A FORMAL BASIS FOR IMPLICIT NAVIGATION
IN ENTITY-RELATIONSHIP QUERY LANGUAGES

by

Peter Kandzia
and
Rainer Manthey

Institut für Informatik und
Praktische Mathematik
Universität Kiel

Abstract
Main issue is to reduce extent and complexity of queries expressed in a calculus-like database query language for the entity-relationship model. The reduction is performed by syntactical constructs added to the calculus which allow to infer the non-reduced calculus version of a query by inspection of the underlying database scheme. A hierarchy of five calculus-based formal query languages is introduced where each level provides more powerful possibilities of reduction than the lower ones. The semantics of the languages are formally defined. The highest-level language has the additional property that some constructs can be replaced by natural-language structures, such that the semantics can immediately be recognized by the user.
Introduction

In database literature several contributions are dealing with attempts to reduce extent and complexity of queries expressed in a database query language, e.g., [C/K 76], [Osb 79], [M/U 80], [B/K 81]. The object of these papers is to develop query interpreting systems which require from the user a minimal amount of information necessary to answer queries correctly and unambiguously.

While certain parts of a query are essential (e.g., attribute value specifications), others are dispensable if some standard interpretation is assumed under which it is possible to infer the missing parts from the rest of the query. This holds especially for the description of semantic connections between the entities addressed in a query.

In the area of relational databases query languages have been proposed - among others by Maier and Ullman based on the notion of maximal objects - that do not make use of the underlying relations but express queries simply in terms of attributes. The semantic connections between the attributes are assumed to be reproduced by some standard join of the relations to which they belong and - possibly - some intermediate relations necessary to guarantee losslessness of the join.

The present paper provides query languages that take from the user the burden of explicitly stating every semantic connection, too. It is, however, not based on the relational model, but is concerned with the entity-relationship model, as the elements of the latter are more closely related to the terms required for query formulation in natural language. While most authors are concerned with interpretational conventions for queries expressed in a given syntax (e.g., QUEL without range-statements, or SQL without FROM-clause) we develop syntactical tools to reduce query complexity.

We assume that specification of certain semantic connections (corresponding to the composition of relationships in our ER-environment) is dispensable and can be algorithmically inferred and that existential quantification of "intermediate" variables is provided when composed relationships are reconstructed. Some of our ideas are related to those in [B/K 81].
We present a formal basis for the definition of syntax and semantics of ER-query languages which support the implicit semantical navigation described above.

We are dealing with a modified version of the ER-model where the roles of entities in relationships are explicitly stated, and we use an applied calculus for this model. Both, modified ER-model and applied calculus, are formally defined in section 1. In sections 2 and 3 a hierarchy of five calculus-based formal query languages is defined where each level provides more powerful possibilities of implicit navigation than the lower ones. The syntax of each of these languages is defined by means of terms and formulas of the calculus augmented by an increasing amount of "calculus-like" syntactical features able to abbreviate more complex expressions.

The semantics of the higher-level languages are defined by transformation rules taking queries stepwise back to the basic-level language which itself can be interpreted by means of the calculus semantics.

Queries that are not meaningful (i.e., the semantics of which are undefined in a given environment) are characterized by two sets of "contextual conditions" added to the syntax, one being independent (e.g., scope conditions), the other being dependent on the underlying database scheme (e.g., type checking). The complexity of these conditions is increasing with the level of the resp. language. A similar method has been used for the definition of block-structured programming languages.

A special characteristic of our approach is that on each level the methods of implicit navigation can equally be applied to queries with and without universal quantifiers. Furthermore, no restriction to the class of database states to which it is applicable is necessary. Both topics cause problems in the area of universal relation query languages, e.g..

Finally, in section 4 we propose several features of a query language which is able to express the constructs formally developed in this paper by means English-(natural language-)like syntactical structures. There has been a great amount of mutual influence between the formal model and the individual features of this language.
1. The entity-relationship model

1.1 Definition of database schemes and states

An entity-relationship database scheme \( \mathcal{S} \) is defined by means of the following five components:

- a collection of entity type schemes \( \{(E_1, \xi_1), \ldots, (E_m, \xi_m)\} \)
  where \( E_1, \ldots, E_m \) denote entity types and \( \xi_1, \ldots, \xi_m \) are sets of attributes

- a collection of relationship type schemes \( \{(R_1, \eta_1), \ldots, (R_n, \eta_n)\} \)
  where \( R_1, \ldots, R_n \) denote relationship types and \( \eta_1, \ldots, \eta_n \) are sets of roles

- a collection of (not necessarily disjoint) value domains \( \{V_1, \ldots, V_s\} \), each \( V_i \) being a finite non-empty set of character strings (i.e., printable objects)

- a non-empty internal identifier domain \( \text{IDF} \) which is disjoint of each value domain

- a function \( \text{dom} \) assigning a value domain to each attribute

- a function \( \text{role} \) assigning an entity type to each role

(i.e., we are not dealing with multi-valued attributes in this paper, nor do we allow attributes of relationships)

The respective collections of schemes are satisfying the uniqueness conditions

\[
E_i \neq E_j \quad \text{and} \quad \xi_i \cap \xi_j = \emptyset \quad \text{for all} \quad i, j \in \{1, \ldots, m\}, i \neq j
\]

\[
R_i \neq R_j \quad \text{and} \quad \eta_i \cap \eta_j = \emptyset \quad \text{for all} \quad i, j \in \{1, \ldots, n\}, i \neq j.
\]

Moreover, the sets \( \text{Attr}, \text{Ent}_f, \text{Rel}_f \) of attributes, entity types, roles and relationship types, resp., are mutually disjoint.

Thus, there is a surjective function \( \text{poss} \) describing the assignments expressed by the schemes:

\[
\text{poss}: \text{Attr} \cup \text{Rel} \rightarrow \text{Ent}_f \cup \text{Rel}_f
\]

\[
\text{poss}(A) := E_i \quad \text{for} \quad A \in \xi_i, \quad i \in \{1, \ldots, m\}
\]

\[
\text{poss}(P) := R_j \quad \text{for} \quad P \in \eta_j, \quad j \in \{1, \ldots, n\}
\]

\( E_i \) is the possessor of attribute \( A \), \( R_j \) that of role \( P \).

