SISU rapport

nr 6

Konceptuell modellering med naturligt språk

Harriet Dahlgren

SISU
Svenska Institutet för Systemutveckling
Box 1250, 164 28 KISTA

Konceptuell modellering med naturligt språk

ISSN: 0282-9924

Innehåll

1. Introduktion

Denna rapport beskriver ett system för ett naturligtspråkgränssnitt till konceptuell modellering, utarbetat vid SISU (Svenska Institutet för Systemutveckling) sommaren 1989.

Jag vill här passa på att framföra ett stort tack till min handledare Jan Ljungberg som bistått med råd och hjälp.

Rapportens inledande avsnitt ger en kortfattad beskrivning av modellering och gränssnitt i naturligt språk. Dessutom redogörs för systemets omfattning.

Det andra avsnittet består av en systemöversikt, för en övergripande förståelse av systemet.

Systemet beskrivs i detalj i det tredje avsnittet. De ingående delarna granskas djupare tillsammans med en studie av modelleringsansatsen (EMOL) och andra använda verktyg.

Representationen av konceptuella modeller som kan genereras med systemet analyseras. Detta i avsnitt fyra, främst med avseende på EMOL som modelleringsansats och modellernas förutbestämda Prolog-representation.

Sammanfattning och slutsatser avslutar rapporten.

Exempel på modellkonstruktion från naturligt språk till färdig modell med systemet ges i Appendix.

1.1 Konceptuell modellering

Inför informationssystemutveckling behövs en detaljerad analys av den verksamhet som systemet skall stödja, bland annat för att kartlägga informationsbehovet vid verksamhetsfunktionerna. Resultatet av analysen representeras med fördel i form av en Konceptuell Modell (KM). Modellen kan utgöra grund för databas- och programutformning, nya informationssystem eller vidareutveckling av informationssystem i användning.

Vid konceptuell modellering abstraherar man i önskad grad på olika punkter inom objektssystemet (verksamheten eller systemet som skall modelleras). Generellt i modelleringssammanhang betraktas en verksamhet som objekt och samband mellan dessa. Vilken typ av system som

skall modelleras och vad i systemet som är av intresse avgör vilka begrepp, objekt och samband som skapas i modellen [Bubenko 84].

Konceptuell modellering innefattar att ge begrepp och objekt som används inom verksamheten en klar definition. De beskrivs med datatermer (dataelement) som också skall definieras. Definitioner av begrepp är ett kännetecken för konceptuella modeller som utnyttjas vid målanalys och begreppsutveckling i samband med affärsutveckling. Den klara strukturen av en konceptuell modell, är ett annat kännetecken som är avgörande vid tillämpningar som organisationsutveckling, administrativ utveckling och för verksamhets- och funktionsanalys i samband med det.

En konceptuell modell kan delas upp i ett konceptuellt schema och en konceptuell informationsbas. Ett konceptuellt schema talar om vilka typer av uppgifter om verksamheten som är tillåtna (i en syntaxdel) och vilka regler som gäller för uppgifterna (i en regeldel). Regler kan till exempel specificera hur vissa uppgifter kan härledas från andra och vilka förändringar av uppgifter som är tillåtna. En konceptuell informationsbas (faktadel) innehåller en mängd enskilda uppgifter av de typer som definierats i schemat. Uppgifterna som ingår i informationsbasen är utsagor om företeelser och samband mellan företeelser i en verksamhet.

1.2 Naturligtspråkgränssnitt

Gränssnitt i naturligt språk mot olika typer av system har många fördelar. Den kanske största fördelen med ett naturligtspråkgränssnitt är att användarna slipper lära sig ett speciellt språk för systeminteraktionen.

För ovana datoranvändare går mycket energi åt till att memorera kommandon i datorspråk, komma ihåg vad de står för och när de skall användas. Med sitt modersmål som kommunikationsspråk till systemet slipper nya och sporadiska användare lägga ner tid på inlärning av ett språk och kanske dessutom fortare kommer över eventuella osäkerhetskänslor inför systemanvändning. Med naturligt språk erbjuds också alternativa uttryck för samma sak.

Som nackdel med naturligtspråkgränssnitt kan anföras att det innebär långa "kommandon", och med mycket att skriva ökar risken för felstavning. Liksom dataspråkgränssnitt är gränssnitt i naturligt språk ofta hårt styrda av den syntax (form och ordföljd) som kan hanteras och därmed känsliga för t ex stavfel.

Att använda obegränsat naturligt språk, som människor gör sinsemellan, även i kommunikation med datorer är en utopi. I stället används delmängder av naturligt språk, s k begränsat NL (Natural Language) [Amble 87a] En applikation berör alltid en begränsad domän (verksamhetsområde), varför det blir möjligt att använda naturligt språk till gränssnitt.

1.2.1 Analys av naturligt språk

Beroende på applikationen för ett naturligtspråkgränssnitt är genereringen av språk till en dialog mer eller mindre viktig. Däremot krävs en utförlig analys av de inskrivna språkliga satserna. Detta för att ur de oprecisa uttrycken extrahera något tillräckligt formellt för utvärdering inom applikationen. Analysen består av flera faser som oftast utförs modulärt i nedanstående ordning.

Lexikalanalys
Kontrollerar att orden i indatasträngen finns upptagna i systemlexikonet för gränssnittet. Ju större lexikon desto mer uttrycksmöjligheter men det kan också innebära lagringsproblem. En rättstavningskontroll kan vara involverad här [Amble 82].

Syntaxanalys
- Relaterar orden i en sats till sina respektive ordklasser.
- Utför satslösning, dvs bestämmer satsdelsfunktionen för ord och fraser.
- Kontrollerar ordföljden i en sats.
- Ser till att böjningskongruens uppfylls.
  Tex: Artikel, adjektiv och substantiv måste överensstämma i: *Ett gott äpple. Den god äpple* är otillåten eftersom ordklasserna har olika numerus, genus och species ("bestämdhet").
- Ger oftast ifrån sig en analysstruktur i form av ett "syntaxträd" som visar satsens syntaktiska information enligt ovan.

Semantisk analys
Skall analysera betydelsen av en sats. Utgår ofta från ett syntaxträd och en kunskapsbas med allmän och domänspecifik kunskap. Den semantiska analysen kan ge ett uttryck i predikatlogik som resultat. Detta är en tillräckligt formell representation att utgå ifrån, i ett gränssnitt där funktioner och information i systemapplikationen är målet för analysen.

Automatisk analys av naturligt språk kännetecknas av svårigheter med språkets tvetydigheter på alla nivåer i analysen [Winograd 84]. Många av dessa tvetydigheter utgör aldrig något problem för människor, tack vare sammanhang och "kunskap om världen". Ibland följs den semantiska analysen av en pragmatisk analys där denna typ av information är avgörande.

1.3 Varför naturligtspråkgränssnitt till modellering?

En felaktig konceptuell modell kan ge dyra följder och förödande resultat vid systemutveckling och databasdesign. Många svårigheter föreligger vid konceptuell modellering och några orsaker till felaktiga modeller [Kersten 86] ges av att:

- Modellerarna ofta har otillräcklig domänkunskap. Bristen skall avhjälpas med intervjuer av de som ingår i verksamheten. Detta är dock en komplicerad apparat, som kan tyckas dyr i både tid och pengar. Även om intervjuer genomförs på ett exemplariskt sätt kan information vara svårfångad.

- Applikationen (objektssystemet) är ytterst komplicerad. Även för de mest erfarna modellerarna kan informationsstrukturering vara ett problem vid stora mängder av objekt, relationer och aspekter.

- Användare, som har domänkunskap, har inga kunskaper i modellering.

1.3.1 Existerande system NL-KM

Kersten et al [Kersten 86] vid Centre for Mathematics and Computer Science, Amsterdam, Holland, har utvecklat ett expertsystem med NL-gränssnitt för databasmodellering. ACME-systemet skall från engelsk text interaktivt utveckla en KM. Denna process består av textanalys, modellgenerering och modellförbättring.

Den språkliga analysen (parsning) resulterar i en mängd predikationer (enligt ramar med subjekt, objekt och modifierare) där hänsyn har tagits till satsers beroendeförhållanden. Predikationerna tolkas och kombineras, med hjälp av fördefinierad domänkunskap, till ett preliminärt XEAR-schema (eXtended Entity Attribute Relationship), en utökad EAR-modell beskrivning.

Förutom grundbegreppen

entitet	- objekt, företeelser i objektsystemet som man vill ha information om
attribut	- egenskaper för en entitet
relation	- förhållanden mellan objekt

används i XEAR-modellen bl a hierarkier för typer av entiteter (objektstyper) och händelser som beskriver förändringar i objektssystemet (med bl a villkor och följder).

En mängd heuristiska regler för kontroll av strukturell inkonsistens, tvetydighet och fullständighet appliceras på den konceptuella modellen i modellförbättringen, där brister i modellen rapporteras. Utgående från denna diagnos kan användaren iterativt förändra och förbättra med systemets hjälp. Resultatet som ACME ger ifrån sig är en lista av entiteter, namngivna förhållanden och händelser (med villkor och följder).

OICSI är ett utvecklat hjälpmedel för design av konceptuella datamodeller, presenterat av Cauvet et al [Cauvet 88] vid Paris1 och Paris6 universitetet. Här används franska som beskrivningsspråk av applikationen (objektsystemet). Utifrån kunskapen hos modelleringsexperter har ett antal heuristiska regler skapats som med analysen av den språkliga beskrivningen sedan genererar ett konceptuellt schema.

1.3.2 En utvecklingsmiljö för konceptuell modellering

Expertsystem för modellering ger en möjlighet för de inblandade vid en verksamhet att själva skissera en konceptuell modell. De har den bästa kunskapen om objektssystemet. Ett naturligtspråkgränssnitt kan göra det lätt att beskriva verksamheter i ett expertsystem som inte kräver utförlig kunskap om modellering [Black 87] .

Dialoger i gränssnittet kan leda till att all nödvändig information för kvalitativa KM ges av användarna. Gränssnittet kan även innehålla en komponent, för genering av språkliga formuleringar om den uppbyggda modellens utseende, för validering.

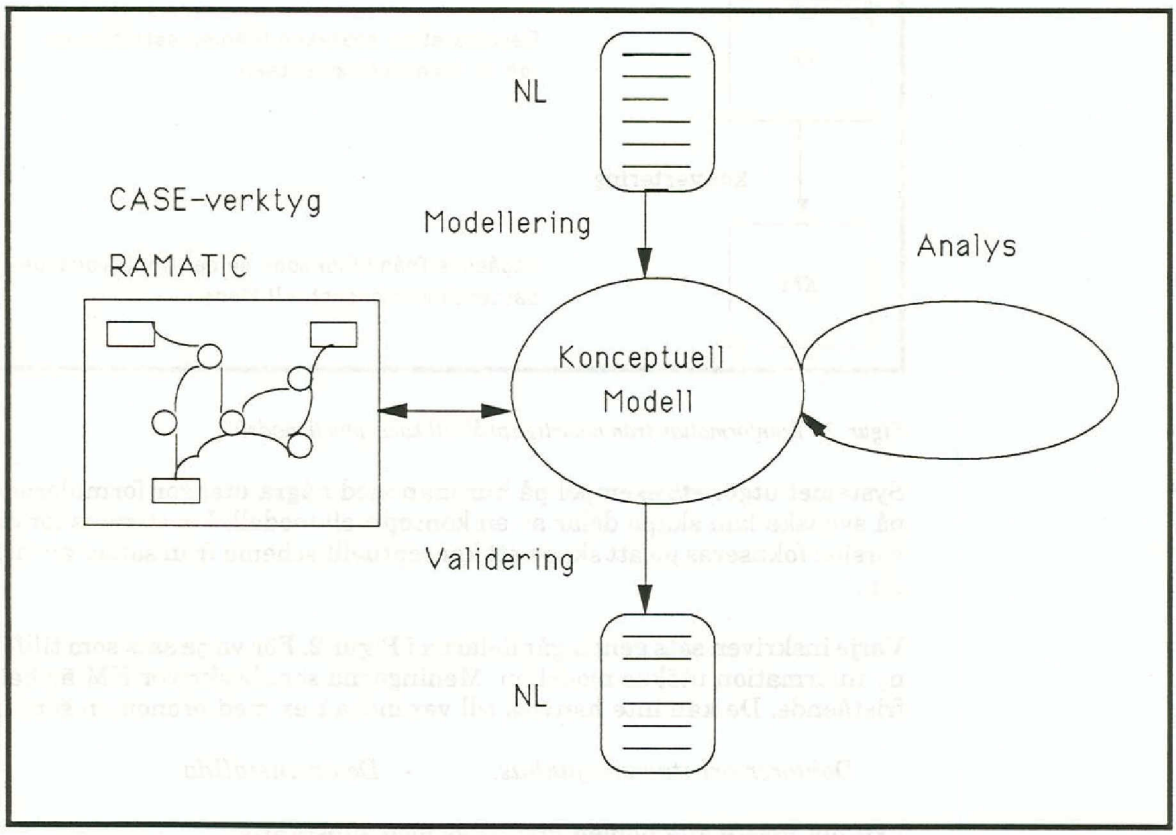
Det finns många datorstöd i form av sk CASE-verktyg (Computer Aided System Engineering) för systemutveckling. RAMATIC är ett CASE-verktyg [Dahl 1985], utvecklat vid SISU, med vars hjälp konceptuella modeller kan skapas på grafisk väg. Modellerna lagras internt i RAMATIC som CS5-databaser.

Ingående i ett kontrollsystem för kvalitet i KM finns ett program skrivet i C som läser om modeller från databasen till instanser av några fördefinierade Prologpredikat. På dessa instanser tillämpas ett antal kvalitetsbestämda regler, som kontrollerar syntax, semantik, konsistens m m [Wangler & Wohed 88].

