\in \mathcal{A}$ och
2. {a:C,a:D} $\cap \mathcal{A} = \emptyset$
då $\mathcal{A} \rightarrow \mathcal{A} \cup$ {a:X } för något X $\in$ {C,D}
$\exists$-regel: om 1. a:$\exists$r.C $\in \mathcal{A}$ och
2. det finns inte b så att {(a,b):r,b:C } $\subseteq \mathcal{A}$, och
3. a är inte blockerad.
då $\mathcal{A} \rightarrow \mathcal{A} \cup$ {(a,d):r,d:C} där d är ny i $\mathcal{A}$
$\forall$-regel: om 1. {a:$\forall$r.C,(a,b):r } $\subseteq \mathcal{A}$ och
2. b:C $\notin \mathcal{A}$,
då $\mathcal{A} \rightarrow \mathcal{A} \cup$ {(b:C }
$\sqsubseteq$-regel: om 1. a:C $\in \mathcal{A}$, $\top \sqsubseteq$D$\in \mathcal{T}$ och
2. a:D$\notin \mathcal{A}$
då $\mathcal{A} \rightarrow \mathcal{A} \cup$ {(a:D }

Figur.7: Syntaktiska expansionsregler för $\mathcal{ALC}$ KB konsistens [4, Kapitel 4.2.3].

I en sluten ABox, konsistens kan kontrolleras genom att hitta en motsägelse vilken kallas krock (**eng**. clash).

Definition 3.2.4: (komplett och krock-fri ABox)
En ABox $\mathcal{A}$ som har ett individuellt namn och ett begrepp C så att {a:C, a:$\neg$C} $\subseteq \mathcal{A}$, innehåller en krock. En ABox $\mathcal{A}$ som inte innehåller en krock kallas krock-fri. $\mathcal{A}$ kallas *sluten* om det har en krock eller inga av expansionsreglerna är tillämpliga [4, Kapitel 4.2.1].

En *algoritm* för $\mathcal{ALC}$ som avgör konsistensen av en $\mathcal{KB}$, tar som input en $\mathcal{KB}$ och tar reda på om den är satisfierbar eller inte, vilken görs på följande sätt:

1. Ersätter axiom:
   - $C \equiv D$ med $C \sqsubseteq D$ och $D \sqsubseteq C$.
   - $C \sqsubseteq D$ med $\neg C \sqcup D$.
2. Transformerar en KB till NNF (**eng.** Negation Normal Form) vilket betyder att negation placeras direkt framför atombegrepp (Figur.8).
3. Använder påstående på individer för att skapa en initial-graf.
4. Använder expansionsregler för att lägga till begrepp till noder tills antingen:
   (a) Motsägelse, t.ex: ¬C och C är in i samma graf, då är KB inte satisfierbar.
   (b) Inga fler begrepp kan läggas till, då KB är satisfierbar.

Nedan ser vi regler som omvandlar en KB till NNF:

Begrepp	**NNF**
NNF(C)	C, om C är atomär
NNF(¬ C)	¬ C, om C är atomär
NNF(¬(¬ C))	NNF(C)
NNF(¬⊤)	⊥
NNF(¬⊥)	⊤
NNF(C ⊓ D)	NNF(C) ⊓ NNF(D)
NNF(C ⊔ D)	NNF(C) ⊔ NNF(D)
NNF(¬(C ⊓ D))	NNF(¬C) ⊔ NNF(¬D)
NNF(¬(C ⊔ D))	NNF(¬C) ⊓ NNF(¬D)
NNF(∀R.C)	∀R.NNF(C)
NNF(¬∀R.C)	∃R.NNF(¬C)
NNF(∃R.C)	∃R.NNF(C)
NNF(¬∃R.C)	∀R.NNF(¬C)

Figur.8: Regler för att omvandla en KB till NNF.

Nedan definieras algoritmen formellt:

Algorithm consistent()
Input: en normaliserad $\mathcal{ALC}$ ABox $\mathcal{A}$
if expand($\mathcal{A}$) $\neq \emptyset$ then
    return "consistent"
else
    return "inconsistent"

Algorithm expand()
Input: en normaliserad $\mathcal{ALC}$ ABox $\mathcal{A}$
if$\mathcal{A}$ är inte komplett **then**
    väljs en R regel som gäller för $\mathcal{A}$ och en påstående eller ett par påstående α i $\mathcal{A}$ till vilken R är tillämpliga
    if det finns $\mathcal{A}'$ $\in$ exp($\mathcal{A}$,R,α) med expand($\mathcal{A}'$) $\neq \emptyset$ **then**
        return expand($\mathcal{A}'$)
    else
        return $\emptyset$
else
    if $\mathcal{A}$ innehåller en krock **then**
        return $\emptyset$
    else
        return $\mathcal{A}$

Figur.9: Tablå-algoritmen konsistent för $\mathcal{ALC}$ KB och KB expansion algoritmen expand [5, Kapitel 4.2.3].

Definition 3.2.7: (Algoritm för $\mathcal{ALC}$ KB konsistens) Algoritmen som är konsistent för $\mathcal{ALC}$ KB konsistens tar som indata en normaliserad $\mathcal{ALC}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ och använder algoritmen "*expand* " för att tillämpa regler från figur 8 till $\mathcal{A}$ med avseende på axiom i TBox $\mathcal{T}$ [5, Kapitel 4.2.3].

Den här delen av rapporten avslutas med några exempel. Algoritmen som vi använder för detta ändamål är nästan samma som vi förklarade ovan.