An entity-relationship database scheme \( \mathcal{S} \) may be graphically represented by means of an ER-diagram in the following way:

- Each entity type \( E_i \) is repr. by a labelled node \( E_i \)

- Each relationship type \( R_j \) is repr. by a labelled node \( R_j \)
- Each value domain \( V_k \) is repr. by a labelled node \( V_k \).
- The functions dom, role and poss are repr. by labelled edges whenever
  \[
  E_i \xrightarrow{A} V_k \quad \text{or} \quad R_i \xrightarrow{P} E_i
  \]
  or \( \text{poss}(A) = E_i \) and \( \text{dom}(A) = V_k \)
  or \( \text{poss}(P) = R_j \) and \( \text{role}(P) = E_i \), resp., holds.

An ER-diagram may contain arbitrary cycles.

An entity-relationship database state \( S \) for a given scheme consists of three components:

- a collection of \( m \) finite, non-empty sets of internal identifiers, one for each entity type of \( \mathcal{Y} \):
  \[
  \{ \text{ext}_S(E_1), \ldots, \text{ext}_S(E_m) \} \quad \text{with} \quad \emptyset \neq \text{ext}_S(E_i) \subseteq \text{IDF}
  \]
- a collection of \( n \) finite, non-empty sets of relationships, one for each relationship type of \( \mathcal{Y} \):
  \[
  \{ \text{ext}_S(R_1), \ldots, \text{ext}_S(R_n) \} \quad \text{with} \quad \emptyset \neq \text{ext}_S(R_i) \subseteq \left\{ \nu \mid \nu: \mathcal{Y}_i \rightarrow \bigcup_{P \in \mathcal{Y}_i} \text{ext}_S(\text{role}(P))) \right\}
  \]
  i.e., each relationship \( \nu \) in \( \text{ext}_S(R_i) \) assigns an element of the extension of entity type \( \text{role}(P) \) to each \( P \) in \( \mathcal{Y}_i \).
- a collection of functions, one for each attribute \( A \) of \( \mathcal{Y} \):
  \[
  \{ A_S: \text{ext}_S(\text{poss}(A)) \rightarrow \text{dom}(A) \} \quad A \in \mathcal{A}_\mathcal{Y}
  \]

As there are no integrity constraints, each triple of collections with the above properties is a database state for \( \mathcal{Y} \).

Fig. 1 shows the ER-diagram of an example scheme \( \mathcal{Y}_1 \) which will frequently be referenced in the remainder of this paper.

Fig. 2 outlines a possible representation of a state for \( \mathcal{Y}_1 \) by means of a relational database state according to Codd's extended relational model RM/T ( [Cod 79] ).

Columns containing elements from the internal identifier domain IDF - called entity domain in [Cod 79] - are marked by \( \rule{1cm}{0.15mm} \) .
Fig. 1

(combined) entity-/property relations:

<table>
<thead>
<tr>
<th>EMPL</th>
<th>DEPT</th>
<th>PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE NAME</td>
<td>EMPLOYEE #</td>
<td>PART TYPE</td>
</tr>
<tr>
<td>AGE</td>
<td>FLOOR</td>
<td>COLOR</td>
</tr>
<tr>
<td>SALARY</td>
<td>BUDGET</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COMP.</th>
<th>association relations:</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP. TO NAME</td>
<td>MANAGEMENT</td>
</tr>
<tr>
<td>CITY</td>
<td>MANAGER</td>
</tr>
<tr>
<td></td>
<td>MEMBER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUPPLY</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDERER</td>
<td>SELLER</td>
</tr>
<tr>
<td>SUPPLIER</td>
<td>SOLD ITEM</td>
</tr>
<tr>
<td>ORDERED ITEM</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2
1.2 A first-order language \( \mathcal{L}_R \) for a given scheme \( R \)

**Syntax of \( \mathcal{L}_R \):**

a) **logical symbols:**
   - Boolean connectives: \( \neg, \land, \lor, \Rightarrow \)
   - quantifiers: \( \exists, \forall \)
   - parentheses: (,), [],
   - delimiters: .:, st (denoting "such that")
   - variables: \( x,y,z,u,v,w,x_1,x_2, \ldots \)

b) **non-logical symbols:**
   - a constant symbol 'v' for each value v in some value domain \( V_1 \) of \( R \)
   - a unary function symbol A for each attribute A of \( R \)
   - a unary predicate symbol E for each entity type E of \( R \)
   - a \( j \)-ary predicate symbol \((R,S)\) for each relationship type \((R,S)\) of \( R \) with \(|S|=1\)
   - the binary predicate symbols (relational operators)
     \(<,\leq,\geq,=,\neq\)

c) **terms:**
   - each constant symbol
   - each variable
   - **selective terms** \( x.A \), where \( x \) is a variable and A a unary function symbol

d) **formulas:**
   - **comparative formulas** \( s \Theta t \), where \( \Theta \) is a relational operator and \( s,t \) are selective terms or constants, but not both are constants
   - **associative formulas** \( R(P_1:x_1, \ldots, P_t:x_t) \), where \((R,S)\) is a \( j \)-ary predicate symbol with \( S=\{P_1, \ldots, P_t\} \) and \( x_1, \ldots, x_t \) are variables
   - **(direct) type formulas** \( E(x) \), where \( E \) is a unary predicate symbol and \( x \) is a variable
   - (existentially or universally) quantified formulas
     \( (q x st E(x))(\phi) \), where \( q \) is a quantifier, \( x \) a variable, \( E(x) \) a type formula and \( \phi \) an arbitrary formula
   - molecular formulas \( \neg(\phi) \) and \((\phi \land \psi) \), where \( \phi \) and \( \psi \) are formulas and \( \lambda \in \{\land, \lor, \Rightarrow\} \)
Semantics of $\mathcal{L}_r$:

Let $S$ be a database state for scheme $r$.

a) semantics of terms:
Let $\alpha$ be a term of $\mathcal{L}_r$, $\bar{x} = x_1, \ldots, x_m$ a sequence of variables containing all those that occur in $\alpha$, and let $\bar{e} = e_1, \ldots, e_n$ be a sequence of internal identifiers.

