# LOGIKMODELLERING

En kortfattad inledning till inlämningsuppgiften

för DSV1:L samt 2I1032

Love Ekenberg
DSV, 2003-2004

# Inledning

Denna handbok är en mycket kortfattad introduktion till hur du kan använda predikatlogik för modellering och evaluering av informationssystem. Boken består av två delar:

- en beskrivning av predikatlogik som modellerings- och analysverktyg
- ett litet exempel på hur du kan använda predikatlogik för att modellera och analysera ett informationssystem

# I. Terminologi och arbetssätt

Denna del går igenom begrepp och idéer som du behöver känna till för att kunna modellera och analysera informationssystem genom att använda predikatlogik.

# Konceptuell Modellering

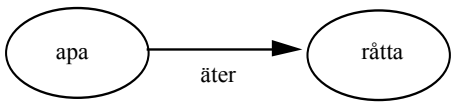
Det finns en stor mängd formalismer för att beskriva olika aspekter av ett system och valet av formalism betingas naturligtvis av olika faktorer. Inte sällan är ett systems funktionsspecifikation uttryckt i någon konceptuell modell. En sådan modell kallas ofta ett *konceptuellt schema* och beskrivs vanligen i ett bestämt språk med en fastställd och exakt syntax såväl som en entydig semantik. Ett språk som äger dessa egenskaper, och som man därför ofta använder för att uttrycka konceptuella modeller, är första ordningens logik. Så skulle man exempelvis kunna uttrycka att alla apor som äter råttor är förnuftiga med formeln:

$\forall x \forall y[(\text{apa}(x) \wedge \text{råtta}(y) \wedge \text{äter}(x,y)) \rightarrow \text{förnuftig}(x)]$.

Genom användning av formler av denna typ kan man bygga upp en modell av ett system. Denna kan sedan analyseras på olika sätt, exempelvis genom att man bevisar egenskaper hos systemet.

Ett system modellerat uteslutande med formler kan te sig en smula svåröverskådligt. Detta kan till stor del överbryggas genom att man introducerar en grafisk notation för vissa typer av formler. Så är innebörden av det konceptuella schemat i figur 1–1 exakt densamma som innebörden av formeln:

$\forall x \forall y[\text{äter}(x,y) \rightarrow (\text{apa}(x) \wedge \text{råtta}(y))]$.




**figur 1–1**

Ovalerna i figuren korresponderar mot klasser av objekt i det system man avser att modellera. Pilen, det vill säga den riktade kanten, representerar en relation eller ett förhållande mellan objekt i de respektive klasserna. Denna representation har inte någon särskilt stark uttryckskraft och är därför något begränsat. Inte desto mindre är det oftast mycket lättare att se hur olika objektsklasser och relationer hör samman. Genom att använda grafisk notation kombinerat med textuella formler kan man dessutom åstadkomma ett väsentligt förbättrat uttryckssätt.

Som exempel på ett verktyg, som delvis utnyttjar en grafisk representation av första ordningens logik, kan nämnas UML. Emellertid är möjligheterna att utföra olika typer av konsistenskontroller starkt begränsade.

En konceptuell modell innehåller vanligen en statisk och en dynamisk del, där den förstnämnda uttrycker statiska egenskaper hos modellen (det statiska schemat) och den senare uttrycker tillståndsövergångar (hur tillståndsbeskrivningar för scheman förändras). Den statiska delen av en modell fäster således inget avseende vid dess händelseregler, det vill säga den uttrycker vilka entiteter som möjligen kan existera och hur dessa förhåller sig till varandra. I princip hanterar därför den statiska delen grafer med ytterligare textuella formler, vilka är uppbyggda av predikat och funktioner som är definierade i grafen. Den dynamiska delen av en specifikation beskriver tillståndsövergångar som är initierade av händelser i systemet eller i dess omedelbara omgivning. Denna introduktion fokuserar främst på den statiska delen av ett konceptuellt schema.