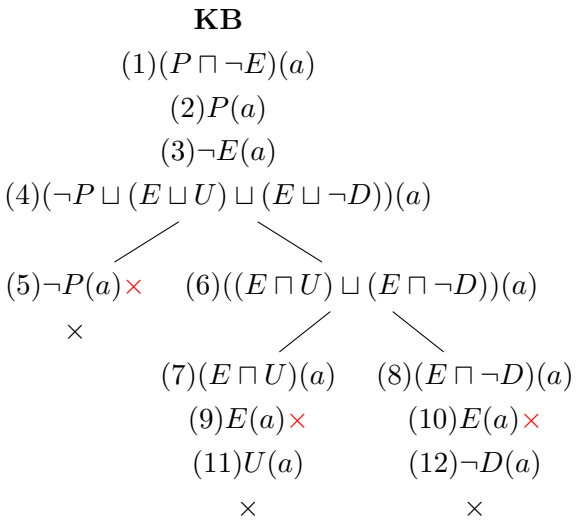
Jag tillämpar första algoritmen (konsistens av $\mathcal{KB}$) för att avgöra bevisbarhet av formel $\mathcal{S}$ ($\mathcal{KB} \vDash \mathcal{S}$) genom att lägga till $\neg\mathcal{S}$ till $\mathcal{KB}$ och testa konsistens av utvidgad $\mathcal{KB}$, vilken görs genom följande steg:

1. Vi lägger till negationen av $\mathcal{S}$ till $\mathcal{K}$.
2. Vi konverterar:
   - A $\sqsubseteq$ B till $\neg$A $\sqcup$B.
   - A$\equiv$B till A$\sqsubseteq$B och B$\sqsubseteq$A.
3. Vi konverterar $\mathcal{K}$ till NNF.
4. Använder påståenden på individer för att skapa en initial-graf.
5. Vi använder expansionsregler för att lägga till begrepp till noder tills antingen:
   (a) Motsägelse förekommer i alla val, då $\mathcal{K} \vDash \mathcal{S}$.
   (b) Inga fler begrepp kan läggas till, då $\mathcal{K} \nvDash \mathcal{S}$.

Första exemplet är:
P.....Professor.
E.....Person.
U.....Universitets-anställd.
D.....Student.
betraktar följande KB:
P$\sqsubseteq$(E$\sqcap$U)$\sqcup$(E$\sqcup\neg$D)

Är P$\sqsubseteq$E en logisk konsekvens av vår KB? Vi ska bevisa att en professor är en person, det är självklart men vi ska bevisa det på ett formellt sätt. Först skriver vi vår KB med negationsfrågan i NNF:

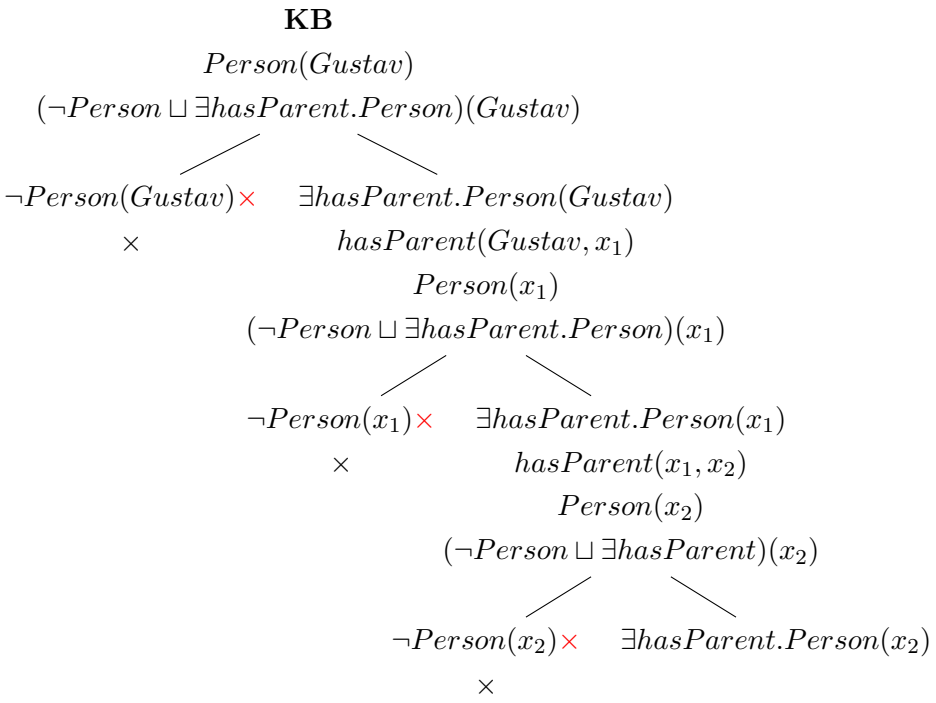
KB = {$\neg$P$\sqcup$(E$\sqcap$U)$\sqcup$(E$\sqcap\neg$D), (P$\sqcap\neg$E)(a)} .



Motsägelsen bevisar att en *Professor* är en *Person* [1].

Nästa exempel som Sack [1] tar upp är en KB som innehåller kvantifierare.
KB: ¬Person ⊔ ∃harFöräldrar.Person
Vi ska dra slutsatsen att Gustav inte är en person: ¬Person(Gustav) ??
Andra delen initieras med individ ”Gustav”.

KB = {¬Person ⊔ ∃hasParent.Person, Person(Gustav)}.



Vi kan dra slutsatsen att det måste vara en annan person som är förälder till x_1, och det måste vara en person. Vi inducerar existensen av individ x_2 och här har vi rollen ” hasParent ” som ansluter x_1 och x_2, och x_2 är en person. Vi expanderar KB med ny individ och vi har disjunktion igen med en ny individ x_2. Sedan grenas ut disjunktion igen på samma sätt. Vänstra delen är redan stängd (**eng.** closed) och vi har existentiell begränsning och kan börja igen men detta kommer aldrig att sluta.

Detta är ett problem eftersom en algoritm bör sluta och vara avgörbar. Då detta aldrig kommer att sluta av sig själv, behövs en blockerande mekanism. Såna problem händer ibland med $\exists$ och ingen avslutning är möjlig. Istället att gå igen och igen utan att producera ny kunskap och data används blockeringsmetoden (Definition 3.2.6).

Vi har två möjligheter när algoritmen är avslutad (**eng. Terminated**):

1. vår tablå-algoritm är stängd vilken betyder att KB är inte satisfierbar.
2. Alla inte stängda delar från tablå-algoritm leder inte till en ny förlängning vilken betyder KB är satisfierbar.

I vårt fall visar tablå-algoritmen att Gustav är en person [1].