The evaluation of $\alpha$ in $S$ under substitution $\bar{e}$ for $\bar{x}$ (denoted by $\alpha^S[\bar{e}]$) is defined as follows:

- $\alpha$ is a constant symbol 'v':
  \[ \alpha^S[\bar{e}] := v \]

- $\alpha$ is a variable $x_i$:
  \[ \alpha^S[\bar{e}] := \begin{cases} e_i, & \text{if } i \leq n \\ \text{undefined else} \end{cases} \]

- $\alpha$ is a selective term $x_i.A$:
  \[ \alpha^S[\bar{e}] := \begin{cases} A_s(x_i^S[\bar{e}] ), & \text{if } x_i^S[\bar{e}] \in \text{ext}_S(\text{poss}(A)) \\ \text{undefined else} \end{cases} \]

b) semantics of formulas:
Let $\phi(\bar{x})$ be a formula of $\mathcal{L}_r$, $\bar{x} = x_1, \ldots, x_m$ the (possibly empty) sequence of free variables in $\phi$ and $\bar{e}$ as above.

The truth value of $\phi(\bar{x})$ in $S$ under substitution $\bar{e}$ for $\bar{x}$ (denoted by $\phi(\bar{x})^S[\bar{e}]$) is defined as follows:

- $\phi(\bar{x})$ is a comparative formula $\alpha_1 \Theta \alpha_2$:
  \[ \phi(\bar{x})^S[\bar{e}] := \begin{cases} \text{true, if 1) there are value domains } V' \text{ and } V'' \text{ with} \\
  \alpha_1^S[\bar{e}] \in V' \text{ and } \alpha_2^S[\bar{e}] \in V'', \\
  \text{and } \Theta \text{ is defined between elements of these domains} \\
  \text{and} \\
  2) \alpha_1^S[\bar{e}] \Theta \alpha_2^S[\bar{e}] \text{ holds} \\
  \text{false, if 1) and not 2)} \\
  \text{undefined else} \end{cases} \]

- $\phi(\bar{x})$ is a type formula $\epsilon(x)$:
  \[ \phi(\bar{x})^S[\bar{e}] := \begin{cases} \text{true, if } x^S[\bar{e}] \in \text{ext}_S(\epsilon) \\
  \text{false else} \end{cases} \]
- \( \phi(\vec{x}) \) is an associative formula \( R(P_1:x_1, \ldots, P_t:x_t) \):

\[
\begin{cases}
\text{true, if } 1) \ x_i^s[e] \in \text{ext}_s(\text{role}(P_i)) \\
\quad \text{for all } i \\
\quad \text{and} \\
\text{2) for some } v \in \text{ext}_s(R): \\
\quad v(P_i) = x_i^s[e] \text{ holds} \\
\quad \text{for all } i \\
\text{false, if } 1) \text{ and not } 2) \\
\text{undefined else}
\end{cases}
\]

- \( \phi(\vec{x}) \) is an existentially quantified formula

\[
(\exists x \ \text{st} \ E(x))(\psi(\vec{x}, x)):
\begin{cases}
\text{true, if } \psi(\vec{x}, x)^s[e] = \text{true} \text{ for some } e \in \text{ext}_s(E) \\
\text{false, if } \psi(\vec{x}, x)^s[e] = \text{false} \text{ for all } e \in \text{ext}_s(E) \\
\text{undefined else}
\end{cases}
\]

\[ (\vec{x}, x) \text{ denotes the concatenation of sequence } \vec{x} \text{ with } x \]

The definition of the semantics for the remaining classes of formulas is obvious and therefore has been omitted.

Evaluations of terms and truth values of formulas have been partially defined to distinguish formulas like

\[
(\exists x \ \text{st} \ \text{EMPL}(x))(\exists y \ \text{st} \ \text{PART}(y))(y.\text{AGE} < x.\text{FLOOR})
\]

which is meaningless with respect to scheme \( \gamma_1 \), from those terms and formulas that are semantically correct but don't evaluate to \text{true} in the state \( S \) under consideration.

The technique used for the notation of substitution semantics has been influenced by that in [Jac 82].

- 8 -
2. A formal query language $L^0_{\mathcal{I}}$ based on $\mathcal{I}$

**Syntax of $L^0_{\mathcal{I}}$:**
The query language is the set of all syntactical queries of the form

\[
\text{get } x.A_1, \ldots, x.A_n \text{ where } \Psi(x) !
\]

target list qualification

where each $x.A_i$ is a selective term of $\mathcal{I}$ and $\Psi(x)$ is either a type formula $E(x)$ or the conjunction of $E(x)$ and an arbitrary formula $\phi(x)$ of $\mathcal{I}$.

In $L^0_{\mathcal{I}}$ we have only one variable in the target list. This restriction has been introduced to simplify the discussion of the important features in this paper; all results can be extended to queries with multi-variable target lists.

**Semantics of $L^0_{\mathcal{I}}$:**
Let $S$ be a database state for $\mathcal{I}$ and $Q$ a syntactical query.

The semantics of $Q$ in $S$ (denoted by $\text{sem}_S(Q)$) are defined as follows:

\[
\text{sem}_S(Q) := \begin{cases} 
  \{(x.A_1^S[e], \ldots, x.A_n^S[e]) \mid e \in \text{ext}_S(E) \land \phi(x)^S[e] = \text{true}\}, 
  & \text{if for all } e \in \text{ext}_S(E) \text{ both } \\
  \text{undefined} & \text{else }
\end{cases}
\]

If the qualification of $Q$ consists of $E(x)$ only, all parts of the definition dealing with $\phi$ have to be omitted.

**Contextual conditions for "meaningful" queries:**
Only queries $Q$ with $\text{sem}_S(Q) \neq \text{undefined}$ for every state $S$ can be considered meaningful. They can be characterized by syntactical properties of the resp. queries with respect to the underlying scheme $\mathcal{I}$.

Let $Q$ be an $L^0_{\mathcal{I}}$-query. $Q$ itself as well as every occurrence of a quantified formula in $Q$ is called a range of $Q$. The main part of a range $R$ is obtained by removing from $R$ every range properly contained in $R$. 