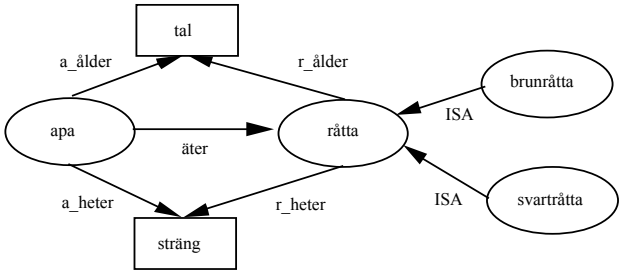
Ett konceptuellt schema kan således befinna sig i olika tillstånd. Dessa kan uttryckas med *diagram* [1] som ska vara förenliga (det vill säga konsistenta) med det konceptuella schemat. Informellt kan man säga att ett diagram uttrycker vad som är fallet i ett visst tillstånd och ett exempel på ett diagram som är förenlig med schemat i figur 1–1 är {`apa(ulf)`, ¬`apa(carl)`, `råtta(carl)`, ¬`råtta(ulf)`, `äter(ulf,carl)`, ¬`äter(ulf,ulf)`, ¬`äter(carl,ulf)`, ¬`äter(carl,carl)`}. Detta uttrycker att *ulf* är en apa, men inte en råtta, och att *carl* är en råtta, men inte en apa. Dessutom gäller det att *ulf* äter *carl*, men inte tvärtom, och att ingen äter sig själv. Detta diagram är konsistent med det konceptuella schemat, dvs det strider inte mot regeln som utgör det konceptuella schemat. I motsats till detta strider {`apa(ulf)`, ¬`apa(carl)`, `råtta(carl)`, ¬`råtta(ulf)`, ¬`äter(ulf,carl)`, `äter(ulf,ulf)`, ¬`äter(carl,ulf)`, ¬`äter(carl,carl)`} mot det konceptuella schemat. Här är *ulf* en apa som äter sig själv, medan schemat säger att om något blir ätet så är det en råtta.

# MODELLERING

För att underlätta logikmodellering så brukar man dela upp objektklasser i olika kategorier.

## Entitetstyper, relationer och datatyper

I figur 1–2 kan du se tre typer av symboler. Ovalerna symboliserar enställiga predikat eller *entitetstyper*. Pilarna symboliserar tvåställiga predikat eller *relationer*. Rektangeln symboliserar *datatyper*, vilka är enställiga predikat av vissa typer.




**figur 1–2**

Intuitivt uttrycker schemat i figur 1–2 att apor kan äta råttor. Såväl apor som råttor har en ålder och ett namn. `apa(a1)` är ett exempel på ett enställigt predikat och `äter(a1,r1)` är ett tvåställigt predikat. `a1` är här en *instans* av entitetstypen *apa* och `<a1, r1>` är en instans av relationen *äter*. Ställigheten anger således antalet argument till predikatet.

[1] Med ett diagram avses här en negationsfullständig Herbrandmodell. En sådan består i princip av en mängd instansierade predikat som representerar vad som gäller samt negerade predikat som representerar vad som inte gäller.

Det finns exakt två sorters datatyper: *sträng* och *tal*. Datatypen *sträng* i schemat representerar teckensträngar i det representerade systemet och datatypen *tal* representerar tal i systemet. Så kan exempelvis `ulf`, som är en instans av datatypen sträng, representera namnet *Ulf* i systemet. Notera här att namnet *Ulf* naturligtvis inte är detsamma som apan Ulf. Denna skillnad yttrar sig bland annat i att namnet *Ulf* består av tre tecken under det att apan Ulf består av hår, benstomme, muskler, etc. På motsvarande sätt representeras *talet tretton* i systemet av instansen `13`. Den sistnämnda är en instans av datatypen *tal*.

*ISA-relationerna* i schemat uttrycker att entitetstyperna *brunråtta* och s*vartråtta* är s*ubtyper* till *supertypen råtta*. Den exakta innebörden av detta uttrycks enklast i logik med följande formler:

$\forall x[\text{brunråtta}(x) \rightarrow \text{råtta}(x)]$

$\forall x[\text{svartråtta}(x) \rightarrow \text{råtta}(x)]$

Intuitivt innebär detta helt enkelt att om något är en brunråtta eller en svartråtta så är det en råtta.

# Informationsbas

En informationsbas representerar i princip ett *tillstånd* i systemet eller med andra ord vad som är fallet i systemet. Det som ej är explicit angivet är underförstått negerat. En informationsbas som korresponderar mot schemat i figur 1–2 kan se ut som i figur 1–3.

```
apa(a1).
apa(a2).
brunråtta(r1).
svartråtta(r2).
a_heter(a1,ulf).
a_heter(a2,britt).
r_heter(r1,carl).
r_heter(r2,per).
a_ålder(a1,2).
a_ålder(a2,3).
r_ålder(r1,4).
r_ålder(r2,4).
äter(a1,r1).
äter(a1,r2).
äter(a2,r1).
äter(a2,r2).
```

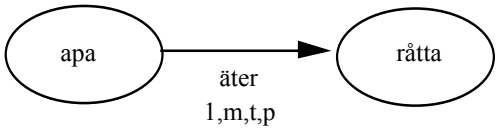
**figur 1–3**

Informationsbasen representerar att det finns två apor, en brunråtta samt en svartråtta i systemet. Aporna heter *Ulf* respektive *Britt* och de är två respektive tre år gamla. Råttorna heter *Carl* respektive *Per* och de är båda fyra år gamla. Det gäller även att båda aporna äter samtliga råttor i systemet.