Med alla komponenter ovan kan en produktiv utvecklingsmiljö med användarvänliga gränssnitt och stöd för konceptuella modeller skapas (Fig 1).

Under tiden som en applikationsbeskrivning ges på svenska, växer den grafiska representationen av en KM fram i RAMATIC (med fönsterhantering ges en simultan lättöverskådlig presentation av de båda). Denna "feed-back" i strukturskapande form kan visa om resultatet blev det önskade. Det är även ett sätt att lära sig en ny grafisk notation för en modelleringsansats.

Kontrollsystemet kan användas på den skapade modellen för en analys av möjliga förbättringar. Till sist kan modellen återges i naturligt språk för en slutlig validering om modellen motsvarar ambitionerna.

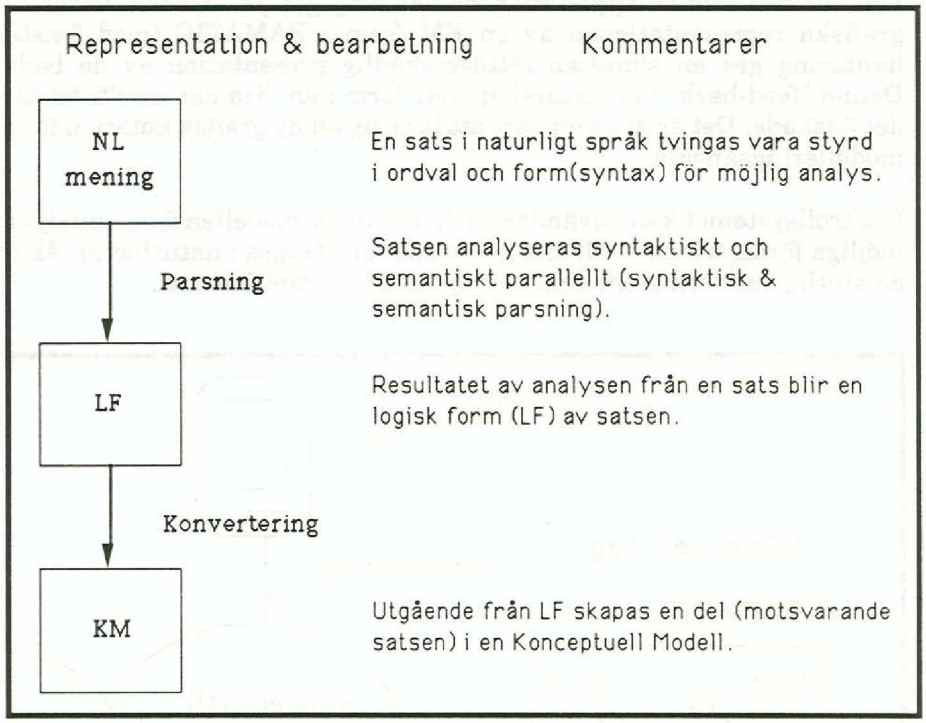


Figur 1. En utvecklingsmiljö för konceptuell modellering.

1.4 Systemets omfattning

Systemet som den här rapporten beskriver utgör modelleringsdelen från naturligt språk (NL) till konceptuell modell (KM) i utvecklingsmiljön (Figur 1).

Övergången från satser i naturligt språk till konceptuell modell innehåller flera delar (Fig 2). Satserna bearbetas till en mellanform (logisk form). Från den representationen skapas sedan den konceptuella modellen.



Figur 2. Tranformation från naturligtspråk till konceptuell modell.

Systemet utgör ett exempel på hur man med några utsagor formulerade på svenska kan skapa delar av en konceptuell modell. I systemets första version fokuseras på att skapa ett konceptuellt schema från satser av viss typ.

Varje inskriven sats genomgår delarna i Figur 2. För varje sats som tillför ny information utökas modellen. Meningarna som beskriver KM är helt fristående. De kan inte hänvisa till varandra t ex med pronomen som i:

Doktorer arbetar på sjukhus. - *De är anställda.*

I stället måste alla nomen uttryckas med substantiv:

- *Doktorer är anställda.*

Domänen inom vilken modellen skapas är bestämd till en sjukhusverksamhet.

Av utsagor om typer av företeelser (begrepp och objektstyper) vid ett sjukhus och samband dem emellan kan en modell motsvarande syntax- och regeldelen i KM konstrueras för sjukhusverksamheten. I fortsättningen refereras till en sådan modell som Konceptuell Modell (KM). Domänbestämningen inför naturligtvis begränsningar på ordvalet i utsagorna som beskriver modellen.

Uttryckskraften hos modelleringsansatsen (modelleringsspråket) avgör också vilka KM som kan skapas. I systemet används EMOL (Ericsson MOdelling Language) som modelleringsspråk [Ericsson Telecom 88]. Det beskrivs utförligt i avsnitt tre. Sammanfattningsvis är EMOL ett mindre modelleringsspråk som grundar sig på SIMOL (utvecklat inom SISU) [Lindencrona 86].

EMOL är avsett för datamodellering och har tre grundbegrepp:

OBJEKT - något man vill hålla information om. T ex: 'Avdelning'

ATTRIBUT - beskriver förhållande mellan ett objekt och en domän/ ett objekt
T ex: avdelningsnummer (ett attribut för avdelning till en domän 'NUMMER') behandlar (ett attribut mellan 'Doktor' & 'Patient')

DOMÄN - värdeförråd (mängd av möjliga värden) som reglerar tillåtna attributvärden för domänvärda attribut
T ex: 'DATUM' på formen ÅÅMMDD
där MM<= 12 DD<= 31

EMOL innehåller inte händelser (event).

"Mapping" (avbildningsegenskaper) inom ett modelleringsspråk beskriver restriktioner för attribut. Till exempel för attributet "äga" kan antalet ägda föremål (objekt) vara bestämda för en ägare som i:

En bilägare äger minst en bil.

I EMOL skulle mapping för "äga" här skrivas som (1:m) = en eller många på formen (MINSTA ANTAL: FLEST ANTAL) förekomster ur "bilmängden" som en bilägare kan äga.

Avbildning (mapping) är den enda formen av regler som kan uttryckas inom systemet. Vissa nyckelord i utsagorna som kan uttryckas inom systemet betyder speciell mapping (se avsnitt 3.2.1.1).

Det finns inget relevant material som beskriver informella satser som kan komma i fråga vid modellering. Detta kan man konstatera med insikten om att systemet måste ha ett relativt begränsat utseende. Grammatiken som beskriver satser som kan användas kan till exempel inte spegla "obegränsat NL" [Amble 87a]. I stället får ett antal stipulerade satser utgöra grunden för grammatikbeskrivningen (se avsnitt 3.2.1).

För att få fram relevanta och troliga satser användes följande utgångspunkter:

- Satsernas innehåll måste vara knutet till objektsystemet, dvs den verksamhet som skall modelleras; Här en sjukhusmiljö, vilket framförallt bestämmer lexikonets innehåll.

- Målet med utsagorna är att konstruera en konceptuell modell. Endast satser som beskriver objektsystemet på ett relevant sätt för modellering hör hit.

- Satserna i systemets första version skall inte användas till en dialog, dvs man skall bara använda deklarativa satser som motsvarar utsagor om objektsystemet.

- Med ovanstående krävs förutom kunskap om verksamheten också att användarna behärskar vilken information man vill extrahera vid modellering.

- Modelleringsansatsen (modelleringsspråket) för modellen är avgörande. Även denna måste man ha kunskap om. Vilka regler gäller? Vad måste definieras för att modellen skall bli komplett? Vad kan inte beskrivas med den här ansatsen?

I försöket att formulera satser för modellering i systemet, läggs tonvikten vid aktuella utsagor som återfinns under schemadelen av en modell, dvs övervägande positiva utsagor om förhållanden.

Vid konstruktionen av en konceptuell modell vill man generalisera över objekt och förhållanden. Utsagorna som är aktuella skall därför redogöra för typer av objekt och förhållanden. I fortsättningen refererar objekt (entitet) och förhållande (relation) i rapporten till objekt- och förhållandetyper (på schemanivå). Förekomster med samma egenskaper samlas alltså under ett objekt eller ett förhållande.

För specifika regler i schemat är de språkliga formuleringarna annorlunda. Till exempel innehåller de mycket förbud och definitioner av kvantifierad typ:

Företag får inte ha leveransavtal avseende varor av en viss typ med företag i krigförande länder.

En anställds nettolön är bruttolön minus skatt.

Regler har tillsvidare lämnats därhän liksom utsagor som beskriver uppgifter i faktadelen.

Till grund för att bestämma typformuleringar av plausibla satser på svenska vid konceptuell modellering av en sjukhusverksamhet, fanns en relationdatabas med tabellerna:

Sjukhus, Avdelning, Personal, Doktor, Patienter, Diagnos, Labb, Tester, Sjukhuslab, BehDoktor, Upptagna.

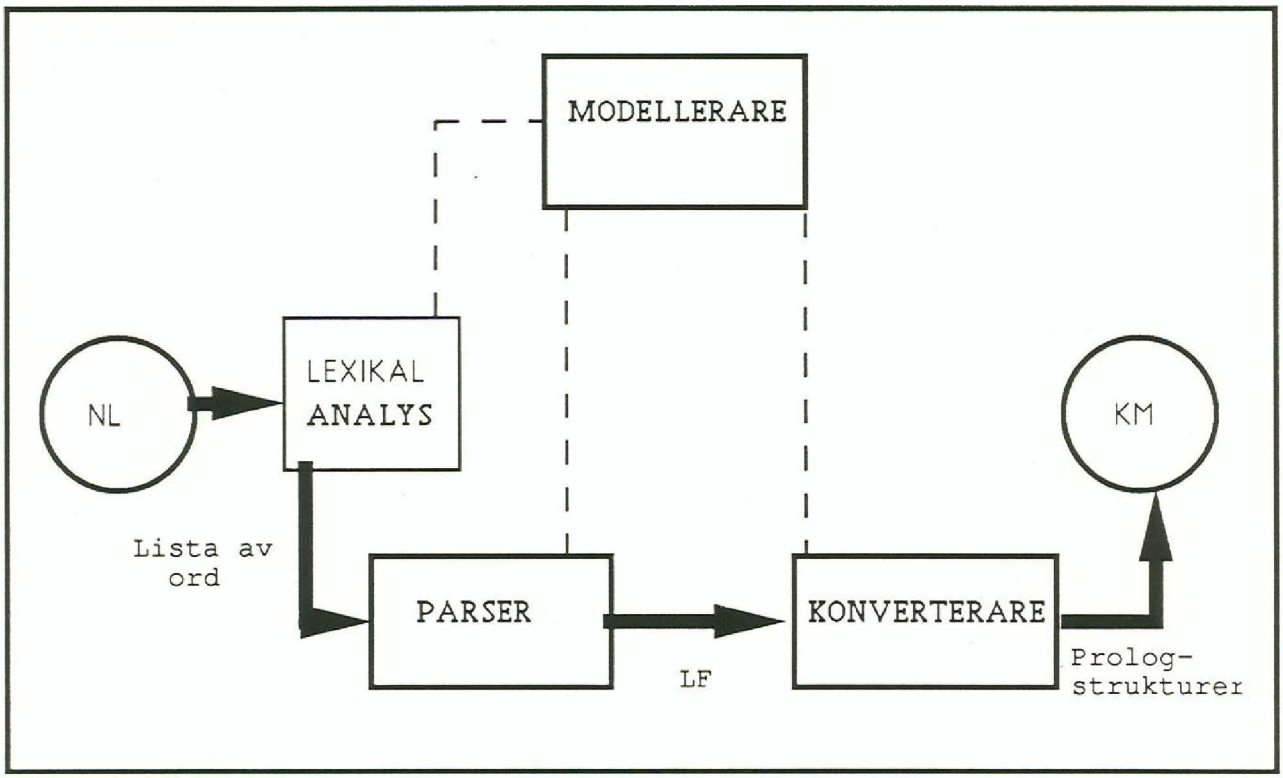
Med utgångspunkt ifrån denna har en enklare konceptuell modell redan skapats [Ljungberg 89].

Dessutom fanns ett lexikon för sjukhusdomänen ("svenska_hosp") att vägleda ordförrådet. Lexikonet har använts till ett naturligtspråkgränssnitt för databasutfrågning i HSQL-projektet [Amble 87b]. Det bygger på ett lexikon till "PRAT 88", en norskt NL-gränssnitt [Teigen 88] inom projektet.

"PRAT 88" är omvandlad från ett engelskt naturligtspråkgränsnitt till en geografidatabas "Chat-80" [Warren 82] utvecklat vid Edinburghs universitet av D.H.D. Warren och F.C.N. Pereira.

Lexikonet i detta system bygger till stor del på det ursprungliga svenska lexikonet inom HSQL-projektet.

2. Systemöversikt



Figur 3. Översikt av systemet som skapar konceptuella modeller (KM) från naturligt språk (NL).

Systemet består av tre huvuddelar: Modellerare, Parser och Konverterare (Fig 3). Modelleraren är gränssnittet mot användarna och samordnar de andra modulerna. Vid konstruktionen av modellen läses en mening i taget in för att först genomgå lexikalanalys. Resultatet av denna analys är en lista av ord. Denna lista skall analyseras syntaktiskt och semantiskt vilket tillkommer Parsern. Parsern ger ifrån sig en Logisk Form (LF) av satsen, ur vilken Konverteraren skapar globala Prologstrukturer. Dessa strukturer av modellpredikat bildar slutligen den konceptuella modellen.

SICStus Prolog [Carlsson 88] har använts genomgående i systemet.

2.1 Modellerare

Systemet är av interaktiv typ och är beroende av användaren för att lösa valsituationer. Användaren ges under interaktionen möjlighet att skriva in flera utsagor för konstruktion av en sammanhängande modell och blir efter varje mening underrättad om modellens utseende. För varje mening utökas modellen med den nya informationen.