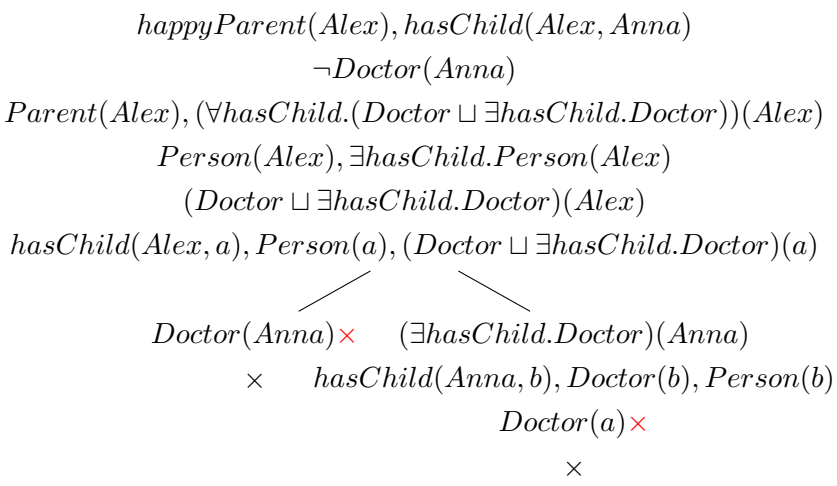
Definition 3.2.6 ($\mathcal{ALC}$ blockering):
Ett individnamn b i en $\mathcal{ALC}$ ABox $\mathcal{A}$ är blockerad med ett individnamn a om
a är en förfader av b, och
$con_{\mathcal{A}}(a) \supseteq con_{\mathcal{A}}(b)$ [6] [4, Kapitel 4.2.3].

När ett individnamn b är blockerade i $\mathcal{A}$ om det är blockerad med något individnamn a, eller om en eller fler av dess förfäder är blockerad i $\mathcal{A}$.
Två följder av den senaste definitionen är: (1) när ett individnamn är blockerat, då är alla dess ättlingar (**eng**. descendants) blockerade, (2) eftersom en individ i roten inte har någon förfader (**eng**. ancestors) kan den aldrig bli blockerad [4, Kapitel 4.2.3].

Betrakta följande KB:
$\mathcal{T}$={Läkare $\sqsubseteq$ Person, Förälder $\equiv$ Person $\sqcap$ $\exists$harBarn.Person,
GladFörälder $\equiv$ Föräldrar $\sqcap$ $\forall$harBarn.(Läkare $\sqcup$ $\exists$harBarn.Läkare)}
$\mathcal{A}$={Alex:GladFörälder, Alex harBarn Anna}
$\mathcal{KB}$ $\models$ Anna:Läkare ?
•Först omvandlar man logisk konsekvens till KB (inte) satisfierad
- $\mathcal{KB}$ $\models$a:C omm $\mathcal{KB}\cup$ {a:($\neg$C)} är inte satisfierad
-$\mathcal{KB}$ C $\sqsubseteq$ D omm $\mathcal{KB}$ $\cup$ {a:(C $\sqcap$ $\neg$D)} är inte satisfierad (för en ny a)
•Man börja med fakta som uttryckligen hävdats i ABox
t.ex: Alex:GladFörälder, Alex harBarn Anna
•Man använder expansionsregler för att härleda implicita fakta
t.ex: Alex:Förälder, Alex:$\forall$harBarn.(Läkare $\sqcup$ $\exists$harBarn.Läkare)
•Konstruktion misslyckas om uppenbar motsägelse (krock (**eng**. clash)) dyker upp
t.ex: Anna:Läkare, Anna:$\neg$Läkare.

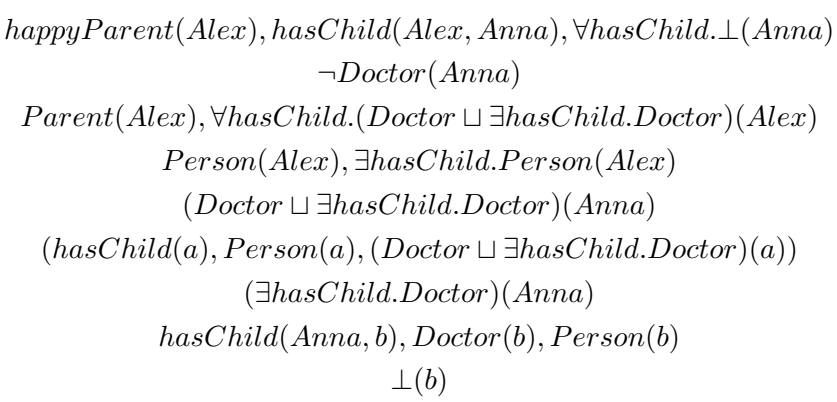
[6] En mängd av begrepp C i begrepp påstående av formen: a:C, betecknas med $con_{\mathcal{A}}(a)$ d.v.s: $con_{\mathcal{A}}(a) = \{ C|a : C \in \mathcal{A} \}$.

Ovanstående steg används för att reda ut frågan:



Nu ser vi en motsägelse.
Vi ändrar i ABox som vi har och börjar igen.

$\mathcal{T}$={Läkare $\sqsubseteq$ Person, Förälder $\equiv$ Person $\sqcap$ $\exists$harBarn.Person,
Gladförälder $\equiv$ Föräldrar $\sqcap$ $\forall$harBarn.(Läkare $\sqcup$ $\exists$harBarn.Läkare)}
$\mathcal{A}$={Alex:GladFörälder, Alex harBarn Anna, Anna:$\forall$harBarn.$\bot$}
$\mathcal{KB}$ $\models$ Anna:Läkare ?



Nu bevisades att KB$\models$ Doctor(Anna) [12].

4 Diskussion

Cutland [6, Kapitel 6] skriver att det enklaste logiska systemet som återspeglar något av matematiska resonemang är satslogik, det är ganska enkelt när den satslogiska beräkningen har definierats noggrant för att se att den är avgörbar (**eng**. decidable), med detta menar vi att det finns en effektiv procedur för att avgöra om ett påstående σ av kalkylen är allmänt (**eng**. universally) giltigt vilket betyder att det är sant i alla möjliga tolkningar.