Exempelvis gäller nu även att instansen *r1* inte har någon relation till strängen *per*, dvs ¬r_heter(r1,per), men sådana negationer är som sagt underförstådda i en informationsbas.

# Avbildningsregler

Det grafiska schemat kan även representera *avbildningsregler*. Dessa uttrycker ytterligare egenskaper hos relationerna. Det grafiska schemat i figur 1–4 innehåller avbildningsregeln (1,m,t,p).




**figur 1–4**

Avbildningsregeln i figuren har följande innebörd:

1: anger att en apa får äta högst en råtta.

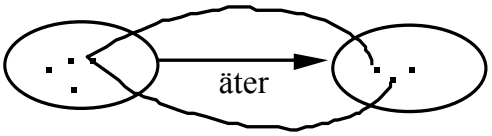
m: anger att flera apor kan äta samma råtta, vilket inte behöver representeras i logik.

t: anger att alla apor äter någon råtta.

p: anger att inte alla råttor måste ätas av någon apa, vilket inte behöver representeras i logik.

Generellt gäller följande för avbildningsregler på formen (1,2,3,4):

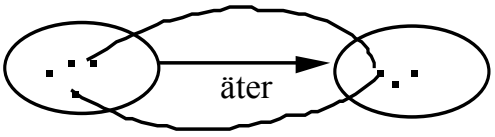
1: anger huruvida relationen måste vara envärd (1) eller flervärd (m). En envärd relation avbildar ett element i domänen på högst ett element i värdeförrådet. Behöver detta inte gälla är relationen flervärd.




**figur 1–5 relationen *äter* är flervärd**

- Avbildningsregeln *1* på position 1 har således samma innebörd som formeln $\forall x \forall y \forall z[(\text{äter}(x,y) \wedge \text{äter}(x,z)) \rightarrow y=z]$.
- Observera att avbildningen *m* tillåter allt och inte behöver översättas.

2: anger huruvida relationen är injektiv (1) eller icke-injektiv (m). En injektiv relation avbildar högst ett element i domänen på varje element i värdeförrådet. Behöver detta inte gälla är relationen icke-injektiv.

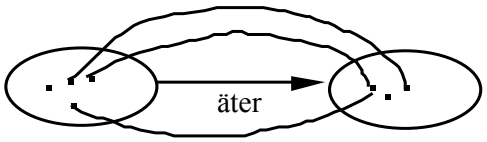



**figur 1–6 relationen *äter* är icke-injektiv**

- Avbildningsregeln *1* på position 2 har således samma innebörd som formeln $\forall x \forall y \forall z[(\text{äter}(y,x) \wedge \text{äter}(z,x)) \rightarrow y=z]$.

- Observera att avbildningen *m* igen tillåter allt och inte behöver översättas.

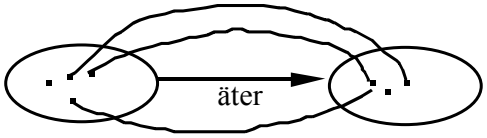
3: anger huruvida relationen är total (t) eller partiell (p). En total relation avbildar varje element i domänen på något element i värdeförrådet. Behöver detta inte gälla är relationen partiell.




**figur 1–7 relationen *äter* är partiell**

- Avbildningsregeln *t* på position 3 har således samma innebörd som formeln $\forall x \exists y[apa(x) \rightarrow (äter(x,y) \wedge råtta(y))]$.
- Observera att avbildningen *p* tillåter allt och inte behöver översättas.

4: anger huruvida relationen är surjektiv (t) eller icke-surjektiv (p). Varje element i värdeförrådet till en surjektiv relation är en avbild av något element i domänen. Behöver detta inte gälla är relationen icke-surjektiv.




**figur 1–8 relationen *äter* är icke-surjektiv**

- Avbildningsregeln *t* på position 4 har således samma innebörd som formeln $\forall x \exists y[råtta(x) \rightarrow (äter(y,x) \wedge apa(y))]$.
- Observera att avbildningen *p* tillåter allt och inte behöver översättas.