Systemet inrymmer ingen åtgärd för att hålla KM konsistent. Ett speciellt kontrollsystem för utvärdering av kvalitet hos en konceptuell modell (representerad som i det här systemet), som bl a innehåller krav på konsistens [Wangler 88] skall så småningom kopplas till systemet.

Inom Modellerarens ram kan man, förutom att skapa modeller, även spara, hämta, komplettera eller ta bort modeller. Dessutom sker specificering av identifierande attribut för objekt i modellen genom en dialog inom Modelleraren.

2.2 Parser

Av grammatikregler skrivna enligt DCG-formalism (Definite Clause Grammar [Pereira 80]) och tillhörande lexikon genereras Prolog-klausuler som sedan fungerar som parser.

I en och samma parser analyseras parallellt syntax och semantik för en sats. Parsern analyserar en lista av ord från en sats och genererar, vid lyckad analys, en logisk form (LF) för satsen.

Analysgrammatiken bygger på en DCG i ett naturligtspråkgränssnitt "Talk" mot en databas [Pereira 87]. I grammatiken beskrivs en mängd svenska deklarativa satser som kan användas för formuleringen av en typorienterad konceptuell modell för en sjukhusverksamhet.

Grammatiken gör inte anspråk på att hantera formuleringar för regler om modellen, ej heller utsagor om förhållanden på förekomstnivå. Frågekonstruktioner för kontroll av modellen ligger tillsvidare också utanför grammatikbeskrivningen.

Grammatiken är tvetydig, dvs den ger flera analyser för en satskonstruktion. Satser med relativbisats och avslutande adverbialfras ger upphov till tvetydigheter.

T ex: *Doktorer behandlar patienter som ligger på sjukhuset.*
Här kan "på sjukhuset" bestämma "Doktorer behandlar patienter" såväl som "Patienter som ligger". Omvänt gäller att samma logiska form kan fås ur olika satskonstruktioner.

LF som genereras av grammatiken skall sedan användas för konstruktion av KM. Den logiska formen måste därför kunna inrymma all information som behövs för modelldesign. Modelleringsansatsen (modelleringsspråket) som används för innehållet i KM är avgörande för vilken information som skall extraheras i analysen. Här används EMOL - Ericsson MOdelling Language [Ericsson Telecom 88], en mindre ansats som grundar sig på SIMOL - SISU MOdelling Language [Lindencrona 86]. För en närmare beskrivning av EMOL se 3.1.2.

Den logiska formen för en huvudsats genererar parsern ur LF för satsdelarna i enlighet med sk kompositionell semantik. Montagues kompositionella semantik [Dowty 81], som använder sig av en högre ordningens logik med typad Lambda-kalkyl, utgör grundidén för grammatikens semantiska beskrivning.

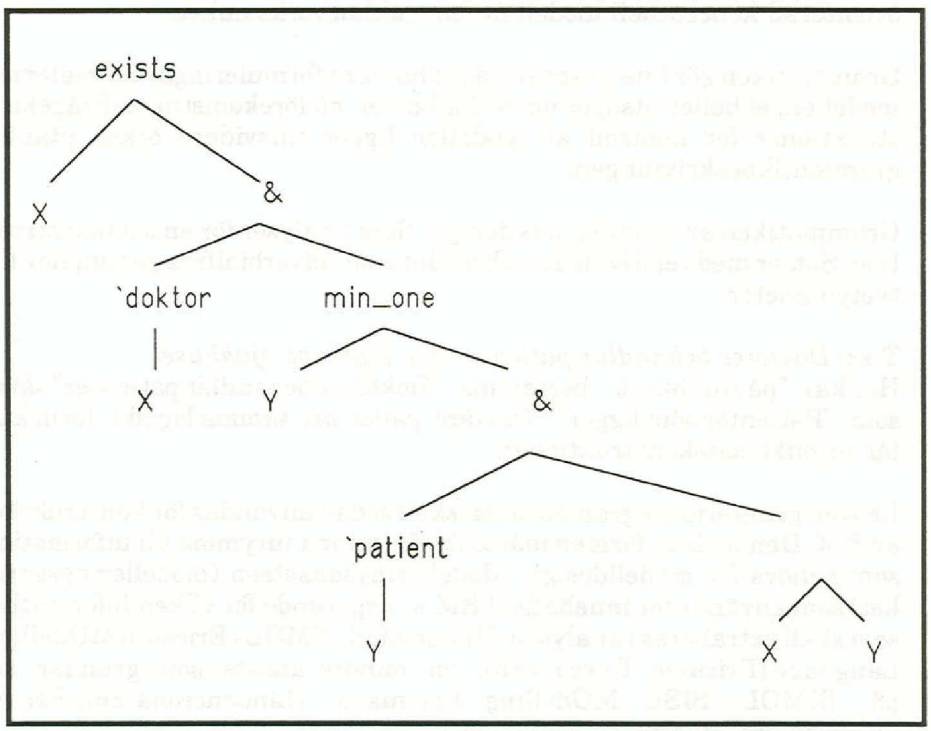
LF i grammatiken motsvarar en Första Ordningens Logik (FOL) utökad med otypad Lambda-kalkyl och speciella kvantifikatorer för mapping (avbildning). FOL:s existens- och all-kvantifikator räcker inte till för att exakt uttrycka EMOL:s mapping.

Mapping i EMOL anger minsta och flest antal tillåtna förekomster av attributvärdet för ett attribut till ett objekt. För att kunna extrahera mapping på formen (MIN,MAX) för attribut och deras inverser används speciella kvantifikatorer, som kan uttrycka alla möjliga kombinationer av 0,1 och M (många).

Kvantifikatorerna utgör LF för vissa artiklar, som används som nyckelord för mappingbeskrivning, i utsagorna om modellen (se 3.2.1.1). Andra artiklar motsvaras av "all" och "exists" (Prologs funktionssymboler för FOL-kvantifikatorer), och ur dem extraheras ej mapping.

2.3 Konverterare

Trots att parsningen av en sats kan ge upphov till olika analyser används bara den första analysen som indata till Konverteraren. Konverteringen utgår ifrån LF i form av en semantisk struktur (träd) för en sats (Fig 4). Den semantiska strukturen traverseras djupet-först, vänster till höger. För konverterarens predikat finns argument motsvarande särdrag som får värden nerifrån trädet (sk syntetiserade) eller uppifrån trädet (ärvda).



Figur 4. Den logiska formen, som är resultatet från parsern, kan ses som en semantisk trädstruktur. Strukturen skall konverteras till en Prolog-representation som utgör del i en konceptuell modell.

Den semantiska strukturen skall konverteras till instanser av Prolog-predikat av den typ som kontrollsystemet appliceras på [Wohed 87]. Prolog-predikaten representerar grundbegrepp i KM.

För EMOL:s grundbegrepp (3.1.2) används predikaten data_object (DOMÄN), entity (OBJEKT), attribute (ATTRIBUT).

Prolog-predikaten som har använts för att skapa KM enligt EMOL har följande utseende:

```
data_object     (NAMN,TYP).

entity          (NAMN,INIT_HÄNDELSE,
                 AVSLUTANDE_HÄNDELSE,
                 ÖVERORDNADE_OBJEKT).

attribute       (NAMN,INVERS_NAMN,
                 ATTR_URSPRUNG,ATTR_DESTINATION,
                 mapping(MIN,MAX,INVERS_MIN,INVERS_MAX),
                 INIT_HÄNDELSE,ÄNDRANDE_HÄNDELSE).
```

EMOL innehåller inga händelser (event), så de representeras som tomma listor.

Argumenten ovan (som används) motsvaras av:

NAMN	- på domän, objekt eller attribut.
TYP	- fördefinierad primitiv typ som preciserar värdeförrådet (integer, float, char, string, date).
ÖVERORDNADE_OBJEKT	- en lista med objekt av vilka objektet är en "direkt" delmängd.
INVERSNAMN	- namn på inversattributet (förhållandet i motsatt riktning)
ATTR_URSPRUNG	- objektet från vilket attributet gäller.
ATTR_DESTINATION	- attributvärdet: objektet/ domänen som attributet gäller till.
MIN	- minsta antal förekomster av attributvärdet för ett ATTR_URSPRUNG
MAX	- högsta antal förekomster av attributvärdet för ett ATTR_URSPRUNG
INVERS_MIN	- som MIN men för inversattributet.
INVERS_MAX	- som MAX men för inversattributet.

Generellt gäller följande förhållande mellan LF och KM:

LF	KM-representation
Egenskap t ex 'doktor(X) (logisk form för substantiv) I bland vid 'har'-relation	OBJEKTNAMN ATTRIBUTNAMN
2-ställigt predikat (relation) 1-ställigt predikat med preposition från adverbialfras	ATTRIBUTNAMN ATTRIBUTNAMN
'isa'-relation	'isa'-relation (i form av en lista med överordnade element till ett objekt)
'har'-relation typ: har(sjukhus,avdelning) typ: har(sjukhus,sjukhusnummer)	 ATTRIBUTNAMN ('har') argument 2 bildar ATTRIBUTNAMN ('sjukhusnummer') till ett domänvärt attribut
kvantifikator "mappingkvantifikator" "annan kvantifikator"	 mapping obestämd mapping (mapping(_,_,_,_))

Strukturen för en sats traverseras och informationen av intresse enligt tabellen ovan extraheras genom Konverterarens olika predikat för delstrukturer.

Information som används till strukturingen av modeller som inte finns ovan är:

LF-VARIABLER - Alla predikat i LF har variabler som argument. T ex X och Y i *exists (X,'doktor(X) & (exists (Y,'patient(Y) & 'behandlar(X,Y)))* Inga egennamn förekommer som t ex i *doktor('Karlsson')*.

Egenskaper som skall bli objekt (LF för substantiv) är 1-ställiga predikat. Argumenten för dessa sparas med egenskaperna under traversering. På så sätt kan det bestämmas i vilken riktning ett förhållande (attribut) råder, dvs vilket objekt (egenskap) som är ATTR_URSPRUNG respektive ATTR_DESTINATION. I exemplet ovan gäller attributet 'behandlar' : *från 'doktor'* - (på X syns att det är ATTR_URSPRUNG som 1:a argument i relationen) *till 'patient'* - som blir ATTR_DESTINATION (eftersom Y är relationens andra argument).

INVERSNAMN - För varje attributnamn finns ett inversnamn lagrat i en liten kunskapsbas.

I slutet av genomlöpningen av strukturen återfinns predikatet (2-ställigt eller 1-ställigt). Då skapas globala Prolog-strukturer (instanser av predikaten ovan) utav informationen som extraherats under traverseringen.

Ur tabellen framgår att det finns flera alternativa modellrepresentationer av LF vid 'har'-relation. Om "det som något/någon har" (2:a argumentet) är ett objekt eller ett attribut till en domän får i detta interaktiva system avgöras genom att fråga användaren. Bara genom denna dialog kan en DOMÄN extraheras.

3. Systembeskrivning

3.1 Verktyg

3.1.1 Prolog

Systemet använder sig av SICStus Prolog [Carlsson 88] utvecklat vid SICS (Swedish Institute of Computer Science). SICStus Prolog är kompatibel med Quintus Prolog och DECsystem-10 Prolog och följer huvudströmmen i Prolog-tradition för syntax och fördefinierade predikat. Den har en interpretator för Definita Klausul Grammatiker (DCG) som har använts för systemets grammatik.

3.1.2 EMOL - Modelleringsansats

Modelleringansatsen EMOL (Ericsson MOdelling Language), som kan användas under RAMATIC, skall bestämma syntaxen och semantiken för representationen av den resulterande konceptuella modellen.

I avsnitt 2.3 presenterades Prolog-predikat som utgör representationer av grundbegrepp i en konceptuell modell. Främst är Prolog-predikaten till för kvalitetskontroll av konstruerade modeller. I det här systemet används predikaten till genereringen av en konceptuell modell enligt EMOL.

Predikaten passar SIMOL [Lindencrona 86] och inkluderar dessutom ett predikat för händelser (event) som inte alls används här.
EMOL beskriver nämligen inga händelsebegrepp.

EMOL [Ericsson Telecom 88] är avsett för datamodellering och har tre grundmodelleringsbegrepp:

OBJEKT - är något man vill hålla information om.

ATTRIBUT - beskriver förhållande mellan ett objekt och en domän/ett objekt.

DOMÄN - är värdeförråd som reglerar tillåtna värden (för domänvärda attribut).

Grundbegreppen motsvaras av följande Prolog-predikat:

OBJEKT -- `entity`
Objektet 'Patient' kan t ex representeras i Prolog-format som:
`entity('Patient', [] , [] , ['Person'`
där 'Patient' är objektnamnet, de tomma listorna står för skapande resp borttagande händelse.
'Person' är en överordnad mängd till 'Patient'.

ATTRIBUT -- `attribute`
Attributet 'har' representeras med "attribut"-predikatet som:
`attribute(har,tillhör,'Sjukhus','Avdelning',`
`mapping(1,m,1,1),[],[]).`
där 'har' är attributnamn,'tillhör' är namnet på invers-attributet, dvs *En avdelning "tillhör" ett sjukhus*
'Sjukhus' är objektet "för vilket attributet gäller"
Sjukhus' har 'Avdelningar'
'Avdelning' är attributvärdet - ett objekt
mapping(1,m,1,1) säger att
Ett sjukhus har en eller flera avdelningar"&
En avdelning tillhör exakt ett sjukhus.
De tomma listorna står för skapande resp borttagande händelse.

DOMÄN -- `data_object`
NAMN som en domän blir i Prolog-formatet:
`data_object('NAMN',string).`
'NAMN' är domänens namn
string är en datatyp som specificerar att värdet måste vara en alfanumerisk sträng.