-9-
Expl.:

\[ \text{get } x, \# \quad \text{where DEPT}(x) \land \]
\[ \begin{cases} \exists y \text{ st PART}(y) \land \text{COLOR} = ' \text{red}' \land \end{cases} \]
\[ \begin{cases} \exists z \text{ st COMP}(z) \land \end{cases} \]
\[ \text{SUPPLY} \text{ORDERER}:x, \text{SUPPLIER}:z, \text{ORDERED ITEM}:y) \]

(-----: main part of \( R_1 \); =====: main part of \( R_2 \); main part of \( R_3 \) is identical with \( R_3 \))

Q is called a proper query if the following conditions hold:

P1) Each range of Q contains in its main part exactly one type formula.

P2) For each occurrence \( O \) of a variable \( y \) in a range \( R \) of Q there exists an occurrence of a type formula \( E(y) \) in the main part of some range containing \( R \) as a sub-range.

(Let the smallest range with this property contain a type formula \( E'(y) \). \( E' \) is then called the type of \( y \) at \( Q \), denoted by \( \text{type}_{y, Q} \).)

A proper query is in addition called admissible if the following holds:

A1) For each occurrence \( O \) of a variable \( x \) in a selective term \( x : A \) in Q: \( \text{type}_{x, Q} = \text{posa}(A) \)

A2) For each comparative formula \( x : A \ominus 'v' \) or \( 'v' \ominus x : A \) in Q: \( v \in \text{dom}(A) \)

A3) For each comparative formula \( x : A \ominus y : A' \) in Q: \( \ominus \) is defined between elements of \( \text{dom}(A) \) and \( \text{dom}(A') \)

A4) For each occurrence \( O \) of a variable \( x_i \) in an associative formula \( R(\ldots,P_i:x_i,\ldots) \) in Q: \( \text{type}_{x_i, Q} = \text{role}(P_i) \)

The meaningful queries in \( L^\text{SF} \) are exactly those syntactical queries which satisfy the contextual conditions P1, P2 (which are independent of \( \mathcal{F} \) ) and A1, ..., A4 (which are dependent on \( \mathcal{F} \)).

Theorem:

A syntactical query Q is admissible iff for every database state \( S \) for scheme \( \mathcal{F} \) \( \text{sem}_S(Q) \neq \text{undefined} \) holds.
3. Simplifying navigational parts of a query by syntactical tools

3.1 Incomplete associative formulas: \( L^1_f \)

A first and very obvious relaxation of the strict specification requirements in \( L^0_f \) is possible in cases where relationships of arity > 2 are involved.

\[
(Q^0_1)^* \quad \text{get } x.\# \text{ where DEPT}(x) \land (\exists y \text{ st PART}(y))(y.\text{COLOR}='\text{red}') \land (\exists z \text{ st COMP}(z)) \\
(SUPPLY(\text{ORDERER}:x,\text{SUPPLIER}:z, \text{ORDERED ITEM}:y))!
\]

In this query variable \( z \) is applied only once, namely in the associative formula, whereas \( y \) occurs in the comparative formula, too. If a standard agreement exists that missing participants of an associative formula are considered as additional existentially quantified variables not used otherwise, then \( z \) can be omitted and \( Q^0_1 \) may be expressed by

\[
(Q^1_1)^* \quad \text{get } x.\# \text{ where DEPT}(x) \land (\exists y \text{ st PART}(y))(y.\text{COLOR}='\text{red}') \land \text{SUPPLY}(\text{ORDERER}:x,\text{ORDERED ITEM}:y))!
\]

We extend \( L^0_f \) to cover this kind of incomplete associative formulas:

\text{Syntax of } L^1_f :

Terms and formulas are those of \( L^0_f \) plus incompl. assoc. formulas

\( R(P_1:x_1,\ldots,P_k:x_k) \) where \((R,S)\) is a predicate symbol of \( \mathcal{X}_f \) with \( \{P_1,\ldots,P_k\} \subseteq S \) and \( x_1,\ldots,x_k \) are variables.

Syntactical queries are defined in the same way as in \( L^0_f \) allowing incompl. assoc. formulas to occur in the qualification, too.

\text{Semantics of } L^1_f :

The semantics of \( Q \in L^1_f \) are defined to be that of the \( L^0_f \)-query transform\(^1_0\)(Q)

introduced as follows:

If \( Q \) does not contain any incompl. assoc. formula, then transform\(^1_0\)(Q):=Q.
Else for each incompl. assoc. formula in \( Q \) the following is executed:
The non-specified roles are determined. For each such \( P_i \) a new existentially quantified variable with type \( \text{role}(P_i) \) is added. The associative formula itself is completed by insertion of the missing roles coupled with "their" variable.
The contextual conditions are the same as in \( L^0_f \).

* The upper index \( i \) of an example query \( Q^i_j \) indicates that \( Q^i_j \in L^i_j \).
3.2 Indirect type formulas: $L^2_T$

An effective method for the introduction of implicit navigation is provided on the next level; it can be applied if relationships are involved. In most cases associative formulas - even incomplete ones - contain parts which can be inferred from others; e.g.,

```
SUPPLY(ORDERER:x ,ORDERED ITEM:y)
```

in $Q^1_1$ is a redundant way of expressing the association between $x$ and $y$. The specification of either the involved relationship, or of one of the participating roles would be sufficient. Thus, a notion like

```
x.ORDERED ITEM(y)
```

to be read as "$y$ plays the role of an ordered item in its relationship to $x"$, can represent the semantic connection between the two variables. The relationship type, SUPPLY, as well as the non-specified role, ORDERER, of $x$ are uniquely inferable from ORDERED ITEM and DEPT, which is the type of $x$ in $Q^1_1$. However, this is only possible because the role of DEPT in SUPPLY is unique!

Furthermore, from the notion $x.ORDERED ITEM(y)$ the entity type of variable $y$ is uniquely inferable, too, so that the following abbreviation of $Q^1_1$ is possible:

```
(Q^2_1) get x.# where DEPT(x) \& (\exists y st x.ORDERED ITEM(y)) (y.COLOR='red')
```

As $x.ORDERED ITEM(y)$ replaces the type formula of $y$, it is called indirect type formula.

