Structured descriptions

The syntax of FOL makes it easy to say things about objects. Frames organize knowledge in terms of categories of objects. Description logics are notations that are designed to make it easier to describe definitions and properties of categories, by adding structure to the definition of objects. The focus is on declarative aspects of object-oriented representation, going back to concepts like predicates and entailment from FOL.
Description logic systems evolved from frames/semantic networks by formalizing what the networks mean, while keeping the emphasis on taxonomic structure as an organizing principle (that helps in organizing a hierarchy of categories).
The principal inference tasks for description logics are subsumption (checking if a category is a subset of another by comparing their definitions) and satisfaction (checking whether an object belongs to a category).

In standard FOL systems, predicting the solution time is often impossible. In description logics, the subsumption testing can be solved in time polynomial in the size of the description. But (hard) problems either cannot be stated at all in description logics, or they require exponentially large descriptions.
In FOL, we represent categories of objects with simple predicates like Mother(x), Boat(x), Company(x). To represent more interesting types of constructions like "a man whose children are all girls", we need predicates with internal structure. We would expect that if Child(x, y) and FatherOfOnlyGirls(x) were true, then y would have to be a girl (somehow) by definition.

We have category nouns like FatherOfOnlyGirls, Girl describing basic classes of objects and relational nouns like Child that are parts/attributes/properties of other objects.
In description logics, we refer to the first type as a <u>concept</u> and to the second type as a <u>role</u> (in frame systems we saw a similar distinction between frames/slots).
In contrast to the slots in frame systems, roles can have multiple fillers. Thus, it can be described naturally a person with several children, a salad made from more than one type of vegetable.
Although much of the reasoning in description logics concerns generic categories, constants are included to allow for descriptions to be applied to individuals.

A description language

In a description language (DL) there are two types of symbols:

- logical symbols, with a fixed meaning
- nonlogical symbols, which are application dependent

There are four types of logical symbols:

- punctuation: [,], (,)
- positive integers: 1, 2, 3 ...
- concept-forming operators: ALL, EXISTS, FILLS, AND
- connectives: $\sqsubseteq$, $\doteq$, $\rightarrow$

There are three types of nonlogical symbols:

- atomic concepts – the name starts with upper-case
  Person, FatherOfOnlyGirls
  Thing – a special atomic concept
- roles – the name starts with upper-case, prefixed by:
  :Age, :Child
- constants – the name starts with lower-case
  table17, johnSmith

There are four types of legal syntactic expressions:

- constants c
- roles r
- concepts d, e ; atomic concepts a
- sentences

The set of concepts of DL satisfies the following:

- every atomic concept is a concept;
- if r is a role, d is a concept then [ALL r d] is a concept;
- if r is a role, $n \in \mathbb{N}^*$ then [EXISTS n r] is a concept;
- if r is a role, c is a constant then [FILLS r c] is a concept;
- if $d_1, \ldots, d_n$ are concepts then [AND $d_1 \ldots d_n$] is a concept.

There are three types of sentences in DL:

- if d_1, d_2 are concepts then $(d_1 \sqsubseteq d_2)$ is a sentence;
- if d_1, d_2 are concepts then $(d_1 \doteq d_2)$ is a sentence;
- if c is a constant and d concept then $(c \rightarrow d)$ is a sentence.

A Knowledge base KB in DL is a collection of sentences.

Constants represent individuals in the application domain; concepts represent categories or classes of individuals; and roles represent binary relations between individuals.

The meaning of a complex concept derives from the meaning of its parts.

For example, [EXISTS n r] represent the class of individuals in the domain that are related by relation r to at least n other individuals. [EXISTS 1 :Child] represents someone who has at least one child.

if c is a constant that stands for some individual, the concept [FILLS r c] represents those individuals that are in relation r with c. [FILLS :Cousin george] represent someone whose cousin is George.

if concept d represents a class of individuals, [ALL r d] represent individuals who are in relation r only to individuals of class d. [ALL :Employee UnionMember] describes companies whose employees are all union members.