Förutom grundbegreppen inom modelleringen finns andra modelleringsbegrepp:

(Notation: Efter - följer en beskrivning.
Efter -- motsvarighet i modellrepresentationen på Prolog-form.)

För OBJEKT finns tex:

TEXTBESKRIVNING - för att precisera och förtydliga objektet när inte namnet räcker till.
-- kan inte representeras på Prolog-form.

IDENTIFIERARE - används för att unikt kunna referera till ett objekt, består av ett eller flera attribut.
-- representeras som en * efter attributnamnet.

DELMÄNGD - används för att generalisera över objekt delmängden ärver samtliga attribut hos den överordnande mängden.
-- fjärde argumentet till predikatet 'entity' innehåller en lista av överordnande objekt.

För DOMÄNER finns tex:

FORMAT - specificerar värdenas utseende (längd och typ).
-- typen framgår av värdeförrådets representation.

VÄRDEFÖRRÅD - för en precisering till ett exakt värdeförråd.
-- andra argumentet till 'data_object' håller en fördefinierad primitiv datatyp (integer, float,char,string eller date).

För ATTRIBUT finns tex:

TEXTBESKRIVNING - för att precisera och förtydliga attributet när inte namnet räcker till.
-- kan inte representeras på Prolog-form.

INVERS - i samband med OBJEKT/OBJEKT-relationer kan två olika perspektiv på förhållandet intas: huvudriktning och inversriktning.
-- ett inversattributnamn kan anges som andra argument till 'attribute'.

MAPPING - reglerar hur många samband som är tillåtna för en viss förekomst av det objekt som attributet tillhör.
Uttrycks med hjälp av min antal och max antal attributvärden för en objektförekomst. Mapping för inversen måste anges. Det sker på samma form.

-- anges som femte argument till attribute på formen:
mapping(MIN1,MAX1,INVERSMIN,INVERSMAX).

3.2 Parsning

En parser i systemet analyserar satsernas syntax och semantik parallellt. Den genereras i Prolog av DCG-reglerna från filerna "grammatik" och "lexikon". Givet en lista av ord, från en sats, som kan analyseras korrekt ger parsern satsens logiska form (LF). Parsningsstrategin blir i enlighet med sökstrategin i Prolog "Top-Down".

Den språkliga analysen av satsen är alltså inte modulär med separat syntaktisk och semantisk analys. Modularitet har många fördelar vad avser generalitet och möjlighet att ändra, utöka, undersöka och använda delarna separat. Framförallt med ett system som bearbetar stora språkfragment.

Eftersom något syntaktiskt parseträd inte behövs inom systemet och den syntaktiska analysen ändå skulle leda till en semantisk analys utförs i stället analyserna parallellt där den semantiska analysen simultant använder resultat från den syntaktiska.

3.2.1 Vilka språkliga konstruktioner förekommer?

Innan någon grammatik kan skrivas måste man veta VAD den skall beskriva. I det här fallet gäller det att beskriva ett antal satser som kan komma att användas vid beskrivning av en konceptuell modell för en sjukhusverksamhet i naturligt språk.

3.2.1.1 Uttryckssättet styrs av modelleringsansatsen

Vilken modelleringsansats som används vid konstruktionen av en konceptuell modell, styr vilket schema som kan skapas. Grundbegrepp som finns för modelleringsspråket avgör den blivande modellens utseende och därmed också formuleringen av satser. Händelsebegreppet saknas t ex för EMOL vilket annars skulle medfört många satskonstruktioner.

Stor vikt har lagts vid mapping i EMOL. Eftersom information om vilken avbildning (mapping) som gäller måste vara entydig så har ett antal nyckelord bestämts för att uttrycka antalet samband av en viss typ från en objektförekomst. Nyckelorden motsvarar artiklar i nominalfraser. Här har en styrning av den blivande modellens utseende även kommit att styra tolkningen av uttrycken för modellering.

EMOL:s avbildning uttrycks som minimalt antal och maximalt antal attributvärden för en objektförekomst (MIN:MAX). OBS! Förväxla inte denna avbildning med den beskriven nedan (+)!

Följande nyckelord har använts för att bestämma exakt avbildning:

(1:1) En bil har exakt en ägare	(minst en och högst en)
(0:0) En bil har ingen ägare	(minst noll och högst noll)
(0:1) En bil har högst en ägare	(minst noll och högst en)
(1:M) En bil har åtminstone/minst en ägare	(minst en kanske fler)
(0:M) En bil har ingen eller flera ägare	(minst noll kanske flera)

Övriga artiklar extraheras inte mapping ur.

(+) Ofta talas om 1 till 1(1:1), 1 till Många(1:M), Många till 1(M:1) och Många till Många(M:M) förhållanden. Detta funktionalitetsförhållande kan erhållas från mapping på EMOL-format genom MAX : INVERSMAX för ett attribut.

3.2.1.2 Satstyper och deras motsvarighet i en konceptuell modell

Normalt kan nominalfraser i grammatiken motsvaras av objekt i den konceptuella modellen liksom verb kan motsvaras av attribut (relationer). De överlägset viktigaste relationerna som ingår i en konceptuell modell av det slag som skall skapas här är delmängdsrelationen ('isa') och 'har'.

Relationen 'isa' används för att beskriva ett semantiskt nätverk (typhierarki) mellan objekten i en modell. Man kan även bl a tänka sig användningen av 'isa' för att beskriva förekomster av objekt som ingår i en objektmängd (objektstyp). Eftersom förekomster inte hanteras här, så kommer 'isa'-relationen bara att gälla mellan objektstyper. För två objektstyper A och B, gäller 'isa'(A,B), om förekomsterna av typ A också är av typ B.

På svenska uttrycks lämpligen 'isa'-relationen med verbet "är". En viktig satstyp vid modellering är följaktligen: A *är* B , där A och B är nominalfraser (se #1).

Attributen i en modell används, förutom till att beskriva relationer mellan objekt, framförallt till att beskriva själva objekten med hjälp av attributvärden. Sådana specificeringar av objekten uttrycks lämpligen i naturligt språk med verbet "har".

Med 'har' vill man kanske uttrycka vilka delar som objektet består av, dvs vilka olika andra objekt som förekommer inom objektets ram.
Till exempel: *Ett sjukhus HAR avdelningar.*
Ett sjukhus HAR anställda.

Eller också vill man definiera egenskaper som är hårt knutna till objektet och som ofta kan användas för att identifiera det. Satser som kan komma på fråga för detta är till exempel: *Ett sjukhus HAR ett sjukhusnamn.*
Ett sjukhus HAR en adress.

I det första fallet motsvarar HAR en relation mellan sjukhus och avdelningar respektive mellan sjukhus och anställda. I modellen motsvaras "verklighetens" sjukhus, avdelning och anställda av objekt med samma namn.

I det senare fallet vill man i modellen koppla namn och adress till en domän där de tillåtna värdena för adress och namn specificeras. Här vill man inte att HAR skall motsvara ett attribut 'har' mellan sjukhus och namn utan istället att sjukhus har ett attribut 'sjukhusnamn' kopplat till en domän NAMN, som är ett värdeförråd.

A *har* B, där A och B är nominalfraser, är en annan vanlig satstyp för modellering (se #2), oavsett vilken betydelse HAR tilldelas enligt ovan.

Observera att det inte finns något antagande om att satssubjektet måste vara levande ("animate"), utan tvärtom görs antagandet att man gärna vill ha "icke-levande" subjekt i satser för modellering. I exemplet ovan är sjukhus subjekt trots att det inte är levande. Av den här anledningen har inga semantiska särdrag av denna typ (t ex animate/inanimate) använts.

Vilka andra satstyper kan förekomma när man vill beskriva en konceptuell modell på svenska? Naturligtvis måste satser med andra verb ingå för att redogöra för relationer mellan objekt.

Transitiva verb kan återge direkta relationer mellan objekt (entiteter). Satssubjektet i en aktiv sats med transitivt verb representeras i modellen av ett objekt som har ett attribut motsvarande verbet till ett objekt som motsvarar satsens objekt (se #3 nedan). Satser med ditransitiva verb har tillsvidare inte beskrivits i grammatiken.

Satser med passiva verb kan användas för att beskriva inversen av attribut, som motsvarar aktiva transitiva verb. Framförallt kan det utnyttjas när man vill beskriva mapping för inversattributet, som måste göras explicit. Alla passiva satskonstruktioner motsvarar inverser i modellen. "Aktiva huvudattribut", som gäller från satsens agent till satssubjektet, skapas dock av de passiva verben (se #4).

För att satser med intransitiva verb i en modell skall motsvara något mer än ett objekt (för subjektet) och ett attribut med okänt värde (för verbet) (se #5) krävs något mer i satsen. Om satsen utökas med ett adverbial i form av en prepositionsfras kan attributet som värde få ett objekt som motsvarar den nominalfras som ingår i prepositionsfrasen. Denna satskonstruktion kan vara speciellt användbar för modellering utan informationsluckor (inga ofärdiga attribut accepteras av systemet). Attributvärdet visar nu var predikatet tar rum. Prepositionen kan tillsammans med verbet utgöra ett tydligt attributnamn som motsvarar en 2-ställig relation (se #6, #7).

Inom följande satstypers ram kan en enkel informationsbas för modellering skapas.

SATSTYPER med exempelsatser	**Kommentarer**
#1. 'isa'	
En patient är en person. *Patienter är personer.*	Dessa satser får samma generiska tolkning.
#2. 'har'	
a. *En avdelning har minst en patient.*	Ger en relation har(Avdelning, Patient).
b. *En person har exakt ett personnummer.*	Ger ett attribut personnummer(Person, NUMMER) där Person är objekt och NUMMER är domän
#3. Sats med transitivt verb	
Ett sjukhus anlitar minst ett labb.	Ger en relation anlitar(Sjukhus, Labb)
En doktor behandlar ingen eller flera patienter.	Ger en relation behandlar(Doktor Patient).
#4. Sats med passivt verb	
Minst en patient behandlas av en doktor.	Ger en aktiv relation behandlar(Doktor,Patient).
#5. Sats med intransitivt verb	
En anställd arbetar.	Ger för objektet (Anställd) attributet arbetar.
#6. Sats med intransitivt verb följt av en prepositionsfras (adverbialfras)	
Personal arbetar på högst en avdelning.	Fastställer relationen arbetar_på(Personal,Avdelning).
#7. Sats med intransitivt verb och framförställd prepositionsfras (adverbialfras)	
På sjukhuset arbetar en sköterska	Ger liksom #6 relationen arbetar_på(Sköterska,Avdelning).

Med satstyperna för huvudsatser ovan skall en sjukhusverksamhet kunna beskrivas och forma kärnan i en konceptuell modell, med fastställda objekt och attribut.

Mappingen extraheras som beskrivet ovan med nominalfrasernas speciella artiklar.

Domänerna i modellen är dock inte så lätta att extrahera ur satser på formen under #2 ovan. Skillnaden mellan a och b under #2 är omöjlig att upptäcka på syntaktiska grunder. Det krävs här modelleringskunskaper för att avgöra om en nominalfras som följer efter verbet 'har' skall vara ett objekt eller ett attribut till en domän. Dessutom är det en bestämd inskränkning att den typen av attribut inte får föregås av annat verb, med utgångspunkt från informella utsagor om sjukhusverksamheten [Ljungberg 89].

Kort sagt räcker inte analys av meningar i naturligt språk till alla gånger för att extrahera tillräckligt med exakt information. För att göra analysen säkrare krävs förutom lexikon och grammatik, som parsern har tillgänglig, även lagrad expert- och världskunskap och deduktiva komponenter för slutsatsdragning.

Avsaknaden av detta kompenseras dels med begränsning av uttrycksmöjligheterna, till satstyperna och inom dessa. Dessutom används ibland direkta frågor (som vid tvetydigheten med satsobjektet till har) för att få information till den blivande modellen.

Omvänt kan vissa satstyper uttrycka lite mer än vad som kan användas till modellkonstruktion. Exempel på det ges nedan

#8. Sats med hjälpverb	
En patient kan behandlas av en eller flera doktorer.	Hjälpverbet återfinns inte i modellen.
#9. Sats med transitivt verb och prepositionsfras (adverbialfras)	
En doktor undersöker en patient på en avdelning.	Ger bara relation undersöker(Doktor,Patient).

3.2.2 Grammatik

Grammatiken är skriven i DCG-formalism (Definite Clause Grammar). Den bygger på en grammatik för "Talk" [Pereira 87] och beskriver både syntax och semantik.

3.2.2.1 Syntax

Syntaxbeskrivningen i grammatiken redogör för de vanligaste deklarativa satserna. I en text inleds vanligen 2/3 av huvudsatserna med subjektet. Näst vanligast är att huvudsatsen inleds med tids, sätts- eller rumsadverbial. I grammatiken analyseras endast deklarativa satser (decl). Enligt grammatiken följs ett inledande subjekt av en verbfras (vp) och eventuellt en adverbialfras (ap).

Vid intransitivt huvudverb som predikat kan satsen börja med adverbialfras. En prepositionsfras, som adverbialfras, kan skapa en binär relation vid en sats med intransivt verb som ju inte innehåller något"grammatiskt objekt". Därför är denna konstruktion intressant ur modelleringssynpunkt (se satstyp # 6 och # 7.3.2.1.2). Den satskonstruktionen avslutas med subjektet.

En komplett sats måste innehålla både subjekt och predikat (verbfras). Verbfrasen måste innehålla finit verb i huvudsats. Den kan bestå av intransitivt verb eller transitivt verb med "grammatiskt" objekt.

För att i grammatiken undvika vänsterrekursion, med regeln vp --> vp,ap, så är inte adverbialfrasen knuten till verbfrasen.