# Händelseregler och informationsprocessorn

Den dynamiska delen av ett schema representerar hur tillståndsförändringar i systemet sker. Detta görs med *händelsetyper* eller *händelseregler*. En händelsetyp kan instansieras med ett händelsemeddelande vilket resulterar i att en händelse exekveras. Det som hanterar händelser i ett informationssystem brukar ibland kallas *informationsprocessorn*. Denna handhar uppdateringar av informationsbasen. Den utför även kontroller av motsägelsefrihet samt gör det möjligt att besvara ställda frågor till ett konceptuellt schema.

I denna introduktion kommer inte händelseregler att behandlas i större detalj.

## Verifiering av det konceptuella schemat

Det konceptuella schemat kan analyseras på en mängd olika sätt. Ett av dessa är att studera om en given kravspecifikation följer logiskt från den statiska delen av schemat. Det senare innebär ju att varje tillstånd som är förenligt med schemat även ska vara förenligt med kraven i kravspecifikationen. Med andra ord gäller det att alla möjliga tillstånd för schemat ska vara förenliga med kraven. Det får alltså inte finnas ett tillstånd som systemet kan befinna sig i som inte även är ett tillåtet tillstånd visavi kraven som gäller för systemet enligt kravspecifikationen.

Detta förutsätter naturligtvis att även kraven är formulerade i predikatlogik.

Den logiska följden kan exempelvis bestämmas med resolutionsmetoden. I praktiken använder man ofta en automatisk teorembevisare för detta ändamål. Även ett språk som Prolog kan användas för detta. Avsnittet nedan ger en kortfattad introduktion till Prolog.

# EN NOT OM PROLOG

Något slarvigt kan man säga att de analyser som man utför på konceptuella scheman huvudsakligen utgörs av olika konsistenskontroller och kontroller huruvida vissa formler följer av den konceptuella modellen. Formellt sett är därför de tekniker man utnyttjar för att kunna utföra sådana kontroller baserade på teorembevisning. En önskvärd egenskap i detta sammanhang är att teorembevisningen ska kunna göras med hjälp av en dator i en serie på förhand bestämda steg. Av olika skäl, vilka inte berörs här, är det inte helt oproblematiskt att hantera första ordningens logik i en renodlad form, varför man på olika sätt försöker hitta olika alternativ till denna.

Ett sådant är språket *Prolog*, vilket kan hanteras med tekniker baserade på resolutionsmetoden. Ett prologprogram är väsentligen uppbyggt av en mängd *klausuler*:

```
förnuftig(X) :- apa(X), råtta(Y), äter(X,Y).
apa(ulf).
råtta(carl).
äter(ulf,carl).
```

De tre sista klausulerna utgör programmets faktadel och kan läsas som sanna formler. Den första klausulen består av *huvudet* `förnuftig(X)` och *kroppen* `apa(X), råtta(Y), äter(X,Y)`. Denna klausul utgör här *regeldelen* i prologprogrammet och betyder *i princip* detsamma som:

$\forall x \forall y[(\text{apa}(x) \wedge \text{råtta}(y) \wedge \text{äter}(x,y)) \rightarrow \text{förnuftig}(x)]$

Om programmet nu läses som logiska formler inser man att `förnuftig(ulf)` följer från dessa. Hur prövar man detta intuitivt? Om man substituerar *ulf* för `X` och *carl* för `Y` i programmets regeldel erhåller man:

```
förnuftig(ulf) :- apa(ulf), råtta(carl), äter(ulf,carl)
```

Det som står i högerledet ovan finns i programmets faktadel, vilket innebär att `apa(ulf)`, `råtta(carl)` samt `äter(ulf,carl)` är sanna relativt programmet. Eftersom regeln är en implikation och högerledet är en konjunktion av sanna formler där *ulf* har substituerats för `X` och *carl* för `Y`, så är även `förnuftig(ulf)` sant.

Evalueringen av prolog-program fungerar i huvudsak på samma sätt. Negation har emellertid en särställning. Betrakta följande något mer generella program där *x* betecknar en variabel och *a* en konstant:

```
s(X) :- p1(X), p2(X), p3(X).
p1(X) :- q1(X), q2(X).
q1(X) :- r1(X), not(r2(X)).
p2(a).
p3(a).
r1(a).
q2(a).
```

Givet detta program, gäller nu s(a)?