Obviously the new notion can be generalized. For this reason consider a query containing two indirect type formulas associated with two nested existential quantifiers:

```
(Q^2_2) get x.NAME where EMPL(x) \& (\exists y st x.EMPLOYER(y)) ((\exists z st y.SOLD ITEM(z)) (z.TYPE='car') )
```

Variable $y$ serves only as a connection between the target list variable, $x$, and variable $z$ further referenced in the innermost range of the query. This kind of connection can equally be expressed without explicitly using $y$:
\((Q_2^2)\) \textbf{get} x.NAME \textbf{where} EMPL(x) \land (\exists z \textbf{st} x.EMPLOYER.SOLD ITEM(z)) (z.TYPE='car') !

The notion \(x.\text{EMPLOYER}.\text{SOLD ITEM}(z)\) can be considered as a more general kind of indirect type formula subsuming two associative, two type formulas and one intermediate variable. The type part \(x.\text{EMPLOYER}.\text{SOLD ITEM}\) is uniquely assigned to the path \(\text{EMPL} \rightarrow \text{MEMBERSHIP} \rightarrow \text{DEPT} \rightarrow \text{SALES} \rightarrow \text{PART}\) in the \(\gamma_1\)-diagram and is therefore called a path term. Again the usability of a path term in an indirect type formula depends on the fact that the non-specified roles are unique and the intermediate variable is existentially quantified.

However, even paths in which one or more roles might be ambiguous, if not stated explicitly, can be handled by means of a slightly modified kind of path terms. Such paths can be constructed only if the underlying diagram contains "small" cycles with two edges and two nodes, like in

\[
\begin{aligned}
\text{SURGEON} & \rightarrow \text{ANAESTHETIST} & \text{OPERATION} & \rightarrow \text{PATIENT} \\
\text{DOCTOR} & \rightarrow \text{NAME TITLE} & \text{SURGICAL ASSISTANT} & \rightarrow \text{PERSON} \\
\end{aligned}
\]

\((Q_3^2)\) \textbf{get} x.NAME \textbf{where} DOCTOR(x) \land (\exists y \textbf{st} PERSON(y)) (OPERATION(SURGEON:x,PATIENT:y) \land y.\text{AGE}='95') !

This query cannot be further simplified by introduction of \(x.\text{PATIENT}(y)\) as a type formula for \(y\), because the role \text{SURGEON}\) of \(x\) is not inferable from \text{DOCTOR} and \text{PATIENT}.

We have to state \text{SURGEON} explicitly by adding it to the path term as an index of \(x\), yielding

\(x[\text{SURGEON}].\text{PATIENT}(y)\),

to be read as "\(y\) plays the role of a patient in its relationship to \(x\) acting as a surgeon".
Now we are able to allow

\[(Q^2_3) \quad \text{get } x.\text{NAME where } \text{DOCTOR}(x) \land (\exists y \text{ at } x[\text{SURGEON}].\text{PATIENT}(y)) \land (y.\text{AGE}='95')!\]

Indexing may equally be applied to components of "longer" path terms for the disambiguation of non-specified roles. Thus, in

```
E_0 — P_0 — E_1 — P_2 — E_2 — P_3 — E_3 — P_4 — E_4
```

the path between \(E_0\) and \(E_4\) running through the \(P_2\)-edge can be uniquely described by the path term

\[x.P_1[P_2].P_3.P_5\]

(where \(x\) is an \(E_0\)-variable).

Finally, we have to look at another kind of indirect type formulas.

\[(Q^1_4) \quad \text{get } x.\text{NAME where } \text{EMPL}(x) \land (\exists y \text{ at } \text{DEPT}(y)) \land (\text{SALES}(\text{SELLER};y) \land x.\text{SALARY} < y.\text{BUDGET})!\]

In analogy to the replacement

```
\text{PART}(y) \quad \text{..} \quad \text{SUPPLY}(\text{ORDERER } x, \text{ORDERED ITEM};y)
```

in \(Q^2_1\) we introduce a new kind of indirect type formula allowing the replacement

```
\text{DEPT}(y) \quad \text{..} \quad \text{SALES}(\text{SELLER};y)
```

in \(Q^1_1\) which can now be abbreviated by

\[(Q^2_4) \quad \text{get } x.\text{NAME where } \text{EMPL}(x) \land (\exists y \text{ at } \text{SELLER}(y)) \land (x.\text{SALARY} < y.\text{BUDGET})!\]

The indirect type formula \(\text{SELLER}(y)\) is called \textit{independent}, while \(x.\text{ORDERED ITEM}(y)\) is called \textit{dependent} (on variable \(x\)).

Both, dependent and independent indirect type formulas may coexist in a query.
Moreover, an indirect type formula need not follow a quantifier:

\((Q_2^2) \quad \text{get} \ x.AGE \ \text{where} \ \text{EMPL}(x) \ \wedge \ (\exists y \ \text{st} \ x.\text{EMPLOYER}(y)(\text{SELLER}(y)))!\)

As independent indirect type formulas are not describing connections between variables a chaining of roles - like in path terms - is not provided.

**Syntax of \(L^2_r\):**

In addition to the terms and formulas of \(L^1_r\):

- **(complete) path terms**  \(\hat{x} \hat{p}_1 \hat{p}_2 \ldots \hat{p}_{n-1} p_n\) where \(n \geq 1\) and
  - \(\hat{x}\) is a variable or \(\hat{x} = x[p_0]\) where \(x\) is a variable and \(p_0\) a role
  - for \(1 \leq i \leq n-1:\) either \(\hat{p}_i\) is a role \(p_i\)
    or \(\hat{p}_i = p_i[p_i]\) and \(p_i, p_i\) are roles
  - \(p_n\) is a role
- **indirect type formulas**
  - **independent:** \(P_i(y)\) where \(P_i\) is a role and \(y\) a variable
  - **dependent:** \(\hat{x}(y)\) where \(\hat{x}\) is a path term and \(y\) a variable

Indirect type formulas may occur at all places where direct ones are allowed.

**Semantics of \(L^2_r\):**

Again the semantics of an \(L^2_r\)-query is defined as the semantics of an \(L^1_r\)-query transform \(Q\).