Ett logiskt system som har ett större uttryckskraft än den satskalkylen är predikatlogiken. Det finns ett exakt begrepp om ett bevis på ett påstående av predikatlogiken så att:
Ett påstående är bevisbart om och endast om det är giltigt [7].

Cutland [6, Kapitel 6] fortsätter och skriver i sin bok att Church 1936 visade att bevisbarheten (och följaktligen giltighet) i predikatlogiken är oavgörbart (**eng**. undecidable) och använde URM (**eng**. unlimited register machine), några satser, definitioner och begrepp för sitt bevis.

Relationen mellan funktioner och predikat, definition för URM, Z(n): noll instruktion (**eng**. zero instruction), S(n): efterföljare instruktion (**eng**. successor instruction), T(m, n): överföringsinstruktion (**eng**. transfer instruction) och J(m,n,q): hopp instruktionen (**eng**. jump instruction) ges i [6, Kapitel 1].
Definition av standard form, beräkning (**eng**. computation), URM- beräkningsbar funktion ges i [6, Kapitel 1].
Church avhandling ges i [6, Kapitel 6].
Stopproblemet ges i (**eng.**Halting problem) [6, Kapitel 6].

I matematisk logik säger man att $M(x)$ är avgörbar om den karakteristiska funktionen (**eng**. characteristic function) $c_M(x)$ $(x = x_1, x_2....x_n)$ är beräkningsbar, $M(x)$ är oavgörbar om $M(x)$ inte är avgörbar.
Oavgörbart ange begränsningen av beräkningsbarheten och en algoritm för att beräkna $c_M(x)$ kallas en beslutsprocedur för $M(x)$ [6, Kapitel 1 & 2].

[7] Gödels fullständighetssats, vilket är en grundläggande teorem om matematisk logik som etablerar en korrespondens mellan semantisk sanning och syntaktisk bevisbarhet (avsnitt 2.3.2)

Nu går vi genom beviset av Church's sats:

Sats 4.1 (Church's Sats): Giltigheten av den första ordning bevisbarhet är oavgörbar.

Bevis:

Låt P vara ett program[8] i standard form med instruktioner $I_1, I_2, ..., I_s$ och låt $u = \rho(P)$. Vi använder följande symboler för predikatkalkylen,

0 en symbol för en individ,

$'$ en symbol för unär funktion (vars värde på x är $x^{'}$),

R en symbol för en (u+1)-ställig funktion,

$x_1, x_2, ..., x_u, y$ symboler för variabel individuell.

Den tolkning som vi har i åtanke är att 0 representerar numret 0, $'$ representerar funktionen x+1, R representerar det möjliga tillståndet för en beräkning under P.

Alltså om vi skriver 1 för $0^{'}$ och 2 för $0^{''}$, också vidare.

Påståendet $R(r_1, ..., r_u, k)$ där $r_1, ..., r_u, k \in \mathbb{N}$ betyder tillståndet:

$\boxed{r_1}\boxed{r_2}.....\boxed{r_u}\boxed{0}\boxed{0}$; nästa instruktion I_k

inträffar i beräkningen.

Nu för varje instruktion I_i vi kan skriva ett påstående τ_i i predikatkalkylen som beskriver effekten av I_i på tillstånden.

Symbolerna $\wedge$ (and), och $\rightarrow$ (implies), används och

(a) om $I_i = Z(n)$ låt τ_i vara påståendet

$\forall x_1 .... \forall x_u : R(x_1, ..., x_n, ..., x_u, i) \rightarrow R(x_1, ..., 0, ..., x_u, i^{'})$.

(b) om $I_i = S(n)$ låt τ_i vara påståendet

$\forall x_1 .... \forall x_u : R(x_1, ..., x_n, ..., x_u, i) \rightarrow R(x_1, ..., x_n^{'}, ..., x_u, i^{'})$.

(c) om $I_i = T(m, n)$ låt τ_i vara påståendet

$\forall x_1 .... \forall x_u : R(x_1, ..., x_m, ..., x_u, i) \rightarrow R(x_1, ..., x_m, ..., x_u, i^{'})$.

(d) om $I_i = J(m, n, q)$ låt τ_i vara påståendet

$\forall x_1 .... \forall x_u : R(x_1, ..., x_u, i) \rightarrow ((x_m = x_n \rightarrow R(x_1, ..., x_u, q)) \wedge (x_m \neq x_n \rightarrow$

[8]URM har ett oändligt antal register märkta $R_1, R_2, R_3, ...$, som var och en av dem innehåller ett naturligt nummer. Vi kan donera antalet i R_n med r_n. Innehållet i registret kan ändras med URM i förhållande till vissa instruktioner som det kan känna igen. Dessa instruktioner motsvarar mycket enkla operationer som används för att utföra beräkning med siffror. En ändlig lista med instruktioner bildar ett program. instruktionerna är av fyra slag, som följande:noll instruktion, efterföljare instruktion, överföringsinstruktion och hopp instruktion [6, kapitel 1, sidan 9].

$R(x_1, ..., x_u, i^{'}))))$.

Låt nu för varje $a \in \mathbb{N}$, σ_a vara påståendet
$(\tau_0 \wedge \tau_1 \wedge ... \wedge \tau_s \wedge R(a, 0, ..., 0, 1)) \rightarrow \exists x_1 ... \exists x_u R(x_1, ..., x_u, s+1)$,
där τ_0 är påståendet $\forall x \forall y((x^{'} = y^{'} \rightarrow x = y) \wedge x^{'} \neq 0)$, som försäkrar att om $m, n \in \mathbb{N}$ och $\mathcal{M} = \mathcal{N}$ då $m = n$.
Påståendet $R(a, 0, ..., 0, 1)$ motsvarar till en starttillstånd:
| **a** | 0 | 0 |...; nästa instruktion I_1,
Och varje påstående $R(x_1, ..., x_u, s+1)$ motsvarar ett stoppläge (eftersom det finns ingen instruktion I_{s+1}). Således vi kan se att
(*) $P(a) \downarrow$[9] $\Leftrightarrow$ σ_a är giltig.