För att s(a) ska kunna gälla måste $p_1(a)$, $p_2(a)$ samt $p_3(a)$ gälla. Hur avgör vi detta? Från den andra klausulen kan vi se att för att $p_1(a)$ ska kunna gälla måste $q_1(a)$ samt $q_2(a)$ gälla. Huruvida $q_1(a)$ gäller kan vi avgöra med den tredje klausulen som innehåller *not*. Detta motsvarar i någon mening logisk negation, men med innebörden att not($r_2(X)$) gäller om $r_2(X)$ inte gäller. Detta innebär att $q_1(a)$ gäller förutsatt att $r_1(a)$ gäller samt att $r_2(a)$ *inte* gäller. I detta fall innebär det sistnämnda att $r_2(a)$ inte förekommer i faktadelen.

$r_1(a)$ förekommer i faktadelen, men $r_2(a)$ gör det ej. Detta innebär att $q_1(a)$ gäller. $q_2(a)$ förekommer i faktadelen. Således gäller $p_1(a)$. Både $p_2(a)$ och $p_3(a)$ förekommer i faktadelen och tillsammans med att $p_1(a)$ gäller resulterar det i att s(a) gäller.

Detta beskriver väsentligen prologs arbetssätt.

# II. Att skapa och analysera en modell

Denna del beskriver hur du bygger en konceptuell modell och hur du därefter utför analyser av denna.

# Skapa en konceptuell modell

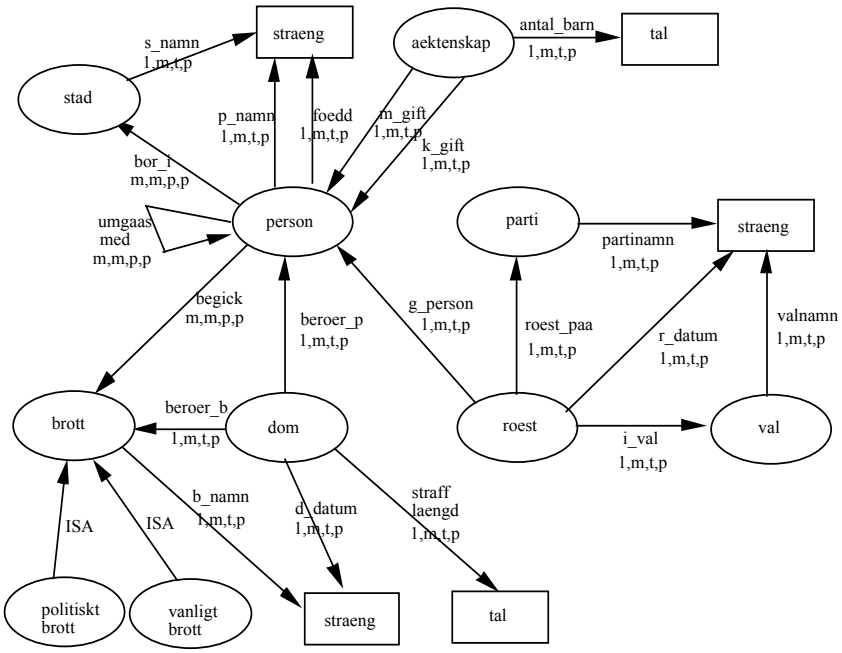
Alfred Hosta är enväldig diktator i landet L, och önskar konstruera ett informationssystem som ger möjlighet att hantera information om befolkningen i L. Du ska nu hjälpa honom med denna motbjudande uppgift.

En viktig sak att tänka på är emellertid att namn på entitetstyper, datatyper samt relationer måste vara unika.

Exempel på utsagor som skall kunna representeras är:

- Personen P bor i staden S, är född den första juni 1987 och är gift med personen Q. Paret har sju barn.
- Personen P röstade på partiet X i valet V datumet D.
- Personen P har begått brottet att välta en staty av Alfred Hosta. Han har dessutom spottat på Hostas hedersvakt. Dessa är politiska brott.
- Personen P har blivit dömd den första juni 1987 till fem års fängelse för brottet B.
- Ett brott är politiskt eller vanligt.
- Personen P umgås med personen Q.

Figur 2–1 visar ett grafiskt schema som kan tänkas beskriva den statiska delen av ett sådant system.




**figur 2–1**

Du ska även komplettera den statiska delen med konsistensregler som uttrycker att ett brott antingen är politiskt eller annat, men inte både politiskt och annat. Det statiska schemat ska även innehålla en härledningsregel som uttrycker att en person är politiskt misstänkt om denne har begått ett politiskt brott eller umgås med en politiskt misstänkt person.

Ytterligare en konsistensregel som ska finnas är att en person inte kan begå ett politiskt brott om personen bor i staden Säffle. Det kan förefalla märkligt, men entydig empirisk forskning har påvisat att detta är fallet.

Det finns även ett användarkrav på detta system: en dom kan inte beröra både ett politiskt brott och ett vanligt brott.