The concept [AND $d_1 \ldots d_n$] represents anything described by d_1 and ... d_n.

```
[AND Wine
     [FILLS :Color red]
     [EXISTS 2 :GrapeType]
]
```

```
(ProgressiveCompany ≐ [AND Company
                          [EXISTS 7 : Director]
                          [ALL :Manager [AND Women
                                             [FILLS :Degree phd]]]
                          [FILLS :MinSalary $5000]
                      ])
```

In DL, sentences are true or false in the domain (like in FOL).

d_1, d_2 concepts and c constant

$(d_1 \sqsubseteq d_2)$ says that d_1 is subsumed by d_2, that is all individuals that satisfy d_1 also satisfy d_2.

(Surgeon $\sqsubseteq$ Doctor)

$(d_1 \doteq d_2)$ says that d_1 and d_2 are equivalent, that is the individuals that satisfy d_1 are exactly those that satisfy d_2. It is the same as saying that both $(d_1 \sqsubseteq d_2)$ and $(d_2 \sqsubseteq d_1)$ are true.

$(c \rightarrow d)$ says that the individual denoted by c satisfies the description expressed by d.

Interpretations in $\mathcal{DL}$

An interpretation $\mathfrak{I}$ is a pair $\langle \mathcal{D}, \mathcal{I} \rangle$, where $\mathcal{D}$ is a non-empty set of objects called the domain of the interpretation and $\mathcal{I}$ is the interpretation mapping that assigns a meaning to the nonlogical symbols of $\mathcal{DL}$, so that:

1. for every constant c, $\mathcal{I}[c] \in \mathcal{D}$;
2. for every atomic concept a, $\mathcal{I}[a] \subseteq \mathcal{D}$;
3. for every role r, $\mathcal{I}[r] \subseteq \mathcal{D} \times \mathcal{D}$.

The set $\mathcal{I}[d]$ is called the extension of the concept d:

- $\mathcal{I}[\text{Thing}] = \mathcal{D}$;
- $\mathcal{I}[[\text{ALL } r\ d]] = \{x \in \mathcal{D} \mid \forall y \text{ if } \langle x, y \rangle \in \mathcal{I}[r] \text{ then } y \in \mathcal{I}[d]\}$;
- $\mathcal{I}[[\text{EXISTS } n\ r]] = \{x \in \mathcal{D} \mid \text{there are at least } n \text{ distinct } y \text{ such that } \langle x, y \rangle \in \mathcal{I}[r]\}$;
- $\mathcal{I}[[\text{FILLS } r\ c]] = \{x \in \mathcal{D} \mid \langle x, \mathcal{I}[c] \rangle \in \mathcal{I}[r]\}$;
- $\mathcal{I}[[\text{AND } d_1 \dots d_n]] = \mathcal{I}[d_1] \cap \dots \cap \mathcal{I}[d_n]$.

Truth in an interpretation

The sentence $(c \rightarrow d)$ is true in $\mathfrak{I}$ if the object denoted by c is in the extension of d – $\mathcal{I}[c] \in \mathcal{I}[d]$.

The sentence $(d \sqsubseteq d')$ is true if the extension of d is a subset of the extension of d' – $\mathcal{I}[d] \subseteq \mathcal{I}[d']$.

The sentence $(d \doteq d')$ is true if $\mathcal{I}[d] = \mathcal{I}[d']$

If a sentence α is true in $\mathfrak{I}$, we write $\mathfrak{I} \models \alpha$.
If S is a set of sentences, we will write $\mathfrak{I} \models S$ to say that all the sentences in S are true in $\mathfrak{I}$.

Entailment

Let S be a set of sentences in DL and α a sentence.

S logically entails α, and we write $S \models \alpha$, iff for every interpretation $\mathfrak{I}$, if $\mathfrak{I} \models S$ then $\mathfrak{I} \models \alpha$.

A sentence α is logically valid, and we write $\models \alpha$, if it is logically entailed by the empty set.

In DL, there are two basic types of reasoning: determining whether or not a constant c satisfies a concept d; and determining whether or not a concept d is subsumed by another concept d'.

$$KB \models (c \rightarrow d)$$
$$KB \models (d \sqsubseteq d')$$

Examples of valid sentences:

([AND Doctor Female] $\sqsubseteq$ Doctor)

(john $\longrightarrow$ Thing).

In more typical cases, the entailment depends on sentences in the KB. For example, if KB contains the sentence (Surgeon $\sqsubseteq$ Doctor), then we can logically entail that

KB $\models$ ([AND Surgeon Female] $\sqsubseteq$ Doctor).