Anta först att $P(a) \downarrow$, och att vi har en struktur där $\tau_0, ..., \tau_s$ och $R(a, 0, ..., 0, 1)$ håller. Att använda påstående $\tau_0, ..., \tau_s$ vi finner att alla påstående $R(r_1, ..., r_u, k)$ motsvarande de successiva tillstånden i beräkningen håller också. Så småningom finner vi det upphörande påståendet $R(b_1, ..., b_u, s+1)$ håller för vissa $b_1, ..., b_u \in \mathbb{N}$, och därmed $\exists x_1 ... \exists x_u R(x_1, ..., x_u, s+1)$ håller. Sålunda σ_a är giltigt.

Omvänt, om σ_a är giltigt, gäller det i synnerhet i strukturen $\mathbb{N}$ med predikatsymbol R tolkad av predikatet R_a där
$R_a(a_1, ..., a_u, k)$= i någon tillstånd i beräkningen $P(a)$ innehåller registren $a_1, a_2, .., a_u, 0, 0, ...$ och nästa instruktion är I_k.

Då $\tau_0, ..., \tau_s$ och $R(a, 0, ..., 0, 1)$ alla håller i den här strukturen, så gör det också $\exists x_1 ... \exists x_u R(x_1, ..., x_u, s+1)$. Därför $P(a) \downarrow$.
Om vi tar P till att vara ett program som beräknar funktionen $\psi_U(x, x)$, ekvivalensen (*) ger en reduktion av problemet "$x \in W_x$" till problemet "σ är giltig". Därför är det senare oavgörbart [6, Kapitel 6].

Lite förklaring för beviset:
Ett universalprogram innebär program som på ett sätt förkroppsliga alla andra program. Betrakta funktion $\psi(x, y)$ som defineras med $\psi(x, y) \simeq \phi_x(y)$.

[9] Låt $a_1, a_2, a_3, ...$ vara en oändlig sekvens från $\mathbb{N}$ och låt P vara ett program, vi kommer att skriva:
$P(a_1, a_2, a_3, ...)$ för beräkning under P med initial konfiguration och $P(a_1, a_2, a_3, ...) \downarrow$ betyder att beräkningen så småningom slutar [6, kapitel 1, sidan 17].

Den enkla funktionen ψ innehåller alla unära beräkningbara funktioner ψ_0,ψ_1,ψ_2,... sedan för någon speciell m, funktionen g som ges med:
$g(y) \simeq \psi(m, y)$
är bara den beräkningsbara funktionen ϕ_m, vilket definieras rent generellt som:

Definition:
En universal funktion för n-ställiga beräkningbara funktioner är den (n+1)-ställiga funktionen $\psi_U^{(n)}$ som definieras med:
$\psi_U^{(n)}(e, x_1, ..., x_n) \simeq \phi_e^{(n)}(x_1, ..., x_n)$.

Sats:
"$x \in W_x$" (eller ekvivalent, "$\phi_x(x)$" är definierad, eller $P_x(x) \downarrow$ ", eller "$\psi_U(x, x)$" är definierad) är oavgörbart [6, kapitel 5].

$\psi_U(x, x)$ är funktionen som definieras i Cutland [6, kapitel 5] genom ett universalprogram $P = I_1, \ldots, I_M$. Från $I_1, .., I_M$ konstruerar man sedan (effektivt) formlerna $\tau_1, \ldots, \tau_M$ och därmed σ_x. Då gäller σ_x giltig omm $\psi_U(x, x)$ definierad omm $x \in W_x$. Det oavgörbara problemet $x \in W_x$ har reducerats till problemet σ_x. Därmed kan σ_x inte vara avgörbart [6, kapitel 6].

Baader, Horrocks & Sattler [5, Kapitel 3.3.1] påstår att DLs kan ses som ett fragment av första ordningens predikatlogik. Den främsta anledningen till att använda DLs snarare än generell första ordningens predikatlogik när man representerar kunskap är det att DLs faktiskt är ett avgörbart fragment av första ordningens predikatlogik.

Vi tittar på rollnamn som binära relationer och begrepps-namn som unära relationer, definierar vi två översättningsfunktioner, π_x and π_y, den induktivt avbildar $\mathcal{ALC}$-begrepp in i första ordningens predikatlogik med fria variabler, x eller y:

$$\begin{array}{ll}
\pi_x(A) = A(x), & \pi_y(A) = A(y), \\
\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D), & \pi_y(C \sqcap D) = \pi_y(C) \wedge \pi_y(D), \\
\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D), & \pi_y(C \sqcup D) = \pi_y(C) \vee \pi_y(D), \\
\pi_x(\exists r.C) = \exists y.r(x, y) \wedge \pi_y(C) & \pi_y(\exists r.C) = \exists x.r(y, x) \wedge \pi_x(C)
\end{array}$$

$$\pi_x(\forall r.C) = \forall y.r(x,y) \Rightarrow \pi_y(C) \quad \pi_y(\forall r.C) = \forall x.r(y,x) \Rightarrow \pi_x(C)$$

Med tanke på detta kan vi översätta en TBox $\mathcal{T}$ och en ABox $\mathcal{A}$ som följer, där $\psi[x/a]$ betecknar formeln erhållen från ψ genom att ersätta alla fria förekomster av x med a:

$$\pi(\mathcal{T}) = \bigwedge_{C \sqsubseteq D \in \mathcal{T}} \forall x.(\pi_x(C) \Rightarrow \pi_x(D))$$

$$\pi(\mathcal{A}) = \bigwedge_{a:C \in \mathcal{A}} \pi_x(C)[x/a] \wedge \bigwedge_{(a,b):r \in \mathcal{A}} r(a,b).$$

Baader, Horrocks & Sattler [5, Kapitel 3.3.1] menar att denna översättning bevarar semantiken: vi kan självklart visa DL-tolkningar som första ordningens tolkningar och vice versa, och det är lätt att visa att översättningen bevarar modeller. Som en lätt följd har vi den resonemanget i DLs motsvarar första ordningens slutledning:

Sats 4.2:
Låt $(\mathcal{T}, \mathcal{A})$ vara en $\mathcal{ALC}$-KB, C, D möjliga komplexa begrepp, a ett individnamn. Då
1. $(\mathcal{T}, \mathcal{A})$ är konsistent om och endast om $\pi(\mathcal{T}) \wedge \pi(\mathcal{A})$ är konsistent,
2. $(\mathcal{T}, \mathcal{A}) \models$ C $\sqsubseteq$ D om och endast om $(\pi(\mathcal{T}) \wedge \pi(\mathcal{A})) \Rightarrow (\pi(C \sqsubseteq D))$ är giltig,
3. $(\mathcal{T}, \mathcal{A}) \models$ a: C om och endast om $(\pi(\mathcal{T}) \wedge \pi(\mathcal{A})) \Rightarrow (\pi(a : C))$ är giltig [5, Kapitel 3.3.1].