Transitiva verb och **verb med partikel** kan ge upphov till såväl aktiva som passiva verbfraser. **Bindeverb,** sk copula (är), som följs av en nominalfras som predikatsfyllnad, bildar en annan typ av finit verbfras. **Infinita (oböjda) verbfraser** kan föregås av finit (böjt) hjälpver. Märk dock att grammatiken inte tillåter hjälpverb för "är" eftersom deras betydelse inte är tolkad och verkar vara mer avgörande vid t ex:
Doktorer kan vara personer än vid
Doktorer kan arbeta vid ett eller flera sjukhus.

Nominalfraser innehåller tillsvidare inga pronomen eller egennamn. För att beskriva en konceptuell modell med typer av objekt är de inte så användbara. De generella utsagor som då kommer på fråga skulle ha bäst användning av pronomenet "det" som utfyllnadssubjekt i satser med "Det finns ..." (en utökning av grammatiken beskriver lämpligen denna konstruktion).

Adjektiv har utelämnats med antagandet om att dessa bara behövs till regler i modellen. Substantiv i bestämd form singularis och i pluralform behöver ingen artikel och ger upphov till speciella regler (np). Alla substantiv kan ha relativbestämning med optionella relativbisatser.

Subjektsrelativ består av ett relativpronomen följt av en verbfras:
(*Mannen* REL(RELPRON(*som*),VP(*går*))).
Objektsrelativ: består av relativpronomen och en sats med utelämnat objekt.
(*Mannen* REL(RELPRON(*som*),DECL(NP(*pojken*),VP(*jagade*,gap(np,X)))).

Utelämnade konstituenter hålls reda på i grammatiken med ett argument "GapInfo" som ingår i alla regler som kan leda till np(nominalfras). Om en konstituent saknas markeras det med en konstant 'gap(NT,Var)' ('gap(np,X)' för np), där NT är icke-terminalen som saknas i ursprungsposition och Var är LF-variabeln (tillika Prolog-variabeln) som binds till platsen där konstituenten återfinns.

Om en konstituent inte får utelämnas låses argumentplatsen med en konstant 'nogap'. Eftersom adverbialfrasen inte återfinns under vp måste en speciell regel för decl skrivas för subjektsrelativsats med ap, som ger utrymme för en sats att sakna subjekt, när det finns ett objekt.
Inga andra nominalbestämningar (reducerade relativsatser som prepositionsfraser o dyl) beskrivs i grammatiken.

3.2.2.2 Semantik

När den semantiska analysen sker simultant med den syntaktiska så kan ett syntaktiskt parseträd bli överflödigt. Åtminstone för den här applikationen där konstruktionen av en konceptuell modellrepresentation bygger på en logisk form av satser. Nu byggs den logiska formen av de analyserbara satserna upp direkt under parsningens gång. Som grammatiken är skriven bildas inget syntaxträd, men den kan lätt utökas med det om man vill se resultatet av den syntaktiska analysen.

Den semantiska analysen (tillika konstruktionen av satsernas logiska form) bygger på Montague-semantik [Dowty 81]. Central för den läran är "kompositionalitetsprincipen", som förkunnar att tolkningen av en sats (eller fras) kan fås ur tolkningen av dess delar. Under den semantiska parsningen används denna princip för att rekursivt komma åt den logiska formen för delar av satsen som sedan unifieras till hela satsens logiska form. I konventionell Montague semantik skulle den generering, (från den logiska formen av satsens terminaler till större satsdelar) som Prolog hanterar med unifiering, motsvara funktionell applikation.

I Montague-semantik anammas en sk "rule-to-rule hypothesis" som innebär en entydig relation mellan syntax och semantik. Hypotesen går nämligen ut på att det för var syntaktisk regel finns en semantisk regel för motsvarande konstituenter. Till exempel finns för den syntaktiska regeln:

(1) `S --> NP, VP`

den semantiska regeln:

(`1) Om den logiska formen av `NP` är `` `NP `` och den logiska formen för `VP` är `` `VP ``, så är den logiska formen för `S = `NP (`VP)` (` står här för icke-terminalens logiska form).

Montague använde en mellanform vid översättning av NL-satser till modellteoretisk tolkning kallad intensionell logik. Den motsvarar en högre ordningens logik, baserad på typad Lambda-kalkyl.

Den logiska formen som används i modelleringsgrammatiken motsvarar Första Ordningens Logik (FOL) utökad med otypad Lambda-kalkyl och speciella kvantifikatorer för mapping.

Lambda-operatorn används för att kunna abstrahera över argument till en funktion. Ett exempel får belysa:

λx.(x+2) är ett uttryck för funktionen (x+2) som tar x som argument
λx.(x+2)(1) är ett uttryck för applikationen av funktionen ovan på argumentet 1 dvs (1+2)

Beta-reduktion kallas den allmänna princip som vi just tillämpade (1+2). Den lyder:

Ett uttryck på formen λx.(ø)a kan reduceras till det semantiskt ekvivalenta uttrycket ¥, där ¥ är ø med alla förekomster av x utbytta mot a.

Som Lambda operator i Prolog används ^, en infix operator. Den logiska formen av ett transitivt verb "behandla" skrivs i Prolog som: `X^Y^(`behandlar(X,Y)`, annars λxλy.`behandlar(x,y). ^ är alltså högerassociativ. Beta-reduktion kan i Prolog beskrivas med klausulen: `reduce(Arg^Expr,Arg,Expr)`, där första argumentet är funktionen och sista argumentet utgör resultatet.

Resultatet av:
reduce(X^`jobbar(X),`Dyrgård',LF) blir LF=`jobbar(`Dyrgård)
En DCG-regel som hanterar både syntax och semantik skrivs:
s(S) --> np(NP),vp(VP), {reduce(NP,VP,S)}. Här håller variablerna (argumenten med versaler) den logiska formen (för enkelhetens skull ej `försedda) för varje syntaktisk kategori. Inom krullparenteser återfinns ett "rent" Prolog-anrop.

Ännu enklare kan reglerna bli med partiell (för-)exekvering, dvs förändringar i programmet som motsvarar händelseförloppet vid exekvering. I den ovanstående regeln vet vi att *reduce* ser till att NP har formen VP^S. För detta behöver vi inte anropa *reduce* utan istället låta kravet finnas klart i regeln, "partiellt exekverat m a p reduce" enligt:
`s(S) --> np(VP^S),vp(VP).` Alla regler i modelleringsgrammatiken är partiellt exekverade.

För Prolog-representationen av FOL och LF i grammatiken gäller: FOL uttryck, formler såväl som termer i FOL motsvaras av Prolog-termer. T ex:
`all(X,`doktor(X) => `jobbar(X))`

Funktionssymboler i Prolog används för att representera FOL:s funktionssymboler, predikat och kvantifikatorer. Konnektiver beskrivs med binära infix funktorer (implikation =>, konjunktion &). Som variabler i FOL används Prolog-variabler.

Nedan följer en förteckning över de olika grammatiska kategorierna och deras logiska form.

	syntaktisk kategori	**semantisk form**
`sentence`	huvudsats	proposition
`decl`	deklarativ sats	proposition
`terminator`	satsavslutare (. för decl)	
`np`	nominalfras	funktion fr VP t S(sats)
`vp`	verbfras	funktion fr individ t S
`ap`	adverbialfras	funktion fr S t S
`pp`	prepositionsfras	funktion fr S t S
`preposition`	preposition	funktion fr NP t PP
`iv`	intransitivt verb	funktion fr individ t S
`tv`	transitivt verb	funktion fr NP t VP
`aux`	hjälpverb	funktion fr huvudVP t VP
`passv`	passivt verb	funktion fr NP t VP
`copula`	bindeverb t predikatsfyllnad	funktion fr NP t VP
`partv`	verb med partikel	funktion fr NP t VP
`vpart`	verbpartikel	funktion fr VP t VP
`det`	artikel	funktion fr N' t NP
`n`	substantiv	funktion fr individ t S
`optrel`	optionell relativsats	funktion fr N till N'
`relpron`	relativpronomen	

där VP betyder den logiska formen av den syntaktiska kategorin vp osv. (Prim (') betecknar här, en grammatisk kategori (enligt Chomskys X-bar teori) som motsvarar en nivå (1-bar) ovan lexikal kategori (0-bar). Fraskategorier räknas oftast som 2-bar-kategorier (" enligt denna notation för bar). N' är alltså en kategori mellan fras(NP,N") och lexikal(N)). Individer motsvarar den logiska formen av egennamn. Inga egennamn finns dock i grammatiken. Kategorier vars semantiska form innehåller individer utgör alltid argument vid den funktionella applikationen.

3.2.3 Lexikon

Ett lexikon måste specificeras med såväl "allmänspråkliga" som domänspecifika (sjukhus-) ord.

Alla pre-terminaler, dvs icke-terminaler vars beståndsdelar är lexikala ord, finns representerade tillsammans med lexikonet. De utgör övergången mellan grammatikreglerna och lexikon med rena Prolog-klausuler (inom krullparenteser) som sköter lexikonuppslag.

I lexikonfilen återfinns även de specificerade operatorerna som används för beskrivning av logisk form: & - konjunktion , => - implikation.

För ordklasser som ingår i lexikonet följer nedan en kort beskrivning.

ARTIKLAR är avgörande för exakt mapping. Tolkningen av artiklarna har därmed styrts. Framförallt tilldelas vissa "nyckelartiklar" en bestämd logisk form för mapping i form av speciella "mappingkvantifikatorer". Artiklarna som fungerar som "nyckelartiklar" och deras tolkning är:

exakt en	(X^S1)^(X^S2)^(one(X,S1 & S2))
ingen	(X^S1)^(X^S2)^(no(X,S1 & S2))
högst en	(X^S1)^(X^S2)^(max_one(X,S1 & S2))
åtminstone/minst en /en eller flera	(X^S1)^(X^S2)^(min_one(X,S1 & S2))
ingen eller flera	(X^S1)^(X^S2)^(any_num(X,S1 & S2))

Övriga artiklar (en,någon,flera,alla,var,varje) har en FOL-tolkning i form av (X^S1)^(X^S2)^(exists(X,S1 & S2)) eller (X^S1)^(X^S2)^(all(X,S1 => S2)).

Inga artiklar i bestämd form finns upptagna eftersom utsagorna redogör för objektstyper och därmed nominalfraser med generisk tolkning. Några synonymer finns tillgängliga motsvarande 'min_one' för flexibilitet.

I lexikoningångarna lagras hela frasen för de artiklar som består av flera ord. Där ges bestämning på numerus, genus och species för att artiklarna skall kongruera med substantiven de bestämmer. Tillsammans med detta finns dessutom den logiska formen av artiklar som gör kvantifikatorn till huvudfunktor av den logiska formen för en hel sats.

Den logiska formen för artiklar kan ses som funktioner från N' till NP, (dvs N' ^ (VP ^ S)).

SUBSTANTIV delas i lexikon upp efter förekomsten respektive frånvaron av regelbunden böjning. Oregelbundna substantiv lagras med sin grundform (obestämd form singularis) och information om numerus och species. Grundformen av alla substantiv (oregelbundna och regelbundna) finns upptagna med genusinformation och logisk form. Via den specificerade grundformen kan information om genus och logisk form extraheras för oregelbundna substantiv.

För regelbundna substantiv representeras de andra böjningsformerna av ordet med hjälp av morfologiska regler. Under analys kan ett regelbundet substantiv "härledas" till sin grundform (med predikatet noun_suffix), som sedan matchas i det explicita lexikonet. Att böjningsformen är

korrekt styrs av en markering för substantivets deklinationsgrupp i lexikon. Deklinationsgrupperna (böjningsgrupperna) är bestämda av böjningsändelsen i obestämd pluralform:

deklinationsgrupp	ändelse i plural obest.
1	-or
2	-ar
3	-er
4	-n
5	-

Den logiska formen av substantiv motsvarar funktioner från individer till propositioner (X^S) där X är en individ.

VERB i lexikon förekommer oftast i presens och infinitiv tempusform tillsammans med verbets logiska form.

För hjälpverben (aux) finns bara presensform, vilket inskränker verbfraser med hjälpverb till att innehålla två verb (inga konstruktioner som: "Han måste få jobba"). Den form som hjälpverbet kräver av det andra verbet specificeras i lexikoningången för hjälpverbet på formatet: hjälpverbets form/ huvudverbets krävda form.

För transitiva verb och för verb med partikel markeras om verbformerna är i aktiv eller passiv. Partikelverben har också information om vilken verbpartikel som krävs av verbet. Bindeverb (copula) slutligen representeras precis som transitiva verb.

Den logiska formen för intransitiva verb är en funktion från individer till propositioner (X^(V(X)), där V är ett logiskt basuttryck för verbet t ex `prata, (här motsvarande en funktionssymbol i Prolog).

Hjälpverbens logiska form är HVP^VP (dvs funktioner från logiska formen av verbfraser till detsamma) där HVP är verbfrasen för huvudverbet.

Övriga verb har den logiska formen X^Y^(V(X^Y)). Här motsvarar V en relation i FOL mellan X och Y och den är alltid aktiv. Det passiva verbets logiska form uttrycks med omkastade argument som X^Y(V(Y^X)). Jämför den aktiva och den passiva satsens logiska form i:

Aktiv *En doktor behandlar en person.*

```
exists(X,`doktor(X) & exists(Y,`person(Y) & `behandlar(X,Y)))
```

Passiv *En person behandlas av en doktor.*

```
exists(X,`person(X) & exists(Y,`doktor(Y) & `behandlar(Y,X))).
```

Den logiska formen av satser med verbet "är", som ger upphov till isa-relationen, är långt ifrån självklar. Representationen av en sats *"En doktor är en person"*, torde i Första Ordningens Logik skrivet med Prologtermer likna: `all(X,doktor(X) => person(X))` Representationen av 'person' skiljer sig inte från ett 1-ställigt FOL-predikat 'jobbar': `all(X,doktor(X) => jobbar(X))`.