We can reach the same conclusion if we have in KB the sentence (Surgeon $\doteq$ [AND Doctor [FILLS :Specialty surgery]]) instead of (Surgeon $\sqsubseteq$ Doctor).

But with the empty KB, we would have no subsumption relation ([AND Surgeon Female] $\sqsubseteq$ Doctor) because we can choose an interpretation $\mathfrak{I}$ in which the sentence is false (for example, I[Doctor] = $\emptyset$ and I[Surgeon] = I[Female] = {1,2,3}).

Computing entailments

Given a KB, we want to determine if $KB \models \alpha$ for α of the form:

$(c \rightarrow d)$ where c constant and d concept

$(d \sqsubseteq e)$ where d, e concepts

[$KB \models (d \doteq e)$ iff $KB \models (d \sqsubseteq e)$ and $KB \models (e \sqsubseteq d)$].

Simplifying the KB

Obs. It can be proven that subsumption entailments are not affected by the presence of sentences $(c \rightarrow d)$ in KB. That is to say that $KB \models (d \sqsubseteq e)$ iff $KB' \models (d \sqsubseteq e)$, $KB' = KB - \{$all sentences $(c \rightarrow d)\}$

For subsumption questions, we assume that the KB contains no $(c \rightarrow d)$ sentences.

Moreover, we can replace sentences of the form $(d \sqsubseteq e)$ by $(d \doteq$ [AND e a]), where a is a new atomic concept used nowhere else.

We will consider the following restrictions in the KB:

- the left-hand sides of $\doteq$ is an atomic concept other than Thing
- each atom appears on the left-hand side of $\doteq$ exactly once in KB - such sentences provide definitions of the atomic concepts

  (RedBordeauxWine $\doteq$ [AND Wine
  [FILLS :Color red]
  [FILLS :Region bordeaux]])
- we assume that sentences $\doteq$ in KB are acyclic. We rule out a KB that contains

  ($d_1 \doteq$ [AND d_2 ...]), ($d_2 \doteq$ [ALL r d_3]), ($d_3 =$ [AND d_1 ...]), Under these restrictions, to determine if KB $\models$ (d $\sqsubseteq$ e) we do the following:

1. put d and e into a special normalized form
2. determine whether each part of the normalized e is accounted for by some part of the normalized d. We are looking for a structural relation between two normalized concepts. For example, if e contains [ALL r e'] then d must contain [ALL r d'] with d' $\sqsubseteq$ e'.

Normalization

It is a preprocessing that simplifies the structure-matching between concepts. It applies to one concept at a time and involves the following steps:

1. expand definitions — any atomic concept in the left-hand side of $\doteq$ is replaced by its definition.

   Example: assume that we have the following sentence in KB

   (Surgeon $\doteq$ [AND Doctor [FILLS :Specialty surgery]]).

   The concept [AND ... Surgeon ...] expands to

   [AND ... [AND Doctor [FILLS : Specialty surgery]] ...]

2. flatten the AND operators

   [AND ... [AND d_1 ... d_n] ...] becomes [AND ... d_1 ... d_n ...]

3. combine the ALL operators

   [AND ... [ALL r d_1] ... [ALL r d_2] ...] becomes

   [AND ... [ALL r [AND d_1 d_2]] ...]

4. combine the EXISTS operators

   [AND ... [EXISTS n_1 r] ... [EXISTS n_2 r] ...] becomes

   [AND ... [EXISTS n r] ...], where $n = \max(n_1, n_2)$.

5. Thing concept – remove Thing, [ALL r Thing] and AND with no arguments if they appear as arguments in an AND concept

[AND ... Thing ...] becomes [AND ...]

[AND Company [ALL :Employee Thing]] becomes Company.

6. remove redundant expressions – eliminate duplicates within the same AND expression.

These six steps are applied repeatedly until no steps are applicable. The result is either Thing, an atomic concept or a concept of the following form:

[AND $a_1 \ldots a_m$

[FILLS r_1 c_1] ... [FILLS $r_{m'}$ $c_{m'}$]

[EXISTS n_1 s_1] ... [EXISTS $n_{m''}$ $s_{m''}$]

[ALL t_1 e_1] ... [ALL $t_{m'''}$ $e_{m'''}$]]

where $a_1, \ldots, a_m$ are atomic concepts (other than Thing), r_i, s_i, t_i are roles, c_i are constants, n_i are positive integers and e_i are normalized concepts.