Tolkningen tillhandahåller inte bara ett alternativt sätt att definiera semantiken för $\mathcal{ALC}$, men berättar också att alla de introducerade resonemangproblemen (**eng.** Reasoning problem) för $\mathcal{ALC}$ kunskapsbaser är avgörbara. I själva verket använder översättningen av en kunskapsbas endast variablerna x och y och ger således en formel i de två-variabel-fragmentet av första ordningens logik, vilken är känt att vara avgöbart i icke-deterministisk exponentiell tid[10] [5, Kapitel 3.3.1].

[10]I datavetenskap är tidskomplexiteten den beräkningskomplexitet som beskriver hur lång tid det tar att köra en algoritm och en algoritm sägs vara exponentiell i tid, om T(n) är övre avgränsad av $2^{poly(n)}$, där poly(n) är något polynom i n.

Gräder, Kolaitis & Vardi [7] skriver att när satisfierbarhetsproblemet för första ordningens logik visades vara oavgörbart,(Church's Sats, Sats 4.3), inledde logiker ett ambitiöst projekt som syftar till att avgränsa gränsen mellan avgörbara och oavgörbara fragment av första ordningens logik. I det här projektet var huvudfokus på prefix-klasser (**eng**. prefix classes) och prefix-vokabulärklasser (**eng**. prefix-vocabulary classes).

T.ex: AEA-klassen består av alla relationella första ordning satser med kvantifierarprefix av formen $\forall\exists\forall$. Efter ett hårt arbete på detta projekt i nästan femtio år kunde forskarna identifiera gränsen mellan avgörbart och oavgörbart för alla prefix vokabulär klasser av första ordningens formler. Ett annat sätt att erhålla syntaktiska fragment av första ordningens logik är att indela formlerna enligt deras variabler. Mer exakt, första ordningens logik med k variabler " FO^k " består av alla relationella första-ordningsformler som högst innehåller k olika individuella variabler, $k \geq 1$ [7].

Ovannämnda klassen AEA finns i FO^3 och eftersom satisfierbarhets-problem för AEA-klassen är oavgörbar, då det följer att FO^3 även utan likhet[11] är oavgörbart.

Gräder, Kolaitis & Vardi [7] skriver att det första avgörbarhets-resultatet för FO^2 erhölls av Scott[12], vilken visade att beslutsproblem för FO^2 kan reduceras till Gödel-[13]klassen (det är klassen av relationella första ordningens satser med kvantitativa prefix av formen $\exists^*\forall\forall\exists^*$, en sträng bestående av ett godtyckligt antal existentiella kvantifikatorer följd av exakt två universella kvantifikatorer följt av ett godtyckligt antal existentiella kvantifikatorer). Men problemet är att Gödel-klassen utan likhet är avgörbar och så är Scott klassen, och täcker inte fallet med FO^2 med likhet.

Hela klassen FO^2 betraktades av Mortimer [8], han visade att den här klassen är avgörbar genom att visa att den har ändliga modellegenskapen,

[11]Ett alternativt förhållningssätt ansåg att likhet förhållandet är en icke-logisk symbol. Denna konvention kallas första ordningens logik utan likhet.

[12]Dana Stewart Scott (född 11 oktober 1932) är Emeritus Hillman University Professor i datavetenskap, filosofi och matematisk logik på Carnegie Mellon University.

[13]Kurt Friedrich Gödel var en Österrikisk, och senare Amerikansk, logiker, matematiker och filosof.

(vilket betyder att varje satisfierad sats i Gödel-klassen har en ändlig modell). En analys av hans bevis visar att han faktiskt etablerade en begränsad modellegenskap för FO^2, om en FO^2-sats är satisfierad, då är det satisfierad i en modell vars storlek är dubbelt exponentiell i längden av formeln α, följer det att satisfierbarhet av FO^2 med likhet är avgörbart i icke-deterministisk dubbel exponentiell tid[14].

Gräder, Kolaitis & Vardi [7] hävdar att om en sats α i Scott klassen (att beslutsproblem för FO^2 kan reduceras till Gödel-klassen) är satisfierad då är den satisfierad i en modell vars storlek är högst exponentiell i storleken av α vilket kallas exponentiell modell-egenskap för Scott klass.

Anta att α är en sats i FO^2 med individuella variabler x och y, låt s vara storleken av α, det är längden på en sträng som kodar α över något fixt alfabet. Vi kommer att konstruera i polynomiell tid en FO^2 sats $\alpha^{'}$ med följande egenskaper:
1. α är satisfierande om $\alpha^{'}$är satisfierande, Vidare, för varje ändlig modell av α' det finns en ändlig modell av α av samma kardinalitet.
2. Varje relation symbol som uppträder i $\alpha^{'}$ har ställighet högst 2.
3. $\alpha^{'}$ innehåller $\mathcal{O}(s/\log s)$ olika relations-symboler och är av storlek $\mathcal{O}(s)$ [7].

Nu behöver vi bara en hjälpsats och en sats innan huvudsatsen bevisas.

Hjälpsats:
$\alpha^{'}$ är satisfierad om α^{*} är satisfierad, Vidare för varje ändlig modell av $\alpha*$ finns en ändlig modell av α' av samma kardinalitet.

Observera att om $\alpha^{'}$ är av storlek s, då innehåller $\mathcal{O}(s)$ olika relations-symboler och är av storlek $\mathcal{O}(s \log (s))$. Faktum är α' kan innehålla upp till $\mathcal{O}(s)$ del-formler så vi behöver $\mathcal{O}(s)$ relations-symboler. Den extra log(s)-faktorn i storleken på α^{*} beror på det faktum att vi behöver ett index av storlek $\mathcal{O}(s)$ för dessa olika relations-symboler.