För att undvika tvetydigheter av detta slag, som med möda måste specialanalyseras, har inte den representationen använts. Den generiska tolkningen av nominalfraser med en/ett som artikel (liksom nomen i plural utan artikel) ger upphov till exists som kvantifikator i analysen. Om

allkvantifiering skulle användas, så skulle analysen av nominalfraser i objektsposition bli underlig. Detta passar de flesta fall men är ännu en svårighet för att nå ovanstående representation.

Representationen som används tillsvidare är (inte så logiskt tilltalande) samma som för transitiva verb, vilket ger:

```
exists(X,`doktor(X) & exists(Y, `person(Y) & `isa(X,Y)))
```

Här framgår inte att isa-relationen egentligen är en relation "på metanivå" mellan mängder (begrepp) (här ungefärligt isa(patient, person)).

Främsta motivationen för detta är att formen i den fortsatta analysen blir lättkonverterad enligt samma mönster som andra binära relationer. Argumenten används för att visa relationens riktning. I nästa grammatikbeskrivning får möjliga lämpliga representationer av 'isa'-relationen granskas. Ett alternativ kan vara en lambda-funktion från egenskapspredikat.

Observera att synonymer i lexikon erhåller en gemensam LF (Logisk Form).

3.3 Konvertering

Konverteringen utgår ifrån LF (Logisk Form) av en sats och resulterar i globala strukturer av Prolog-predikat för en KM (se avsnitt 2.3).

LF representeras som Prolog-primitiver av den variant av FOL som används. Formler såväl som termer i FOL motsvaras av Prolog-termer, t ex

```
exists(X,patient(X) & ligger(X))
```

Funktionssymboler i Prolog används för att representera FOL:s funktionssymboler, predikatsymboler liksom kvantifikatorer. Med Prologdeklarerade binära infix funktorer kan konnektiv och implikation beskrivas. Slutligen används Prolog-variabler som variabler i FOL.

Ur den logiska formen vill vi extrahera följande:

- Predikaten som skall utgöra attribut eller objekt i KM.
- De speciella "mappingkvantifikatorerna" som ger mapping för ett attribut.
- Variablerna som utgör argument till predikaten.

Den logiska formen av en sats kan ses som en semantisk trädstruktur. Denna semantiska struktur traverseras djupet först, vänster till höger. Sist i traverseringen av hela trädet (delträd för relativsats), återfinns det i normalfallet blivande attributet. För att kunna skapa en Prolog-instans av predikatet 'attribute' krävs att objektet som har attributet och attributvärdet är åtkomliga i LF-representationen. Dessutom skall argumentet för mapping instantieras enligt eventuella "mappingkvantifikatorer".

Denna information samlas under traverseringen såsom värden på ärvda (från föräldra- eller syskonnod) eller syntetiserade (från barnnod) särdrag i trädet. Prolog-argumenten, för predikaten som konverterar delträden, motsvarar dessa särdrag som efterhand instantieras, utökas och unifieras med särdragsvärden till att slutligen innehålla en lista på potentiella objekt med variabler respektive funnen kvantifikator.

Från den logiska formen (LF) extraheras KM-representationen (2.3) enligt följande:

OBJEKTNAMN
Egenskaperna (1-ställiga predikat) som oftast skall bli objekt extraheras och sparas på formen (Ent,Arg) i en lista, där Arg är argumentet till det 1-ställiga predikat som egenskapen (Ent) utgör. Arg används för att följa variabelbindningen till slutet av traverseringen (då aktuella egenskaper måste vara kända).

ÖVERORDNADE OBJEKT
2:a argumentet (Y) i en 'isa'-relation: isa(X,Y) konverteras till element i listan av överordnade objekt till X

DOMÄNNAMN
Genom dialog i (check_domain) ger användaren DOMÄNNAMN och TYP. En domän kan bara vara ett attributvärde vid LF med relationen 'har'.

ATTRIBUTNAMN
Sista 2-ställiga relationen i traverseringen av en sats (ej 'isa' och ibland undantaget specialrelationen 'har'). Vid 1-ställigt predikat slås det ihop med prepositionen som är strukturens huvudfunktor t ex: arbetar_på.

INVERSNAMN
Lagras under predikatet inv(ATTRIBUTNAMN,INVERSNAMN).

ATTR_URSPRUNG
Agenten för huvudpredikatet-attributet (1:a argumentet i relationen) ger Arg. I listan med (Ent,Arg) återfinns egenskapen som skall bli OBJEKTNAMN.

ATTR_DESTINATION
Vid 1-ställigt predikat har LF av satsen som kan konverteras alltid preposition som huvudfunktor. Dess första argument är LF för adverbialprepositionsfrasens np(nominalfras). Destinationen blir egenskapen av detta 1:a argument (som i listan är sparat som (Ent,Prep) där Prep är huvudfunktorn). Vid 2-ställigt predikat motsvaras "attributdestinationen" av egenskapen för relationens 2:a argument.

MAPPING
På formen mapping(MIN,MAX,INVMIN,INVMAX).

Speciella kvantifikatorer(MK) för mapping: (one, no, max_one, min_one, any_num) extraheras, lagras som sista argument(Quant) och bestämmer kombinationerna av 0,1 och M för mapping av attributet.

Högst en "mappingvantifikator" förväntas eftersom förhållandet uttrycks:
Ett X "relation" ANTAL-Y.
Bestämningen om mappingkvantifikatorn som gäller för huvud- eller inversattributet är gjord utgående från LF med 2-ställiga relationer (utan Prep).

Följande satsexempel med transitivt verb får belysa huvud- och inversmapping. MK som används i exemplen är exakt en=one:

Huvudmapping

1.a. MK för objekt: *En doktor behandlar exakt en patient.*
LF: exists(X,`doktor(X) & one(Y,`patient(Y) & `behandlar(X,Y)))

b. Passiv & MK för subjekt: *Exakt en patient behandlas av en doktor.*
LF: one(X,`patient(X) & exists(Y,`doktor(Y) & `behandlar(Y,X)))

Inversmapping

2.a. Passiv: *En patient behandlas av exakt en doktor.*
LF: exists(X,`patient(X) & one(Y,`doktor(Y) & `behandlar(Y,X)))

b. MK för subjekt: *Exakt en doktor behandlar en patient.*
LF: one(X,`doktor(X) & exists(Y,`patient(Y) & `behandlar(X,Y)))

Mappingbestämningen antar LF av aktiv verbform som grundform. Därför gäller att MK för 1:a argumentet märks med funktorn 'invmap' om ingen mappingkvantifikator finns för andra argumentet som påträffas i traverseringen (som 2.b).

MÄRK dock att om verbformen är passiv och satssubjektet är "mappingkvantifierat" bli det "huvudmapping" ("dubbel invers" som neutraliserar). Detta gäller för logisk form av sats med transitivt verb.

För logisk form av en sats som börjar med prepositionsfras blir markeringen 'invmap' också bakvänd, ty attributets destination (np i pp) återfinns före ursprunget (agenten).

Om predikatet är en relation (2-ställigt) kan man av variablerna bestämma vilket objekt i listan som är attributursprung respektive attributvärde. Detta görs genom att extrahera relationens semantiska agent och objekt. Agenten är alltid attributets ATTR_URSPRUNG och objektet alltid ATTR_DESTINATION.

Däremot behöver inte agenten motsvara subjektet i en sats (som påträffas före vid traverseringen av LF-strukturen för satsen och hamnar senare i listan av blivande OBJEKTNAMN). Vid passiva verb motsvaras satssubjektet av relationens andra argument. LF för passiva verbformer är nämligen X^Y^(`behandlar(Y,X)), där X är satssubjekt. Om listan av blivande OBJEKTNAMN innehåller t ex [(person,Y),(doktor,X)] och relationen är `behandlar(X,Y)`, så är 'doktor' attributursprung. Om relationen i stället är passiv, `behandlar(Y,X)`, så är 'person' attributursprung.

Två speciella relationer är 'isa' och 'har'. Ett objekts överordnade objekt ges av 'isa' (som är en delmängdsrelation). T ex: listan av blivande objekt innehåller [(person,Y),(patient,X)] och relationen är isa(X,Y). Denna ger inte upphov till ett attribut utan en utökning av listan med överordnade objekt för patient enligt instansen:
entity(patient,[],[],[Person]).

Relationen 'har' kan vara en indikator på ett attribut till en domän i modellen. Om inte tidigare information finns i modellen om relationens argument, så tillfrågas modelleraren (vars input nedan visas i kursiv stil).

T ex:
```
Ett sjukhus har avdelningar.
Är avdelning attribut till en domän? j/n
```

Nej-svar ger:

```
attribute(har,_,'Sjukhus','Avdelning',mapping(_,_,_,_),[],[]).
```

Eller:
```
Ett sjukhus har exakt ett namn.
Är namn attribut till en domän? j/n
```

Vid ja-svar får namnet och typen för domänen specificeras. Resultatet blir förutom att domänen (data_object('NAMN',string)) lagts till:

```
attribute(namn,_,'Sjukhus','NAMN',mapping(1,1,_,_),[],[]).
```

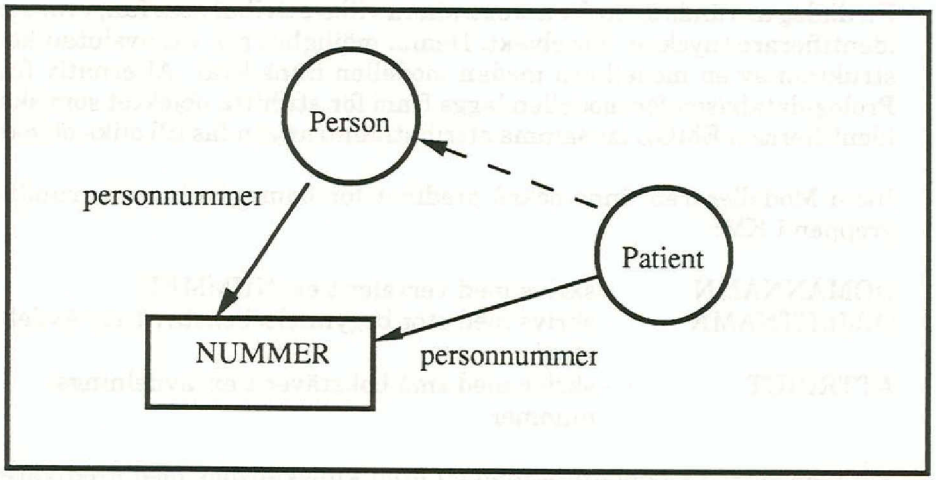
Om predikatet är 1-ställigt (som i LF av satsen *En patient ligger på exakt ett sjukhus*) skapas attributet av kombinationen verb_preposition. Prepositionen sparas under konverteringen i egenskapslistan som objektets argument. Instansen som skapas blir alltså:

```
attribute(ligger_på,_,'Patient','Sjukhus',mapping(1,1,_,_),[],[]).
```

Se APPENDIX A för mer exempel NL-KM.

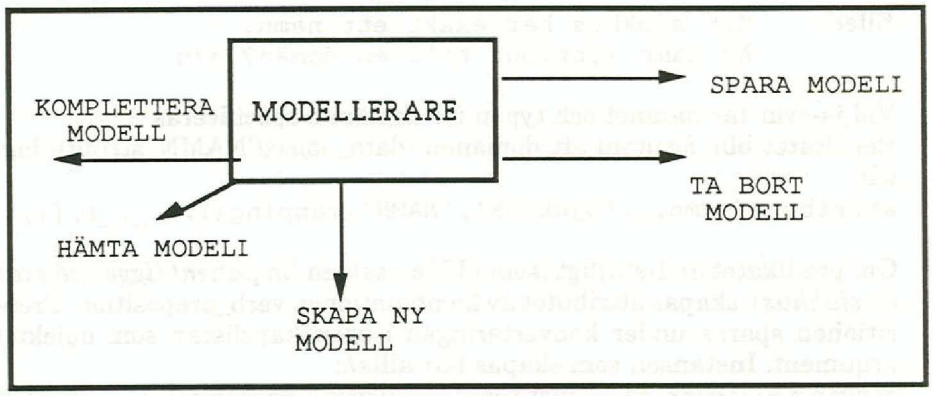
Inom Konverteraren kontrolleras ingen konsistens av modellen. Den enda restriktion som finns är att det inte får finnas flera förekomster av samma del av en KM.

Att viss information kan finnas implicit får användaren av systemet kontrollera. Till exempel kan redundans uppstå som i figur 5. Här har objektet 'Person' ett attribut 'personnummer'. En 'isa'-relation (streckad pil) är fastställd mellan objektet 'Patient' och 'Person'. Systemet hindrar inte att samma attribut 'personnummer' ges till 'Patient', trots att 'Patient' kunde ha ärvt detta attribut av 'Person' (Fig 5).



Figur 5. Systemet hindrar inte att information som kan härledas, såsom ärvda attribut, också representeras explicit.

3.4 Modellerare - användargränssnitt till systemet



Figur 6.

Modelleraren, som utgör systemets användargränssnitt, innehåller predikat för att skapa, spara, hämta, komplettera och ta bort modeller (se 3.5).

Gränssnittet bygger på en interaktiv användning av systemet. Vid konverteringen från LF till KM kan tvetydigheter uppstå som användarna på dialogväg genom Modelleraren får lösa.

Vid 'har'-relation startas en sådan dialog för att utröna om relationens objekt också är ett KM-OBJEKT eller annars ett domänvärt attribut. I det sistnämnda fallet får domännamn och domäntyp extraheras i dialogen.