Example 1 - We have the following KB:

WellRoundedCo ≐ [AND Company
[ALL :Manager [AND B-SchoolGrad
[EXISTS 1 :TechnicalDegree]
]]]

HighTechCo ≐ [AND Company
[FILLS :Exchange nasdaq]
[ALL :Manager Techie]]

Techie ≐ [EXISTS 2 :TechnicalDegree]

Normalize the concept [AND WellRoundedCo HighTechCo]

$\xRightarrow{1}$ [AND [AND Company [ALL :Manager [AND B-SchoolGrad
[EXISTS 1 :TechnicalDegree]
]]]
2
3
[AND Company
[FILLS :Exchange nasdaq]
[ALL :Manager [EXISTS 2 : TechnicalDegree]]
]
]

$\xRightarrow{2,3}$ [AND Company [ALL :Manager [AND [AND B-SchoolGrad
2
[EXISTS 1 :Technical Degree]]
4
[EXISTS 2 : TechnicalDegree]
6
]
]
Company
[FILLS :Exchange nasdaq]]

$\xRightarrow{2,4,6}$ [AND Company
[ALL :Manager [AND B-SchoolGrad
[EXISTS 2 :TechnicalDegree]]]
[FILLS :Exchange nasdaq]]

<u>Structure matching procedure – Subsumption computation</u>

<u>Input</u>: d and e two normalized concepts

d is [AND $d_1 \ldots d_m$]

e is [AND $e_1 \ldots e_{m'}$]

<u>Output</u>: YES or NO according to whether or not $KB \models (d \sqsubseteq e)$

Return YES iff for each e_j, $j \in \overline{1,m'}$, there exists a component d_i, $i \in \overline{1,m}$ such that d_i matches e_j as follows:

1. if e_j is an atomic concept, then d_i must be identical to e_j;
2. if e_j is of the form [FILLS r c], then d_i must be identical to it;
3. if e_j is of the form [EXISTS n r], then d_i must be of the form [EXISTS n' r] for some $n' \geq n$; in the case where $n=1$, d_i can also be of the form [FILLS r c], for any constant c;
4. if e_j is of the form [ALL r e'], then d_i must be of the form [ALL r d'], where recursively $d' \sqsubseteq e'$.

Example 2

```
d      [AND Company
            [ALL :Manager B-SchoolGrad]
          1 [EXISTS 1 :Exchange]]
                                                        4
d'     [AND Company
          [ALL :Manager [AND B-SchoolGrad
                              [EXISTS 2 :TechnicalDegree]]]
       3  [FILLS :Exchange nasdaq]]
```

So $d' \sqsubseteq d$.

Computing satisfaction

We are interested whether $KB \models (b \rightarrow e)$, where b is a constant and e is a concept.

To find out if an individual satisfies a description, we need to propagate the information implied by what we know about other individuals before checking for subsumption. This can be done by a forward chaining procedure.

In the case where there are no EXISTS terms in any concept, the procedure is as following:

1. Construct S a list of pairs (b, d), where b is any constant mentioned in KB and d is the normalized version of the concept [AND $d_1' \dots d_n'$] for all d_i' such that $(b \rightarrow d_i') \in KB$.
2. Find two constants b_1 and b_2 such that $(b_1, d_1) \in S$ and $(b_2, d_2) \in S$, [FILLS r b_2] and [ALL r e] are both components of d_1 but $KB \not\models (d_2 \sqsubseteq e)$.
3. If no b_1 and b_2 can be found, then exit. Otherwise, replace the pair (b_2, d_2) in S by (b_2, d_2'), where d_2' is the normalized version of [AND d_2 e] and go to step 2.

The procedure computes for each constant b the most specific concept d such that $KB \models (b \to d)$. Now, to test whether or not $KB \models (b \to e)$, we need only to test whether or not $KB \models (d \sqsubseteq e)$.