# Definiera relationer

Studera grafen i figur 2–1. Du definierar relationen *beroer_b* som har entitetstypen *dom* som domän och entitetstypen *brott* som värdeförråd med formeln:

$\forall x \forall y[beroer\_b(x,y) \rightarrow (dom(x) \wedge brott(y))]$.

Till denna relation hör avbildningsregeln 1,m,t,p. Denna representeras med formlerna:

$\forall x \forall y \forall z[(beroer\_b(x,y) \wedge beroer\_b(x,z)) \rightarrow y=z]$

$\forall x \exists y[dom(x) \rightarrow (beroer\_b(x,y) \wedge brott(y))]$

# Definiera subtyper

Entitetstypen brott har två subtyper: *politisktbrott* samt *vanligtbrott*. Subtyper deklareras genom att man anger man hur den förhåller sig till sin supertyp, det vill säga att alla instanser av subtypen även är instanser av dess supertyp.

Du definierar således de två subtyperna *politisktbrott* samt *vanligtbrott* med följande två formler:

$\forall x[politisktbrott(x) \rightarrow brott(x)]$.

$\forall x[vanligtbrott(x) \rightarrow brott(x)]$.

# Definiera konsistensregler

För att ange krav på systemet så definierar du olika typer av konsistensregler. Sådana uttrycks här med vanliga formler i predikatlogik. Med konsistensregler kan du ge ytterligare innebörd till det statiska schemat.

Som du såg i början av denna del ska du även komplettera den statiska delen med konsistensregler som uttrycker att ett brott antingen är politiskt eller vanligt, men inte både politiskt och vanligt. Detta gör du t ex med formlerna.

$\forall x[brott(x) \rightarrow (politisktbrott(x) \vee vanligtbrott(x))]$

$\neg \exists x[politisktbrott(x) \wedge vanligtbrott(x)]$ eller $\forall x \forall y[(politisktbrott(x) \wedge vanligtbrott(y)) \rightarrow \neg x=y]$.

I anslutning till detta ska du även skriva en konsistensregel som anger att en person inte kan begå ett politiskt brott om personen bor i staden Säffle. Detta kan man göra med följande formel:

$\neg\exists p\exists s\exists b[person(p) \wedge bor\_i(p,s) \wedge stad(s) \wedge s\_namn(s,saeffle) \wedge politisktbrott(b) \wedge begick(p,b)]$

# Definiera härledningsregler

För schemat i figur 2–1 ska det gälla att en person är politiskt misstänkt om denne har begått ett politiskt brott eller umgås med en politiskt misstänkt person.

$\forall p\forall b[(begick(p,b) \wedge politisktbrott(b)) \rightarrow politiskt\_misstaenkt(p)]$

$\forall p\forall q[(umgaasmed(p,q) \wedge politiskt\_misstaenkt(q) \wedge \neg p=q) \rightarrow politiskt\_misstaenkt(p)]$

Den första formeln representerar att en person är politiskt misstänkt om denne har begått ett politiskt brott. Den andra representerar att en person som umgås med en politiskt misstänkt person också den är en misstänkt person.

# Informationsbas

En informationsbas som representerar ett visst tillstånd för schemat ovan kan exempelvis se ut som den följande.

```
person(p1).
person(p2).
person(p3).
person(p4).
p_namn(p1,carl).
p_namn(p2,per).
p_namn(p3,britt).
p_namn(p4,margareta).
foedd(p1,19481123).
foedd(p2,19331223).
foedd(p3,19700101).
foedd(p4,19221019).
umgaas_med(p1,p4).
umgaas_med(p4,p1).
bor_i(p1,s1).
bor_i(p2,s2).
bor_i(p3,s1).
bor_i(p4,s1).
stad(s1).
stad(s2).
s_namn(s1,stockholm).
s_namn(s2,saeffle).
aektenskap(a1).
m_gift(a1,p1).
k_gift(a1,p4).
antal_barn(a1,0).
g_person(r1,p2).
roest(r1).
r_datum(r1,19951027).
roest_paa(r1,pa1).
parti(pa1).
partinamn(pa1,hostapartiet).
i_val(r1,v1).
val(v1).
valnamn(v1,det_sista_valet).
politisktbrott(b1).
politisktbrott(b2).
b_namn(b1,statyvaeltning).
b_namn(b2,vaktspottning).
```

# Analysera ett konceptuellt schema

Du kan nu genomföra analyser utifrån det konceptuella schemat och kravspecifikationen. I appendix finns det konceptuella schemat i textuell form.