En dialog används även för att extrahera vilka attribut som fungerar som identifierare (nycklar) för objekt. Denna möjlighet ges vid avslutad konstruktion av en modell och medan modellen finns kvar. Alternativ från Prolog-databasen för modellen läggs fram för att hitta objektet som skall identifieras (i EMOL får samma attributnamn användas till olika objekt).

Inom Modelleraren finns också predikat för namngivning av grundbegreppen i KM:

DOMÄNNAMN	- skrivs med versaler t ex 'NUMMER'
OBJEKTNAMN	- skrivs med stor begynnelsebokstav t ex 'Avdelning'
ATTRIBUT	- skrivs med små bokstäver t ex 'avdelningsnummer'

Sluligen ryms i modellerare-filen en liten kunskapsbas med predikatet inv(ATTRIBUTNAMN,INVERSNAMN) för lagring av lämpliga inversnamn.

3.5 Användarhandledning

Systemet körs under SICStus Prolog som startas från UNIX med `sicstus`. Promptern från Prolog interpretatorn `|?-` visar att det är klart för indata. För att systemet skall bli körbart så måste programfiler konsulteras. Skriv på följande vis: `[nl_km].` Punkten på slutet är ett måste när Prolog används och förekommer i hela interaktionen!

Vid ny prompter startar du systemet genom att skriva `modellera`.

Nu kan du:	genom att skriva:
• Skapa en ny modell	skapa_ny.
• Spara den senast skapade modellen	spara.
• Hämta en sparad modell	ta_fram.
• Komplettera en modell (hämtad eller senast skapad)	komplettera.
• Ta bort en modell som nyss skapats eller hämtats	ta_bort.
• Visa den senast skapade eller hämtade modellen	visa_modell.
• Specificera identifierande attribut i den senaste modellen	identifiera.

När en ny modell skapas försvinner den senaste. Om detta inte är önskvärt sparar du modellen först. Modellen skapas genom att upprepat skriva in utsagor på svenska som beskriver sjukhusdomänen. Om den språkliga analysen av satsen (parsningen) misslyckas, meddelas det. Annars ges inget felmeddelande.

För varje sats som konverteras till en del av en konceptuell modell presenteras modellen som hittills har skapats. När du inte vill utöka modellen mer avslutar du skapandet genom att skriva `sluta`.

När en modell hämtas så finns den senast definierade modellen kvar. Om du inte vill slå ihop den senaste modellen med den som skall hämtas så får du ta bort den förstnämnda innan den lagrade tas fram. Genom att skriva 'identifiera.' kan du bestämma identifierande attribut efter det att en modell har skapats eller hämtats.

När du anger attributnamnet så ges förslag på möjliga objekt som attributet kan identifiera. Det får accepteras med j (ja) eller förkastas med n (nej). Identifierande attribut markeras med en * i slutet av attributnamnet sedan objektet som det skall identifiera är fastställt.

Domäner i modellen kan bara skapas vid frågeställning från systemet om något är ett domänvärt attribut. Situationen uppkommer vid 'har'-relation och frågan besvaras n (nej) om 'har' skall utgöra attribut, annars j (ja).

T ex besvaras frågan med j vid satsen : *En person har ett personnummer.* Där är 'personnummer' ett domänvärt attribut. Vidare skall domänen som värdet tas ifrån namnges. T ex kan vi för denna domän skriva `nummer`.

En primitiv datatyp skall också specificeras för värdeförrådet. Som typer används `integer, float, char, string, date` eller `_(okänt)`.

Observera hur domännamnet i modellrepresentationen får stora bokstäver, oavsett hur det skrivs in. Namnkonventionen som används är - stora bokstäver för DOMÄNNAMN, stor begynnelsebokstav för Objektnamn och små bokstäver för attributnamn.

När du vill återvända till Prolog skriver du `sluta`. Kommandot `sluta_helt` avslutar Prolog samtidigt med programmet.

(Prolog-interpretatorn avslutas med `halt`).

4. Analys

Kärnproblemet när naturligt språk kombineras med datorsystem är att man ur oklar och ostrukturerad information i form av språk vill extrahera och omforma till något som är tillräckligt precist, formellt och strukturerat för att kunna användas i ett datorsystem.

Även vid all form av modellering så är abstraktion och precisering liknande den ovanstående viktiga företeelser eftersom:

- en förenkling av det som modellen avbildar återges i modellen.
- man väljer att bara avbilda det som är intressant för modellens syfte.
- modellen och det som modellen avbildar alltid är skilda saker.

Detta innebär en begränsning på gott och ont. Vilka modeller som kan representeras i systemet beror förutom på verksamhetsdomänen till stor del på modelleringsansatsen (EMOL) och representationen av modellen (Prolog-predikat).

4.1 Domänkunskap

För kunskap om domänen användes en enklare konceptuell modell av en sjukhusverksamhet, som har skapats från en relationsdatabas, (enl 1.4) och ett lexikon med domänspecifika termer. Utan dessa kunskapskällor skulle det ha varit svårt att göra rimliga antaganden om vad som är intressant att uttrycka om verksamheten för modellering med systemet. Ett domänlexikon är dessutom nödvändigt för systemet. Om inte kunskapen i form av lexikon eller databas funnits tillgänglig skulle uppbyggnaden av ett lexikon med domänspecifika ord kräva en analys av "verklighetens sjukhusverksamheter".

Med specifikationen av ett domänlexikon skapas redan en viss modellering i det avseendet att abstraktionen och aspekten för avbildningen av objektssystemet görs klar. Det som systemet inte kan hantera kanske inte har betraktats som intressant. Speciellt är detta fallet med domänspecifika lexikon som specificerar tillåtna komplement till ord.

Vitsen med system för NL-gränssnitt till modellering blir starkt förminskad om en omfattande analys för varje domän, som man vill modellera med hjälp av gränssnittet, vore nödvändig bara för att kunna fastställa domänlexikon och annan domänkunskap till systemet.

Däremot kan man samla generell expertkunskap från modellerare och existerande modeller som utgångsmaterial för modellering inom olika domäner i ett system.

Vad avser domänkunskap erhålls förmodligen den bästa lösningen med en parallell utveckling av konceptuell modell och lexikon. På så sätt undviks en hel del dubbelarbete med att extrahera domänkunskap. Detta om syftet med systemet verkligen är design av en modell framför ett samlingverktyg för olika användare med olika modelleringskunskap. Om flera användare är inblandade i analysen av ett objektssystem kan eventuellt en styrning vara önskvärd med vissa avgränsningar för vissa användare beroende på domän- och modelleringskunskap.

4.2 Analys av EMOL

En uppsättning modelleringsbegrepp och regler för hur de får kombineras kan sägas utgöra ett modelleringsspråk (modelleringsansats) [Bubenko 84]. Det är långt ifrån klart vilka modelleringsansatser som är de bästa eller ens bra.

Modeller som är resultat av olika analyser har olika ambitioner. Vilka modelleringsbegrepp i form av objekt, attribut osv som behövs för att formulera en modell för analys eller konstruktion är en vanlig frågeställning inom informationssystemutveckling.

EMOL är ett språk för datamodellering och skall som sådant bäst användas till analys av data i en inledande verksamhetsanalys. Vid sidan av datamodellering i detta skede görs en funktionsanalys för processer i verksamheten.

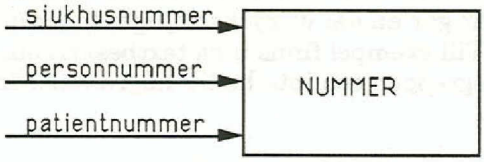
Vid datamodellering vill man undvika systemrelaterade beskrivningar. Modellen skall beskriva data, vad data står för och hur olika data är relaterade. Eventuellt vill man göra en modell som kan förändras med tiden. I så fall måste modellen beskriva hur data får förändras. Händelsebegreppet för tidsaspekt saknas i EMOL liksom i de flesta modelleringsansatser. Med dialogsteg i EMOL kan dock viss tidsaspekt styras: objekt kan skapas och tas bort, och attribut kan tilldelas värden. En modells föränderlighet i tiden berör i största utsträckningen företeelser i modellens informationsbas men ibland förändras även det som tillhör regel- och begreppssystemet.

EMOL fokuserar i sin syntax på grafisk representation. När modellering med domänvärda attribut närmar sig ett databassystem kan det vara önskvärt att i tabellform få beskriva ett objekt. Där får attributvärden för varje attribut framgå tydligt liksom identifierare (primärnyckel) och "främmande" nycklar (foreign keys) till objektet. Större modeller med enbart grafisk notation blir mycket svåröverskådliga.

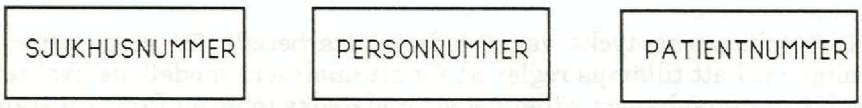
Grundbegreppen i EMOL kan diskuteras.

Domänbegreppet är bra. Det gäller att utnyttja det på bästa sätt med få generella domäner som värdeförråd för attribut. Specificeringen av attributvärden från domänen kan i stället göras med attributen.

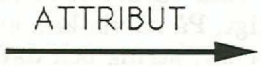
T ex är flera attribut med samma domän:



Att föredra framför flera domäner:



Objekt i EMOL motsvarar det som ofta benämns entitet. Något som man vill hålla information om. Objekten kan identifieras med attribut. Konstigt nog kan samma attributnamn förekomma på attribut för olika objekt. Detta kan starkt ifrågasättas eftersom det gör den unika identifieringen (att hitta en primärnyckel) av ett objekt mycket svårare. Objekt kan dock inte ha samma namn som domäner vilket verkar vettigt.



Attributet består i EMOL av domänvärda- och objektsvärda attribut. Att sammanföra dessa till ett begrepp känns onaturligt. Attributen vars värden tillhör en domän definierar objekten och redogör för egenskaper. Attribut till objekt visar istället på samband mellan olika objekt.

Mycket bättre vore det att ha två olika primitiver för dessa.

- Ett begrepp för förhållanden mellan objekt ("relationer") och
- Ett begrepp för domänvärda attribut, som identifierar objekt med dess ofta funktionella attributvärden (i en bra modell).

Mapping för attributen är inte så lätta att läsa ut alla gånger. Om det vare sig är av intresse att kunna ange exakt hur många attributvärden som är möjliga (nogrannare än M) eller att skilja på 0 och 1 som minimum kunde det räcka att ange om avbildningen är 1 till 1,osv 1:M, M:1 eller M:M. Det som är viktigast att extrahera är om man kan identifiera något unikt (med hjälp av funktionalitet), samt om en relation gäller för alla individer i en klass (totalitet).

4.3 Analys av Prolog-representationen för KM

Att övergå från en representation till en annan innebär nästan alltid en viss mängd förlorad information. Ibland kallas det abstraktion och är önskvärt, ibland utgör det en oönskad förlust.

I systemet används ännu en representation för konceptuella modeller. Grundbegreppen inom ett modelleringsspråk representeras som Prolog-predikat och argumenten för predikaten rymmer primitiver som berör grundbegreppen. Här går en del uttrycksmöjligheter från modelleringsansatsen förlorade. Till exempel finns inga textbeskrivningar för ytterligare definition av begreppen och inte heller några härledningsregler för attribut från EMOL.

Prolog-representationen av KM kan naturligtvis betraktas från den omvända synvinkeln - att det är fantastiskt hur så mycket kan rymmas inom en sådan enkel notation.

Prologstrukturerna tycks vara väldigt lättarbetade för sina syften - nämligen till att tillämpa regler på för att analysera modellens kvalitet. I det här sammanhanget gäller det att undersöka inom en Prolog-instans som representerar en del av KM men också i stor utsträckning titta på samband mellan instanser och se att reglernas krav är uppfyllda. Framförallt involveras flera objektinstanser vid undersökningar om ett objekt har en ärvd identifierare. Det innebär att titta på objekt som är supramängder av andra.

Delmängdsrelationen (isa) i Prolog-representationen visas som en lista av de objekt som är överordnade till ett objekt. Denna relation skiljer sig från relationer som representeras under attribute. De har ingen mapping eller annan bestämning utan återger bara typhierarkin.

Inversattribut är bara intressanta för objekt-objekt relationer. Kanske borde två primitiver "attribute" och "relation" som åtskiljer domänvärda- och objektvärda attribut användas även här. Att som värdeförråd för domäner använda primitiva datatyper är förståndigt. På så vis kan ett välavgränsat antal som nuvarande integer, float, char, string och date finnas till hands. Formatet kan dock inte begränsas speciellt i Prolog-beskrivningen. Om det är viktigt kanske man kan tänka sig subtyper av ovanstående, t ex med maxlängd på string, intervall 1..300 i integer osv.

5. Sammanfattning och slutsatser

5.1 Sammanfattning

Rapporten har beskrivit ett enkelt system för ett naturligtspråkgränssnitt till konceptuell modellering av en sjukhusverksamhet. Verksamheten, företeelser vid ett sjukhus och samband dem emellan, beskrivs genom utsagor på svenska. Systemet genererar från utsagorna via LF (logisk form) en konceptuell modell enligt EMOL (modelleringsansatsen Ericsson MOdelling Language) .

Systemet är interaktivt och löser valsituationer med användares direktiv i en frågebesvarande dialog. Om en större kunskapsbas används så skulle systemet även kunna användas som "batch"-system. I första taget skulle det kräva information som hanterar de valsituationer som uppkommer i detta system. Då behövs information om vilka domänvärda attribut som väntas uppkomma i en konvertering från ett antal satser i NL till KM.

Systemet avser inte att behandla regelkonstruktioner eller utsagor om förekomster inom modellen. Ambitionen har varit att satstyper som kan analyseras korrekt, enligt systemgrammatiken för utsagorna, också skall få utgöra en del av modellen. Grammatiken är begränsad till att beskriva plausibla satser för modellering inom domänen.