The outline of the transformation strategy can be given as follows:

1) Indirect type formulas not occurring immediately behind \(\text{st}\):
   - \(P(y)\) is replaced by \(R(P;y)\) with \(R=\text{poss}(P)\);
   - \(x.P_1(y)\) is replaced by \(R(P_1;y, P_0;x)\) with \(R=\text{poss}(P_1)\) and
     \(P_0\) is the uniquely determined role with
     \(\text{poss}(P_0) = R\) and \(\text{role}(P_0) = \text{type}_{x,0}\)
     (0 denotes the occurrence of \(x\) in the resp. formula).
     If no such \(P_0\) exists, the semantics of \(Q\) are undefined.
   - \(x[P_0].P_1(y)\) is replaced by \(R(P_1;y, P_0;x)\) with \(R=\text{poss}(P_1)\).
     If \(R=\text{poss}(P_0)\) and \(\text{role}(P_0) = \text{type}_{x,0}\) does not hold, the semantics of \(Q\) are undefined.
2) Indirect type formulas occurring immediately behind \texttt{st}:
The following replacements are performed
\[
\begin{align*}
(q \ y \ \texttt{st} \ P(y))(\dagger) & \quad \longrightarrow \quad (q \ y \ \texttt{st} \ E(y))(R(P; y) \ \lambda \ \phi) \\
(q \ y \ \texttt{st} \ x. P_1(y))(\dagger) & \quad \longrightarrow \quad (q \ y \ \texttt{st} \ E(y))(R(P_1; y, P_0; x) \ \lambda \ \phi) \\
(q \ y \ \texttt{st} \ [P_0]. P_1(y))(\dagger) & \quad \longrightarrow \quad (q \ y \ \texttt{st} \ E(y))(R(P_1; y, [P_0]. P_1(y)) \ \lambda \ \phi)
\end{align*}
\]
where \( q \in \{\exists, \forall\} \), \( E = \text{role}(P) \) or \( = \text{role}(P_1) \), resp., and
\[
\lambda = \begin{cases} \\
& \land \; \text{if} \; q = \exists \\
& \Rightarrow \; \text{if} \; q = \forall \end{cases}
\]
The construction of the associative formulas as well as the conditions for the cases of undefined semantics are the same as under 1).

3) Each occurrence of an indirect type formula \( \hat{x}. P_1 \hat{\ldots} P_{n-1}. P_n(y) \) of length \( n > 1 \) within the same \( x \)-range is resolved by means of an additional variable \( z \) not yet occurring in \( Q \). It is bound by \( (\exists \ z \ \texttt{st} \ \hat{x}. P_1(z)) \) where \( \hat{P}_1 \in \{P_1, P_1[P_1]\} \) and added to the respective \( x \)-range. The type formula itself is then replaced by substituting \( z \) for \( \hat{x}. P_1 \) throughout that range.

4) Steps 1 - 3 are repeated until no indirect type formula remains in \( Q \).

The contextual conditions are augmented by two further conditions for admissible queries:

A5) For each occurrence of a complete path term \( \hat{x}. P_1 \ldots P_{n-1}. P_n \) in \( Q \): there exists a unique path
\[
\begin{array}{c}
E_0 \\
\text{P} 1 1 \\
\text{E}_1 \\
\text{P} 1 2 \\
\ldots . . \\
\text{P} n 1 \\
\text{E}_n \\
\end{array}
\]
in the \( \gamma \)-diagram such that
\[
\begin{align*}
- E_0 &= \text{type}_{x, 0} \quad (0 \; \text{denotes} \; \text{the occurrence of} \; x \; \text{in the path term}) \\
- P_{11} &= P_0 \; \text{if} \; \hat{x} = x[P_0] \\
- \text{for} \; 1 \leq i \leq n-1: \\
& \quad - P_{12} = P_i \; \text{if} \; \hat{P}_i = P_i \\
& \quad - P_{12} = P_i \; \text{and} \; P_{(i+1)1} = P_i \; \text{if} \; \hat{P}_i = P_i[P_i] \\
- P_{n2} &= P_n
\end{align*}
\]

A6) For each occurrence 0 of an independent indirect type formula \( P(x) \) in \( Q \):
\[
\text{type}_{x, 0} = \text{role}(P)
\]
Furthermore, condition P2 - assuring that in a proper query there is a type provided for each variable - has to be reformulated to cover indirect type formulas as well. The revised version is:

P2') For each occurrence \( y \) of a variable \( y \) in a range \( R \) of \( Q \), there exists an occurrence of a type formula \( T(y) \) in the main part of some range that contains \( R \) as a sub-range.

( Let the smallest range with this property contain a type formula \( T'(y) \). The type of \( y \) at \( 0 \) is defined to be

\[
\text{type}_{y,0} := \begin{cases} 
T', & \text{if } T' \text{ is an entity type} \\
\text{role}(T'), & \text{if } T' \text{ is a role} \\
\text{role}(P_n), & \text{if } T' = x.P_1 \ldots P_{n-1}.P_n \text{ is a path term}
\end{cases}
\]

\[3.3\] Path terms on the positions of variables: \( L^3_{\rho}\)

The additional syntactical tools of level 1 and 2 allow us to dispense with a considerable amount of associative formulas and intermediate variables. It is, however, still necessary to have in a query at least as many variables as there are entity types addressed by attributes occurring in \( Q \). To get rid of further variables on the third level path terms may be used not only as a description of complex types for variables, but directly as a substitute for variables. This method again can be applied only if certain uniqueness conditions are satisfied.

Look at example query \( Q_1^2 \) from section 3.2:

\[
\begin{align*}
\text{get } x, \# & \text{ where DEPT}(x) \land (\exists y \text{ st } x.\text{ORDERED ITEM}(y)) \\
& \quad (y.\text{COLOR}='\text{red}')
\end{align*}
\]

As \( y \) is the only variable of "type" \( x.\text{ORDERED ITEM}, \) it can be uniquely represented by that path term. We get:

\[(Q_1^3) \quad \begin{align*}
\text{get } x, \# & \text{ where DEPT}(x) \land x.\text{ORDERED ITEM}.\text{COLOR}='\text{red}'
\end{align*}\]

The concatenation of path term \( x.\text{ORDERED ITEM} \) and attribute \( \text{COLOR} \) (by means of ".") can be interpreted as the concatenation of the path

\[
\begin{array}{ccc}
\text{DEPT} & \text{ORDERER} & \text{SUPPLY} & \text{ORDERED ITEM} & \text{PART}
\end{array}
\]
and the "path"

\[ \text{PART} \rightarrow \text{COLOR} \rightarrow \text{V} \]

with \( V = \text{dom} \text{(COLOR)}. \)