<u>Example 3</u>

$$KB \begin{cases} \text{joe} \to \text{Person} \\ \text{conCorp} \to \underbrace{[\text{AND Company } [\text{ALL } \overset{r}{\text{:Manager}}\ \overset{e}{\text{Canadian}}]\ [\text{FILLS } \underset{r}{\text{:Manager}}\ \underset{b_2}{\text{joe}}]]}_{D} \end{cases}$$

Question $KB \models (\text{joe} \to \text{Canadian})$

$$S = \{ (\underset{b_2}{\text{joe}}, \underset{d_2}{\text{Person}}), (\underset{b_1}{\text{conCorp}}, \underset{d_1}{D}) \} \qquad KB \not\models (\text{Person} \sqsubseteq \text{Canadian})$$

$\Rightarrow S = \{ (\text{joe}, [\text{AND Person Canadian}]), (\text{conCorp}, D) \}$. Now the procedure terminates because $KB \models ([\text{AND Person Canadian}] \sqsubseteq \text{Canadian})$.

Because $KB \models ([\text{AND Person Canadian}] \sqsubseteq \text{Canadian})$, it follows that $KB \models (\text{joe} \to \text{Canadian})$.

in the case where there are EXISTS terms of the form [EXISTS 1 r], we will use role chains

$$[\text{AND} \ldots [\text{ALL } r_1 \ldots [\text{AND} \ldots [\text{ALL } r_k\ a] \ldots] \ldots] \ldots]$$

$\tau = r_1 \cdot r_2 \cdots r_k$ is called a role chain.

if b is a constant and r_1, r_2 roles, then $b \cdot r_1 \cdot r_2$ represents an individual (perhaps unnamed) that is in relation r_2 with an individual that is in relation r_1 with b.

if τ is empty, then $b \cdot \tau$ is b.

The forward chaining procedure extends, by adding two additional steps:

(the previous steps at page 11)

4. Find a constant b, a role chain τ (possibly empty) and a role r such that $(b \cdot \tau, d_1) \in S$ and $(b \cdot \tau \cdot r, d_2) \in S$ (if no such pair exists, take d_2 to be Thing), where [EXISTS 1 r] and [ALL r e] are components of d_1, but $KB \not\models (d_2 \sqsubseteq e)$.
5. If these can be found, remove $(b \cdot \tau \cdot r, d_2)$ from S (if applicable) and add the pair $(b \cdot \tau \cdot r, d_2')$, where d_2' is the normalized version of [AND d_2 e]. Repeat.

We start with a property of the individual $b \cdot \tau$ and conclude something new about the (unnamed) individual $b \cdot \tau \cdot r$. Eventually, this can lead to new information about a named individual.

Example 4 Assume that in KB we have the sentence

$b \cdot \tau$ τ is empty

$$\text{ellen} \longrightarrow \underbrace{[\text{AND } [\text{EXISTS } 1 \ \overset{r}{(:\text{Child})}] \ [\text{ALL } \underset{r}{(:\text{Child})} \ \underbrace{[\text{AND } [\text{FILLS } :\text{Pedititician marianne}] \ [\text{ALL } :\text{Pedititician Scandinavian}]]}_{e} \ ] \ ]}_{d_1}$$

$S = \{(b \cdot \tau, d_1)\}$ and $(b \cdot \tau \cdot r, d_2) = (\text{ellen} : \text{Child}, \text{Thing})$

Because $KB \not\models (\text{Thing} \sqsubseteq e)$, S becomes

$$S = \{(b \cdot \tau, d_1), (\underset{b \cdot \tau \cdot r}{\text{ellen}:\text{Child}}, [\text{AND } [\text{FILLS } :\text{Pedititician marianne}] \ [\text{ALL } :\text{Pedititician Scandinavian}]])\}$$

From here, we conclude that (marianne $\rightarrow$ Scandinavian) (case with no EXISTS)

The case of terms of the form [EXISTS n r], $n > 1$ is handled the same as for $n = 1$. There is no need to create n different anonymous individuals because all of them would "produce" the same properties in the forward chaining.

Taxonomies and classification

Given a concept q, in DL it is common to ask for all of its instances, that is to find all c in KB so that $KB \models (c \to q)$.
Also, it is common to ask for all of the known categories that an individual satisfies. That is to say that given a constant c, we should find all concept a so that $KB \models (c \to a)$.

When reasoning in DL, we should exploit the hierarchical organization of the concepts, with the most general ones at the top and the more specialized ones further down.
To represent sentences in KB, we use a taxonomy (a treelike data structure) that allows answering queries efficiently (time linear with the depth of the taxonomy, not with its size).

Obs. Subsumption is a partial order.