Som du såg finns det även ett användarkrav på detta schemat: En dom kan inte beröra både ett politiskt brott och ett vanligt brott. Detta kan representeras med formeln:

$\neg\exists x\exists y\exists z[dom(x) \wedge beroer\_d(x,y) \wedge beroer\_d(x,z) \wedge politisktbrott(y) \wedge vanligtbrott(z)]$

Att detta krav följer från schemat kan nu visas med resolutionsmetoden.

Negationen av kravet (slutsatsen) ger
$\exists x\exists y\exists z[dom(x) \wedge beroer\_d(x,y) \wedge beroer\_d(x,z) \wedge politisktbrott(y) \wedge vanligtbrott(z)]$

Skolemisering ger
$dom(a) \wedge beroer\_d(a,b) \wedge beroer\_d(a,c) \wedge politisktbrott(b) \wedge vanligtbrott(c)$, dvs mängden $\{dom(a), beroer\_d(a,b), beroer\_d(a,c), politisktbrott(b), vanligtbrott(c)\}$.

I schemat finns formeln $\forall x\forall y\forall z[(beroer\_b(x,y) \wedge beroer\_b(x,z)) \rightarrow y=z]$. Vi kan skriva denna på klausulform som $\neg beroer\_b(x,y) \vee \neg beroer\_b(x,z) \vee y=z$.

$beroer\_d(a,b)$ $\quad \neg beroer\_b(x,y) \vee \neg beroer\_b(x,z) \vee y=z$

$\neg beroer\_b(a,z) \vee b=z$ $\quad beroer\_d(a,c)$

$b=c$

I schemat finns även en klausul som säger att
$\forall x\forall y[(politisktbrott(x) \wedge vanligtbrott(y)) \rightarrow \neg x=y]$ som vi kan skriva som klausulformen:
$\neg politisktbrott(x) \vee \neg vanligtbrott(y) \vee \neg x=y$.

$politisktbrott(b)$ $\quad \neg politisktbrott(x) \vee \neg vanligtbrott(y) \vee \neg x=y$

$\neg vanligtbrott(y) \vee \neg b=y$ $\quad vanligtbrott(c)$

$\neg b=c$ $\quad b=c$

$\perp$

# Appendix 1

## Textuell koncepuell modell

$\forall x \forall y[p\_namn(x,y) \rightarrow (person(x) \wedge straeng(y))]$

$\forall x \forall y \forall z[(p\_namn(x,y) \wedge p\_namn(x,z)) \rightarrow y=z]$

$\forall x \exists y[person(x) \rightarrow (p\_namn(x,y) \wedge straeng(y))]$

$\forall x \forall y[foedd(x,y) \rightarrow (person(x) \wedge straeng(y))]$

$\forall x \forall y \forall z[(foedd(x,y) \wedge foedd(x,z)) \rightarrow y=z]$

$\forall x \exists y[person(x) \rightarrow (foedd(x,y) \wedge straeng(y))]$

$\forall x \forall y[bor\_i(x,y) \rightarrow (person(x) \wedge stad(y))]$

$\forall x \forall y[umgaas\_med(x,y) \rightarrow (person(x) \wedge person(y))]$

$\forall x \forall y[begick(x,y) \rightarrow (person(x) \wedge brott(y))]$

$\forall p \forall b[(begick(p,b) \wedge politisktbrott(b)) \rightarrow politiskt\_misstaenkt(p)]$

$\forall p \forall q[(umgaasmed(p,q) \wedge politiskt\_misstaenkt(q)) \rightarrow politiskt\_misstaenkt(p)]$

$\forall x \forall y[m\_gift(x,y) \rightarrow (aektenskap(x) \wedge person(y))]$

$\forall x \forall y \forall z[(m\_gift(x,y) \wedge m\_gift(x,z)) \rightarrow y=z]$

$\forall x \exists y[aektenskap(x) \rightarrow (m\_gift(x,y) \wedge person(y))]$

$\forall x \forall y[k\_gift(x,y) \rightarrow (aektenskap(x) \wedge person(y))]$

$\forall x \forall y \forall z[(k\_gift(x,y) \wedge k\_gift(x,z)) \rightarrow y=z]$

$\forall x \exists y[aektenskap(x) \rightarrow (k\_gift(x,y) \wedge person(y))]$

$\forall x \forall y[antal\_barn(x,y) \rightarrow (aektenskap(x) \wedge tal(y))]$

$\forall x \forall y \forall z[(antal\_barn(x,y) \wedge antal\_barn(x,z)) \rightarrow y=z]$