Sats 4.3:
Låt θ vara en sats i Scott klassen. Om θ är satisfierad, då har den en ändlig

[14]En algoritm sägs vara dubbel exponentiell i tid om T(n) är övre avgränsad av $2^{2^{poly(n)}}$, där poly(n) är något polynom i n.

modell med högst 3s2^r element där s är storleken på θ och r är antalet relationer symboler som förekommer i θ.
Och slutligen har vi:

Sats 4.4:
FO^2har exponentiell modellegenskap: det finns en konstant c så att varje satisfierad FO^2 sats har högst en modell av kardinalitet 2^{cs} där s är storleken på α.
Bevis:
Givet en FO^2 sats α kan vi reducera den i polynomiell tid[15] till en sats α^* i Scott klassen sådan att α är satisfierad om och endast om α^* är satisfierad. Dessutom: för varje ändlig modell av α^* finns en ändlig modell av α av samma kardinalitet. Som tidigare visades om α är av storlek s då är α^* av storlek $\mathcal{O}(s \log s)$ och har högst s olika relations symboler. Med sats 4.3, om $\alpha*$ har en modell då har den en modell av kardinalitet $\mathcal{O}(s \log s2^s) = 2^{\mathcal{O}(s)}$ [7].

[15]En algoritm sägs vara av polynomiell tid om dess löptid är övre avgränsad av ett polynom uttryck i storleken av indata för algoritmen.

5 Slutsats

Semantisk webb syftar till maskinförståeliga webbresurser, vars information sedan kan delas och bearbetas både med automatiserade verktyg, till exempel sökmotorer och av mänskliga användare (agenter). Denna delning av information mellan olika agenter kräver semantisk markering, det vill säga en kommentar på webbsidan med information om dess innehåll som förstås av de agenter som söker på webben. En sådan kommentar kommer att ges på ett standardiserat, uttrycksfullt språk (som till exempel ger booleska operatörer och någon form av kvantifiering (logiken)) och använder sig av vissa termer (som "Human", Plant", etc.). För att säkerställa att olika agenter har en gemensam förståelse av dessa termer, behöver man ontologier där dessa termer beskrivs, och som således skapar en gemensam terminologi mellan agenter. Slutligen bör dess uttrycksfulla kraft vara tillräcklig, dvs. språket bör vara tillräckligt uttrycksfullt för att definiera relevanta begrepp i tillräcklig detalj, men inte för uttrycksfullt för att göra resonemanget omöjligt [11].

Logiken tar inte hänsyn till vad som är sant. Logik är studie av vad som följer av vad, det vill säga slutsatser följer från en mängd premisser. Det kan definieras som studie av principer för korrekt resonemang. Det viktigaste med logik är principer som styr argumentets validitet och kontrollerar om en viss slutsats följer av ett givet antagande. T.ex:

Lars gillar alla som gillar IK
Lena gillar IK

Lars gillar Lena.

Är detta argument giltigt? Hur vet vi det?
Logikprocessen tar in en viss information som kallas premisser och producerar några resultat som kallas slutsatser.

Precis som det framkommer i rapporten på slutet av avsnitt (1.2) den enklaste typen av logik är satslogik vilken har beskrivits i avsnitt (2.2) tillsammans med några relevanta definitioner och satser samt ett bevissystem. Även om en del kan utföras med satslogik är detta inte tillräckligt för att

kunna beskriva världen och ontologi i en semantisk webb, då man inte har tillgång till atomsatser. En atomär formel i satslogik är bara påstående som kan vara sant eller falsk och saknar inre struktur.
Till exempel:

- Vägen blir blöt om det regnar.
- Om månen är gjord av ost, så kan kor flyga.
- Om Oliver är kär så blir han glad.

Ett problem i satslogik är att man bara kan göra påstående om ett enda objekt, man kan inte sammanfatta saker i en mängd, klass och göra uttalanden om ett antal eller en mängd av saker. Världen består av objekt och egenskaper som skiljer ett objekt från en annan därför behövde vi gå ett steg längre och i avsnitt (2.3) visades att predikatlogiken tolkar en atomär formel som ett påstående om relationen mellan olika objekt.
Nu tittar vi på följande påstående :

- Anna är kvinna.
- Sten är man.
- Anna och Sten är kusiner.

I satslogik är de bara atomära påståenden:

- Anna-är-kvinna.
- Sten-är-man.
- Anna-och-Sten-är-kusiner.

Medan i predikatlogikens atomära påståenden kan vi använda predikat med konstanter som argument:

- Kvinna(Anna)
- Man(Sten)
- Kusiner(Anna,Sten)

Det går att påstå att satslogik är ett svagt språk i jämförelse med predikatlogik. Det är svårt att identifiera individerna. t.ex:
Anna, 17,.....
det kan inte direkt beskriva egenskaper hos individer. t.ex:
Anna har långt hår.
Generaliseringar, mönster, regelbundenhet kan inte lätt representeras. T.ex:
Alla rektanglar har fyra sidor.
Predikatlogik är däremot uttrycksfull nog för att konsekvent och kortfattad representera och formulera alla KB eller DB (databas). T.ex:
Alla hundar är svarta: $\forall x(hund(X) \rightarrow svart(X))$.
Det finns gula katter: $\exists x(katt(X) \wedge gul(X))$.

Man kan säga att predikatlogikens kvantifierare tillåter påståenden om en mängd av objekt utan att namnge objekten explicit. Predikatlogik passar perfekt för beskrivningen av ontologier vilket var målet med att tillämpa logiken som vi beskrev i avsnitt 1.1. Predikatlogik är för uttrycksfull vilket är bra men är för voluminös för modellering och har för komplicerade bevis för att kunna utföras automatiskt. Med tanke på dessa problem var man tvungen att hitta på något lämpligt fragment av predikatlogik.

I matematisk logik och datavetenskap är tvåvariabel logik fragmentet av första ordningens logik där formeln kan skrivas med endast två olika variabler som kallas DL och studeras utan funktionssymboler. I logik hänvisar termen avgörbar till beslutsproblemet. Frågan om förekomsten av en effektiv metod för att bestämma medlemskap i en mängd formler eller mer exakt en algoritm som kan och kommer att returnera ett booleskt sant eller falskt värde som är rätt vilket man kan återfinna i DL ALC. Predikatlogik kan däremot lösa på obestämd tid, krascha eller returnera ett felaktig svar. Detta beror på att predikatlogik har en grov model.