The taxonomy have atomic concepts as nodes and edges like $\uparrow^{a_j}_{a_i}$ whenever $a_i \sqsubseteq a_j$ and there is no a_k such that $a_i \sqsubseteq a_k \sqsubseteq a_j$.

Each constant c in KB will be linked to the most specific atomic concept a_i such that $KB \models (c \to a_i)$.

Adding some new atomic concept or constant to a taxonomy corresponding to a KB is called classification. It involves creating a link from the new concept or constant to existing ones in the taxonomy.
This process exploits the structure of the taxonomy.
We start with the concept Thing and then add incrementally new atomic concepts and constants.

Computing classification

I add a sentence ($a_{new} \doteq d$) to the taxonomy, where a_{new} is an atomic concept not appearing anywhere in the KB and d is any concept:

1. Compute S, the most specific subsumers of d

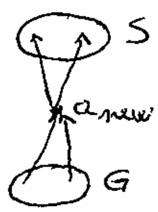
$$S = \{ a\text{-concept in the taxonomy} \mid KB \models (d \sqsubseteq a), \text{ but } \nexists\, a' \neq a \text{ so that } KB \models (d \sqsubseteq a') \text{ and } KB \models (a' \sqsubseteq a) \}$$

2. Compute G, the most general subsumees of d

$$G = \{ a\text{-concept in the taxonomy} \mid KB \models (a \sqsubseteq d), \text{ but } \nexists\, a' \neq a \text{ so that } KB \models (a' \sqsubseteq d) \text{ and } KB \models (a \sqsubseteq a') \}$$

3. if $\exists\, a \in S \cap G$ then a_{new} is already in the taxonomy under a different name – no action needed

4. Otherwise remove all links (if any) from concepts in G up to concepts in S.



5. Add links from a_{new} up to each concept in S and links from each concept in G up to a_{new}

6. Handling constants

Compute $C = \{ c\text{ - constant in taxonomy} \mid \forall\, a \in S,\ KB \models (c \to a) \text{ and } \nexists\, a' \in G \text{ such that } KB \models (c \to a') \}$

Then for each $c \in C$ we test if $KB \models (c \to d)$ and if so, we remove the links from c to S and add a single link from c to a_{new}.

II add a sentence ($a_{new} \sqsubseteq d$) reduces to adding links from a_{new} to the most specific subsumers of d.

III add a sentence ($c_{new} \to d$) reduces to adding links from c_{new} to the most specific subsumers of d.

Compute S - the most specific subsumers of d

start with $S = \{Thing\}$

for all $a \in S$ if $\exists a'$ so that $\begin{array}{c} a \\ \uparrow \\ a' \end{array}$ and $KB \models (d \sqsubseteq a')$ then

remove a from S and add all a' in S.

Repeat until no element in S has a child that subsumes d.

Compute G – the most general subsumees of d

start with $G = S$

if $\exists a \in G$ so that $KB \not\models (a \sqsubseteq d)$, then replace a with all its children (or delete it if it has no children).

Repeat until each element in G is subsumed by d.

Finally, we delete each $a \in G$ that has a parent subsumed by d.

Answering questions in DL

To find all constants c that satisfy a concept q, we should classify q and then collect all the constants at the fringe of the tree below q in the taxonomy.

To find all atomic concepts that are satisfied by a constant c, we go from c up in the taxonomy, collecting all the nodes that can be reached.

KB
$$\left[\begin{array}{l} \text{Toddler} \\ \text{Toddler} \supset \text{Child} \\ \text{Child} \wedge \text{Male} \supset \text{Boy} \\ \text{infant} \supset \text{Child} \\ \text{Child} \wedge \text{Female} \supset \text{Girl} \\ \text{Female} \end{array}\right.$$

Question: Girl

KB in DL
$$\left[\begin{array}{l} (\text{Toddler} \sqsubseteq \text{Child}) \\ (\text{CM} \doteq [\text{AND Child Male}]) \\ (\text{infant} \sqsubseteq \text{Child}) \\ (\text{CF} \doteq [\text{AND Child Female}]) \\ (\text{CM} \sqsubseteq \text{Boy}) \\ (\text{CF} \sqsubseteq \text{Girl}) \\ (\text{ana} \rightarrow \text{Toddler}) \\ (\text{ana} \rightarrow \text{Female}) \end{array}\right.$$

Question: $(\text{ana} \rightarrow \text{Girl})$