This is consistent with the chaining of roles in path terms of length \( > 1 \) where each role represents a link in a path. \( Q_1^3 \) does not make use of any quantifier. The complete navigational information - compare \( Q_1^0 \) - is now "hidden" in the short path term \( x.\text{ORDERED ITEM}. \)

In a similar manner we can simplify other example queries of section 3.2:

\[
\begin{align*}
(Q_2^3) & \quad \text{get} \ x.\text{NAME where } \text{EMPL}(x) \land x.\text{EMPLOYER}.\text{SOLD ITEM}.\text{TYPE}=\text{'car'}! \\
(Q_2^2) & \quad \text{get} \ x.\text{NAME where } \text{DOCTOR}(x) \land x.\text{[SURGEON]}'.\text{PATIENT}.\text{AGE}=\text{'95'}! \\
(Q_3^3) & \quad \text{get} \ x.\text{AGE where } \text{SELLER}(x.\text{EMPLOYER})! \\
\end{align*}
\]

A general interpretation of queries containing a path term \( \hat{x}.\hat{P} = \hat{x}.\hat{P}_1.\hat{P}_2...\hat{P}_{n-1}.\hat{P}_n \) on the position of a variable is given by the rule:

Introduce an additional variable \( y \), quantified by \(( \exists \text{st } \hat{x}.\hat{P}(y))\) at the beginning of the respective range of \( x \), and replace each occurrence of \( \hat{x}.\hat{P} \) in the range under consideration by \( y \).

Problems arise, however, if we are applying this convention to relationships that are non-binary:

\[
\begin{align*}
(Q_6^1) & \quad \text{get} \ x.\text{COLOR where } \text{PART}(x) \land ( \exists \text{y st DEPT}(y)) \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{( SUPPLY( ORDERED ITEM:x, \text{ORDERER:y,} \text{SUPPLIER:z})} \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{y.\#}=\text{'25'} \land z.\text{CITY}=\text{'Paris'}))! \\
\end{align*}
\]

The query

\[
\begin{align*}
(Q_6^3) & \quad \text{get} \ x.\text{COLOR where } \text{PART}(x) \land x.\text{ORDERER}.\# = \text{'25'} \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{A } x.\text{SUPPLIER}.\text{CITY}=\text{'Paris'}! \\
\end{align*}
\]

seems to be a useful abbreviation of \( Q_6^1 \).

---

* In section 3.5 B queries containing multiple occurrences of a path term are discussed.
The proposed interpretation mechanism takes it back to
\( (Q_6^2) \quad \text{get } x.\text{COLOR where } \text{PART}(x) \land \) 
\( (\exists \ y \text{ at } x.\text{ORDERER}(y)) \land \) 
\( (\exists \ z \text{ at } x.\text{SUPPLIER}(z)) \) 
\( (y.\#='25' \land z.\text{CITY}='Paris')! \)

which in turn is transformed to
\( (Q_6^1) \quad \text{get } x.\text{COLOR where } \text{PART}(x) \land \) 
\( (\exists \ y \text{ at } \text{DEPT}(y)) (\text{SUPPLY}(\text{ORDERER}:y,\text{ORD.ITEM}:x)) \land \) 
\( (\exists \ z \text{ at } \text{COMP}(z)) \) 
\( (\text{SUPPLY}(\text{SUPPLIER}:z,\text{ORD.ITEM}:x) \land 
\ y.\#='25' \land z.\text{CITY}='Paris')! \)

This query is apparently not semantically equivalent to \( Q_6^1 \)!
The main reason is that the proposed interpretation strategy exclusively supports binary associations. Non-binary ones, however, should have equal rights!

To obtain this aim the interpretational convention has to be modified as follows:

If in a query there are \( m \geq 2 \) different path terms
\[ \hat{x}.P_1, \hat{P}_1, \ldots, \hat{x}.P_m, \hat{P}_m \]
with the same prefix \( \hat{x}.P_1 \) (where \( P_1 \) is a possibly empty sequence), with \( \text{poss}(P_1) = \ldots = \text{poss}(P_m) = R \) and arbitrary postfix sequences \( \hat{P}_i \) occurring in the same x-range:
Introduce \( m \) new variables \( y_1, \ldots, y_m \) and one common associative formula
\[ R(P_0: \hat{x}.P_1, P_1:y_1, \ldots, P_m:y_m) \]
in the respective range. Quantify each \( y_i \) in the form
\[ (\exists \ y_i \text{ at } \text{E}_i(y_i)) \]
where \( \text{E}_i = \text{role}(P_1) \). Afterwards replace each occurrence of \( \hat{x}.P_i.\hat{P}_i \) in the x-range under consideration by \( y_i \) for \( 1 \leq i \leq m \).
The remaining path terms \( \hat{P}_i \) have to be further reduced if \( \hat{P}_i \)
is non-empty.

Under this modified convention \( Q_6^3 \) is directly reduced to \( Q_6^1 \).

It seems important to us to summarize at this point those two conventions we have chosen for the reduction of path terms on the positions of variables:

1) Within the same x-range each occurrence of a path term \( \hat{x}.P \)
standing on the position of a variable - or even being the prefix of a longer path term occurring on such a position - is replaced in the same way, i.e., by means of the same new variables.
2) In addition, all path term prefixes $x \cdot \overrightarrow{P} \cdot P_1 \ldots \cdot x \cdot \overrightarrow{P} \cdot P_m$ in
the same $x$-range being identical up to the last role are
reduced by variables that are associated by means of one
occurrence of an associative formula, if $P_1 \ldots , P_m$ have the
same possessor.

Syntax of $L^3_\gamma$:
Terms and formulas in addition to those of $L^2_\gamma$ are:
- selective terms $\hat{x} \cdot \overrightarrow{P} \cdot A$ where $\hat{x} \cdot \overrightarrow{P}$ is a path term and $A$
a unary function symbol
- comparative formulas involving this new kind of selec-
tive terms
- associative formulas where one or more positions are
occupied by path terms instead of variables
- type formulas $T(\hat{x} \cdot \overrightarrow{P})$ where $T$ is an entity type, a role
or a path term and $\hat{x} \cdot \overrightarrow{P}$ is a path term, too.