DL ALC är en beskrivning av fakta som kan hanteras av en algoritm. Effektiva algoritmer som kan avgöra en mängd fakta. DL ALC har en metod som är avgörande. Detta beror på att DL ALC är relaterad till modal logiken som är avgörbar.

6 Referenser

[1] **Hasso Plattner Institute**.1998
Semantic Web Technologies Germany
Tillgänglig: https://open.hpi.de/courses/semanticweb# .
Harald Sack undervisar semantic web technologies på openhpi.
OpenHPI är en plattform för massiva öppna online kurser inom datavetenskap och informationsteknik. Det är värd vid Hasso Plattner-institutet i Potsdam. Hasso Plattner-institute, förkortat HPI, är ett tyskt informatik institut och fakultet vid universitetet i Potsdam beläget i Potsdam nära Berlin, Tyskland. Hämtad: (2018-08-20)

[2] **Carlström, Jesper.** 2009
Logik Stockholms Universitet.

[3] **Smullyan, Raymond M.** 1995
First-order logic. Corrected reprint of the 1968 original. Dover Publications, Inc., New York.

[4] **Baader, Franz. Horrocks, Ian. Lutz, Carsten. Sattler, Uli.** 2017
An Introduction To Descpription Logic Cambridge university press. Cambridge

[5] **Baader, Franz. Horrocks, Ian., Sattler, Uli.** 2008
Description Logic sid. 135-179
Ur: *Handbook of knowledge representation*, 1st ed. Edited by Porter, Bruce, Lifschitz, Vladimir. & Van Harmelen, Frank. Elsevier, Amsterdam

[6] **Cutland, Nigel J.** 1997
Computability Cambridge University Press

[7] **Grädel, Erich. Kolaitis, Phokion G. Vardi, Moshe Y.** 1997
On the decision problem for two-variable first-order logic Bull. Symbolic Logic 3, no. 1, 53–69.

[8] **Mortimer, Michael.** 1975
On languages with two variables Z. Math. Logik Grundlagen Math. 21, 135–140

[9] wikipedia. 2017
Valuation(logic)
Tillgänglig: https://en.wikipedia.org/wiki/Valuation_(logic)
Hämtad: (2018-09-30)

[10] webopedia. 2018
Semantic Web
Tillgänglig: https://www.webopedia.com/TERM/S/Semantic_Web.html
Hämtad (2018-09-30)

[11] **Baader, Franz. Horrocks, Ian., Sattler, Ulrike**
Descriptin Logics as Ontology Languages for the Semantic Web
Theoretical Computer Science, RWTH Aachen, Germany
Department of Computer Science, University of Manchester, UK
Tillgänglig: https://lat.inf.tu-dresden.de/research/papers/2005/BaSaJS60.pdf
Hämtad:(2019-10-21)

[12] **Harrocks, Ian**
A Formal Foundation for Ontology Languages and Tools Part 2: Tools [Slides], Information Systems Group, Oxford university, Computing laboratory.
http://www.cs.ox.ac.uk/ian.horrocks/Seminars/download/Horrocks_Ian_pt2.pdf
Hämtad:(2019-03-14)

A Ordlista

Svenska	Engelska	Sida
ARPANET	Advanced Research Project Agency Network	2
Atomär formel	Atomic formula	6
ABox (Påstående rutan)	ABox (assertional box)	21
AL (Attributspråk)	AL (Attributive language)	22
ALC (Attributspråk med komplement)	ALC (Attributive language with complement)	23
Avslutad	Teminated	39
Allmänt	Universally	42
Analytiska tablåer	Analytic tableaux	8
Avgörbar	Decidable	42
Beskrivningslogik	Description logic	3
Biimplikation	Equivalence	5
Bunden	bound	15
Begrepp	Concept	23
Boolesk evaluering	Boolean valuation	9
Boolesk algebra	Boolean algebra	5
Begreppspåstående	Concept assertion	29
CWA	Close world assumption	39
Deduktion	Deduction	8
Egenskap	Property	4
Existential kvantifierare	Existential quantifier	13
Existentiell begränsning	Existential restriction	23
Efterföljare instruktion	Successor instruction	42
Förekomst	Occurrence	13
Fullständighetssatsen	Completeness theorem	8
Giltig	Valid	18
GCI (Generell begreppinkludering)	GCI (General concept inclusion)	27
HTML	Hyper Text Markup Language	3
Hopp instruktion	Jump instruction	42
Implikation	Implication	5
Karakteristisk funktion	Characteristic function	42
Krock	Clash	34
Konceptualisering	Conceptualization	22
KR (Kunskapsrepresentation)	KR (Knowledge representation)	21
Mättad	Saturated	9
Noll-instruktion	Zero instruction	42
Naturlig deduktion	Natural dedection	8
NNF	Negation normal form	35
Omfång	Extension	24
Oavgörbar	Undecidable	42
Omfattning	Scope	15
OWA	Open world assumption	30
OWL	Web Ontology Language	4
Omfång	Extention	24
Prefix-klasser	Prefix classes	46
Prefix-vokabulärklasser	Prefix-vocabulary classes	46
Påståendevariabel	Propositional variable	16
Premiss	Premise	8
Rollpåstående	Role assertion	29
Ren formel	Pure formula	16
RDFS	Resource Description Frame Work Schema	3
Resonemangproblem	Reasoning problem	45
Stängda	Closed	12
Sundhetssatsen	Soundness theorem	8
Sluten	Completed	12
Substitution	Substitution	14
Semantiska webben	Semantic web	3
Sanningsfunktionellt	Truth-functionally	9
Tolkning	Interpretation	7
TBox (Terminologiska rutan)	TBox (Terminological box)	21
Universella kvantifierare	Universal quantifier	13
URM	Unlimited register machine	42
Utfyllnad	Filler	24
UNA	Unique name assumption	30
XML	Extensible Markup Language	3
Överföringsinstruktion	Transfer instruktion	42