$\forall x \exists y[aektenskap(x) \rightarrow (antal\_barn(x,y) \wedge tal(y))]$

$\forall x \forall y[s\_namn(x,y) \rightarrow (stad(x) \wedge straeng(y))]$

$\forall x \forall y \forall z[(s\_namn(x,y) \wedge s\_namn(x,z)) \rightarrow y=z]$

$\forall x \exists y[stad(x) \rightarrow (s\_namn(x,y) \wedge straeng(y))]$

$\forall x \forall y[b\_namn(x,y) \rightarrow (brott(x) \wedge straeng(y))]$

$\forall x \forall y \forall z[(b\_namn(x,y) \wedge b\_namn(x,z)) \rightarrow y=z]$

$\forall x \exists y[brott(x) \rightarrow (b\_namn(x,y) \wedge straeng(y))]$

$\forall x[brott(x) \rightarrow (politisktbrott(x) \vee vanligtbrott(x))]$

$\forall x \forall y[(politisktbrott(x) \wedge vanligtbrott(y)) \rightarrow \neg x=y]$

$\forall x[politisktbrott(x) \rightarrow brott(x)].$

$\forall x[vanligtbrott(x) \rightarrow brott(x)].$

$\neg \exists p \exists s \exists b[person(p) \wedge bor\_i(p,s) \wedge stad(s) \wedge s\_namn(s,saeffle) \wedge politisktbrott(b) \wedge begick(p,b)]$

$\forall x \forall y[d\_datum(x,y) \rightarrow (dom(x) \wedge straeng(y))]$

$\forall x \forall y \forall z[(d\_datum(x,y) \wedge d\_datum(x,z)) \rightarrow y=z]$

$\forall x \exists y[dom(x) \rightarrow (d\_datum(x,y) \wedge straeng(y))]$

$\forall x \forall y[straff\_laengd(x,y) \rightarrow (dom(x) \wedge tal(y))]$

$\forall x \forall y \forall z[(straff\_laengd(x,y) \wedge straff\_laengd(x,z)) \rightarrow y=z]$

$\forall x \exists y[dom(x) \rightarrow (straff\_laengd(x,y) \wedge tal(y))]$

$\forall x \forall y[beroer\_p(x,y) \rightarrow (dom(x) \wedge person(y))]$

$\forall x \forall y \forall z[(beroer\_p(x,y) \wedge beroer\_p(x,z)) \rightarrow y=z]$

$\forall x \exists y[dom(x) \rightarrow (beroer\_p(x,y) \wedge person(y))]$

$\forall x \forall y[beroer\_b(x,y) \rightarrow (dom(x) \wedge brott(y))]$

$\forall x \forall y \forall z[(beroer\_b(x,y) \wedge beroer\_b(x,z)) \rightarrow y=z]$

$\forall x \exists y[dom(x) \rightarrow (beroer\_b(x,y) \wedge brott(y))]$

$\forall x \forall y[r\_datum(x,y) \rightarrow (roest(x) \wedge straeng(y))]$

$\forall x \forall y \forall z[(r\_datum(x,y) \wedge r\_datum(x,z)) \rightarrow y=z]$

$\forall x \exists y[roest(x) \rightarrow (r\_datum(x,y) \wedge straeng(y))]$

$\forall x \forall y[roest\_paa(x,y) \rightarrow (roest(x) \wedge parti(y))]$

$\forall x \forall y \forall z[(roest\_paa(x,y) \wedge roest\_paa(x,z)) \rightarrow y=z]$

$\forall x \exists y[roest(x) \rightarrow (roest\_paa(x,y) \wedge parti(y))]$

$\forall x \forall y[i\_val(x,y) \rightarrow (roest(x) \wedge val(y))]$

$\forall x \forall y \forall z[(i\_val(x,y) \wedge i\_val(x,z)) \rightarrow y=z]$

$\forall x \exists y[roest(x) \rightarrow (i\_val(x,y) \wedge val(y))]$

$\forall x \forall y[partinamn(x,y) \rightarrow (parti(x) \wedge straeng(y))]$

$\forall x \forall y \forall z[(partinamn(x,y) \wedge partinamn(x,z)) \rightarrow y=z]$

$\forall x \exists y[parti(x) \rightarrow (partinamn(x,y) \wedge straeng(y))]$

$\forall x \forall y[valnamn(x,y) \rightarrow (val(x) \wedge straeng(y))]$

$\forall x \forall y \forall z[(valnamn(x,y) \wedge valnamn(x,z)) \rightarrow y=z]$

$\forall x \exists y[val(x) \rightarrow (valnamn(x,y) \wedge straeng(y))]$