5.2 Slutsatser

Målet med systemet var att exemplifiera hur en konceptuell modell kan beskrivas med svenska utsagor om verksamheten. Enligt mitt tycke har detta mål uppfyllts. Med tanke på att systemet är resultatet av tio veckors arbete är begränsningarna för möjliga beskrivningar inom den konceptuella modellens ram förklarliga. Alla satstyper, som har kunnat extraheras som direkt relevanta för en verksamhetsbeskrivning av ett sjukhus och för modellen, kan också konverteras till en del i ett konceptuellt schema.

Modellkonstruktionerna som kan skapas inom systemet begränsas hårt av systemets lexikon som även är ett domänlexikon. För att system som möjliggör design av konceptuella modeller skall vara meningsfyllda krävs att inte en dubbel analys av verksamheten utförs. Om detta är enda sättet

att skapa ett domänlexikon för NL-systemet, så skulle kanske en verksamhetsanalys behöva göras bara för att sedan kunna analysera objektssystemet med naturligt språk.

5.3 Möjliga utökningar

Systemet skulle efter vidareutveckling kunna användas som en del i en utvecklingsmiljö för modellering. Detta tillsammans med kvalitetskontroll för modeller [Wangler 88], grafisk modellrepresentation (RAMATIC) och generering av svenska satser om modellen för validering. I ett längre perspektiv är förhoppningen att personer utan större modelleringskunskaper, som ingår i en verksamhet eller på annat sätt har mycket kunskap om det som skall modelleras, kan använda utvecklingsmiljön.

I det nuvarande systemet kan bara en språklig analys göras på satsnivå. För att interaktionen skall bli smidigare och hela textavsnitt skall kunna behandlas behövs en diskursanalys. Med en sådan analys ovan satsnivå kan sammankopplingar mellan satser analyseras i form av bl a anafor referens (t ex med pronomen) och ellips.

Om grammatiken även hanterar frågesatser kan interaktionen förbättras ytterligare och möjliggöra en enklare dialog.

Den nuvarande grammatiken för satser kan dels utökas för att möjliggöra alternativa uttryckssätt för samma modellbeskrivningar som kan erhållas nu, dels för nya möjliga modeller.

Alternativa uttryckssätt kan fås genom bl a följande konstruktioner:

- "Det finns" ex: *Det finns avdelningar på ett sjukhus.*
- Konjunktioner ex: *På avdelningarna jobbar sköterskor och på avdelningarna jobbar kandidater.*
- Ellips - utelämnade delar ex: *På avdelningar jobbar sköterskor och kandidater.*
- Fler verbformer ex: *Patienter låg på sjukhuset.*

Nya modelltyper kan fås bl a genom att:

- Möjliggöra regelkonstruktioner som förmodligen skulle kräva:

  Adjektiv allmänt ex: *Ett sjukhus är stort om det har 10 000 sängplatser.*

  Komparativa adjektiv ex: *Antalet patienter är större än antalet vårdare.*

  Negationer ex: *Patienter får inte röka på toaletten*

  Modala hjälpverb med tolkning. Nu används de bara för nyanserade konstruktioner (t ex: *Doktorer kan behandla en eller flera patienter)* men med en tolkning av "kan","måste" osv kan regler bli tydliga.

- Beskriva förekomster inom en modell
  För detta krävs förmodligen generell hantering av egennamn

  ex: *Dyrgård som är specialist på Kardiologi jobbar vid avdelningen för hjärtvård.*

- Använda en annan modelleringsansats (för annat än datamodellering) med händelser(event)
  För detta krävs förmodligen

Tidsadverbial	ex *Lönen utbetalas den 26:e i varje månad.*
Olika verbformer och tempus	ex: *Karlsson skrevs in 890820.*

Lexikonet måste utökas i båda fallen ovan. Okända ord som inte förekommer i lexikon kan vara egennamn.

För att inte alla former av substantiv skall vara explicit lagrade i lexikon finns i lexikonet en morfologisk komponent för härledning från olika substantivformer till lexikonlagrad grundform. Den består nu av en procedurell lösning som skall göras om till en deklarativ regelbeskrivning för böjningsändelserna (typ: noun --> nounstem,noun inflection). En liknande beskrivning för verb med tempusändelser som läggs till infinitivstam kan bli aktuell vid utökning av de temporala uttrycksmöjligheterna.

Konverteraren måste utökas i enlighet med nya möjliga språkkonstruktioner och deras logiska form. Nästa systemversions konverterare blir annorlunda, bl a utgår den ifrån en annan logisk form för prepositioner (så en sats huvudfunktor inte kan vara preposition). Även LF för 'isa'-relationen skall omarbetas.

Systemet kan även utökas med ett gränssnitt för att uppdatera lexikon. Det kan vara önskvärt att uppdatera lexikon under modellering. System för naturligtspråkgränssnitt till modellering genererar med fördel ett lexikon simultant med modelleringen. Användarna kan vara mer eller mindre inblandade i denna process beroende på deras språkkunskaper.

Enligt grammatiken kan systemet vid parsning göra antaganden om vilken ordklass/satsdel som ett ord tillhör. I stället för lexikonuppslag som vid konventionell parsing utökas lexikon i detta skede med det påträffade ordet [Rayner 88].

Om ett lexikon har de flesta ord som kan behövas, så skulle användare med tillräckliga språkkunskaper kunna uppdatera lexikon med ett ord, som inte återfinns, enligt ramar för lexikoninformation. Detta innebär en mycket mindre utökning jämfört med den mer automatiserade lexikonuppdateringen ovan. I båda fallen kan det dock kräva en "lexikonansvarig".

Ett NL-gränssnitt för modellering skiljer sig från ett databasgränssnitt i naturligt språk. Viss information om de språkliga uttrycken finns i den redan existerande databasen. Egennamn är just en sådan företeelse som kan bestämmas exakt om det redan finns en databas att söka upp namnet i. Vid modelleringsgränssnitt i naturligt språk finns ingen sådan information att använda sig av; Det är den som skall skapas!

Systemet utgör ett försök till att få utvecklingen av konceptuella modeller enkel genom beskrivningar i naturligt språk. Till syven och sist är detta bara ett redskap för människor som gör jobbet som "modellerare"!

Referenser

Amble T
Stålhane T
Soft systems progress report first year, 1982
SINTEF report

Amble T 87a
Logic Programming and Knowledge Engineering, 1987
Addison-Wesley Publishing Company

Amble T 87b
Knudssen E
Tiihonen H
Tikkanen M
HSQL - En kravspecifikation av ett kunskaps-baserathjälpsystem för SQL,
1987, STATSKONTORET

Black W J
Aquisition of Conceptual Data Models from Natural Language Descriptions
Third conference of the European Chapter of the Association for Computational Linguistics
april 1987,
Köpenhamn, Danmark

Bubenko J
Lindencrona E
Konceptuell Modellering - Informationsanalys, 1984
Studentlitteratur, Lund

Carlsson M
Widén J
SICStus Prolog User's Manual, 1988
Swedish Institute of Computer Science,
SICS R88007, Research Report ISN 0283-3638

Cauvet C,
Proix C,
Rolland C
Information systems design:
An expert system approach,
IFIP working conference
The Role of Artificial Intelligence in Databases and Information Systems, juli,1988
Guangzhou, Kina

Dahl R
Eriksson D
Johansson L-Å
Sundin U
Torbjär H
RAMATIC - a Modeling Support System, 1985
SYSLAB Report Nr 34, aug 1985
SYSLAB, Chalmers Tekniska Högskola,
Göteborg

Dowty D R
Wall R E
Peters S
Introduction to Montague Semantics, 1981
D.Reidel Publishing Company,
Dordrecht, Holland

Ericsson Telecom — EMOL metodhandbok 1988
Ericsson Telecom,
Avdelningen för konstruktionsstöd, system- och programvara.

Kersten M L
Weigand H
Dignum F
Boom J — A conceptual modeling expert system, 1986
5th International Conference on ER-approach
(Ed S Spaccapietra), s 275-288,
Dijon,Frankrike

Lindencrona B
Wangler B — SIMOL - ett språk för konceptuell modellering, 1986, SISU (Svenska Institutet för Systemutveckling), Stockholm

Ljungberg J — Modell över sjukhusdomänen, 1989
Arbetspapper, SISU (Svenska Institutet för System utveckling)

Pereira F C N
Shieber S M — Prolog and Natural-Language Analysis, 1987
Center for the Study of Language and Information,
Lecture Notes nr 10, Stanford, USA

Pereira F C N
Warren D H D — Definite Clause Grammar for language analysis - a survey of the formalism and a comparison with Augmented Transition Networks, 1980
Artificial Intelligence nr 13, s 231-278

Rayner M
Hugosson Å
Hagert G — Using a Logic Grammar to learn a Lexicon,
Swedish Institute of Computer Science,
International Conference on Computational Linguistics, aug 1988, Budapest, Ungern

Teigen J — Syntax analysis of Norwegian Language, 1988
Division of Computer Science,
The University of Trondheim,
The Norwegian Institute of Technology

Wangler B — Kvalitet hos konceptuella schemata, 1988
SISU (Svenska Institutet för systemutveckling),
Stockholm

Wangler B
Wohed R — Diagnosticering av konceptuella schemata, 1988
SISU (Svenska Institutet för systemutveckling),
Stockholm

Warren D H D
Pereira F C N — An efficient Easily Adaptable System for Interpreting Natural Language Queries, 1982
Computational Linguistics, Vol 8 nr 3-4

Winograd T — Computer software for working with language,
1984, Scientific American sep -84

Wohed R — Diagnosis of conceptual schemas, rapport nr 56, 1987, SYSLAB, Institutionen för Data och Systemvetenskap, Kungliga Tekniska Högskolan & Stockholms Universitet,Stockholm

APPENDIX A

NL-beskrivning av en del i en sjukhusverksamhet som skall bilda en KM

1. Ett sjukhus har exakt ett sjukhusnummer.
2. En person har exakt ett personnummer.
3. En avdelning har exakt ett avdelningsnummer.
4. Ett sjukhus har en eller flera avdelningar.

5a. En avdelning består av en eller flera sängplatser.

5b. En eller flera sängplatser tillhör en avdelning.

6. Personal arbetar på en avdelning.
7. Personal är anställd.
8. Anställda är personer.
9. En patient är en person.
10. En patient har exakt en adress.

11a. En doktor behandlar åtminstone en patient på ett sjukhus.

11b. Minst en patient behandlas av en doktor.

12a. En patient behandlas av en eller flera doktorer.

12b. En eller flera doktorer behandlar en patient.

13. På exakt en avdelning ligger en patient.
14. Ingen eller flera patienter ligger på en avdelning.
15. Personal som arbetar på en avdelning är anställd.
16. En läkare som är anställd behandlar en eller flera patienter på ett sjukhus.
17. En specialist är en doktor.
18. Doktorer behandlar patienter som ligger på en avdelning.

Logisk form (LF) för satserna som beskriver en del i en sjukhusverksamhet

1. exists (X, `sjukhus (X) & one (Y,`sjukhusnummer (Y) & `har (X,Y)))

2. exists (X, `person (X) & one (Y,`personnummer (Y) & `har (X,Y)))

3. exists (X, `avdelning(X) & one (Y,`avdelningsnummer (Y) & `har (X,Y)))

4. exists (X, `sjukhus (X) & min_one (Y,`avdelning (Y) & `har (X,Y)))

5a. exists (X, `avdelning (X) & min_one (Y,`sängplats (Y) & `har (X,Y)))

5b. min_one (X, `sängplats(X) & exists(Y,`avdelning (Y) & `har (Y,X)))

6. `på ((X^Z) ^exists (X,`avdelning (X) & Z),
exists (Y,`personal(Y) & `arbetar(Y)))

7. exists (X, `personal (X) & exists (Y,`anställd (Y) & `isa (X,Y)))

8. exists (X, `anställd (X) & exists (Y,`person (Y) & `isa (X,Y)))

9. exists (X, `patient (X) & exists (Y,`person (Y) & `isa (X,Y)))

10. exists (X, `patient (X) & one (Y,`adress (Y) & `har (X,Y)))

11a. `på ((X^Z) ^exists (X,`sjukhus(X) & Z),
exists (Y,
`doktor (Y) & min_one (W,`patient(W) & `behandlar (Y,W))))

11b. min_one (X,`patient (X) & exists (Y,`doktor (Y) & `behandlar (Y,X)))

12a. exists (X,`patient(X) & min_one (Y, `doktor (Y) & `behandlar (Y,X)))

12b. min_one (Y,`doktor(Y) & exists (X,`patient (X) & `behandlar (Y,X)))

13. `på ((X^Z) ^one (X,`avdelning (X) & Z),
exists (Y,`patient (Y) & `ligger (Y)))

14. `på ((X^Z) ^exists (X,`avdelning (X) & Z),
any_num (Y,`patient (Y) & `ligger (Y)))

15. exists (X,
(`personal (X) & `på ((Y^S) ^exists (Y,`avdelning (Y) & S),
`arbetar (X)))
&
exists (Z,`anställd (Z) & `isa (X,Z)))

16. `på ((X^Z) ^exists (X,`sjukhus(X) & Z),
exists (Y,
(`doktor (Y) & exists (W,`anställd(W) & `isa (Y,W)))
&
min_one(S,`patient (S) & `behandlar (Y,S))))

17. exists (X,`specialist (X) & exists (Y,`doktor (Y) & `isa (X,Y)))

18a. exists (X, `doktor (X) & exists (Y, `patient (Y)
&
`på ((Z^W)^exists (Z,`avdelning (Z) & W),
`ligger (Y)))
&
`behandlar (X,Y)))

18b. `på ((Z^W)^exists (Z,`avdelning (Z) & W),
exists (X, `doktor (X) & exists (Y, `patient (Y & `ligger (Y))
&
`behandlar (X,Y))))