Syntactical queries of $L^3_\gamma$ may contain any kind of $L^3_\gamma$-formula
in their qualification. Selective terms with path terms as a
prefix are not allowed in the target list because they would
have to be reduced to queries with more than one target list
variable which are not provided.

Semantics of $L^3_\gamma$:
A query $Q \in L^3_\gamma$ is transformed into a $L^2_\gamma$-query transform$^3_\gamma(Q)$ by
means of an algorithm that eliminates path terms on the posi-
tions of variables according to the above-mentioned conventions.
It shall not be discussed in detail.
The contextual condition that provides a proper application of
path terms in $L^3_\gamma$-queries with respect to their context is:
A 7) For each occurrence of a path term $\hat{x} \cdot \overrightarrow{P} = \hat{x} \cdot \overrightarrow{P} \cdot P_1 \ldots \cdot P_{n-1} \cdot P_n$
- in a selective term: $\hat{x} \cdot \overrightarrow{P} \cdot A$: $\text{role}(P_n) = \text{poss}(A)$
- in an assoc. formula $R(\ldots , P_i : \hat{x} \cdot \overrightarrow{P} , \ldots )$:
  $\text{role}(P_n) = \text{role}(P_i)$
- as the argument of a type formula $T(\hat{x} \cdot \overrightarrow{P})$:
  $\text{role}(P_n) = \begin{cases} T, & \text{if } T \text{ is an entity type} \\
  \text{role}(T), & \text{if } T \text{ is a role} \\
  \text{role}(P_m'), & \text{if } t = \hat{y} \cdot \overrightarrow{P} \cdot P_m \text{ is a path term}
\end{cases}$
3.4 Incomplete path terms and indirect selection: $L^4_R$

Our last step towards the introduction of implicit navigation is based on a further discussion of the paths described by complete path terms.

To cover all relevant cases by means of one example let us look at employees, parts, departments and companies in a slightly different constellation:

(Part of an) ER-diagram for scheme $F^3$:

An example query directed against a $F^3$-database is

$$(Q^3_7) \quad \text{get } x \cdot \text{TYPE where } \text{PART}(x) \land x \cdot \text{PRODUCER.EMPLOYER.SUPPLIER.CITY} = 'Paris'$$

The path term $x \cdot \text{PRODUCER.EMPLOYER.SUPPLIER}$ uniquely identifies one of the two paths directly connecting PART and COMPANY. However, the representation of this path by means of the above path term is still redundant, because two of the three links involved are unique within the diagram.

While EMPLOYER is inevitable to decide via which relationship EMPL and DEPT are to be connected, it is not necessary to mention PRODUCER, as there is an unambiguous direct connection between the EMPLOYER-edge and the PART-node associated with $x$.

Thus, it makes sense to use the abbreviation

$$(Q^4_7) \quad \text{get } x \cdot \text{TYPE where } \text{PART}(x) \land x .. \text{EMPLOYER.SUPPLIER.CITY} = 'Paris'$$
Analogously the link represented by SUPPLIER is an unambiguous direct connection between EMPLOYER and CITY. Consequently we may write

$$(\bar{Q}_7^4) \quad \text{get } x.\text{TYPE where } \text{PART}(x) \land x..\text{EMPLOYER..CITY}='Paris'!$$

Note, however, that we don't have unique paths between PART and EMPLOYER or between EMPLOYER and CITY, resp., as paths may contain more than one occurrence of an edge or a node. So, e.g., the path uniquely described by the sequence EMPLOYER, MANAGER, EMPLOYER, SUPPLIER, CITY is a legal path between EMPLOYER and CITY, too. However, there is a unique direct path (i.e., a path without any node (or edge) occurring more than once) between both PART and EMPLOYER as well as EMPLOYER and CITY. So, it is possible to uniquely infer $Q_7^3$ from $\bar{Q}_7^4$.

Path terms where one or more "missing roles" are indicated by a double dot within the sequence of roles are called incomplete path terms.

Selective terms where a double dot in front of the attribute indicates further "missing roles" are called indirect selective terms.

Both features may only be applied in cases where each "gap" can be filled up by a uniquely determined sequence of roles (associated with a direct path) to yield a complete path term or a direct selective term, resp..

Last not least, in $L_7^4$ we will be able to omit explicit typing of the target list variable, because the uniqueness condition for attributes assures a unique deduction of this type by means of the poss-function. So, the ultimate version of $Q_7$ is

$$(Q_7^4) \quad \text{get } x.\text{TYPE where } x..\text{EMPLOYER..CITY}='Paris'!$$

Syntax of $L_7^4$:

- Terms and formulas not present in one of the lower-level languages are:
  - **incomplete path terms** $\hat{x}.\hat{P}_1\ldots\hat{P}_{n-1}.P_n$ where
    - $\hat{x}$ is a variable or $\hat{x}=x[x[P_0]$ where $x$ is a variable and $P_0$ a role
    - For $1 \leq i \leq n-1$:
      - $\hat{P}_i \in \{\hat{P}_i, P_i\}$ where $\hat{P}_i \in \{P_i, P_i[P_i]\}$ and $P_i, P_i'$ are roles
    - $P_n \in \{P_n, P_n\}$ where $P_n$ is a role
- **indirect selective terms** \( \hat{x}.A \) and \( \hat{x}P.A \) where \( \hat{x} \) is as defined above, \( \hat{x}P \) is a (complete or incomplete) path term and \( A \) a unary function symbol.

- formulas where both kinds of path and selective terms appear

Syntactical queries have the format

\[
\text{get } \ x.A_1, \ldots, x.A_n \ \text{where } \phi(x) \ 1
\]

where \( \phi(x) \) is an arbitrary \( L^A_\gamma \)-formula.

**Semantics of \( L^A_\gamma \):**

The algorithm \( \text{transform}^4_\gamma \) is constructed according to the following principles:

1) A missing type formula for \( x \) is inferred from \( x.A_1 \) by means of poss and conjunctively added to the qualification.

2) Every incomplete path term or indirect selective term, resp., is transformed into a complete path term or direct selective term, resp., as follows:

Every "gap" \( P^1_i P^1_{i+1} \) is filled up by the sequence \( P^1_i \ldots P^k_i \) with \( P^j_i \in R_\gamma \) such that

\[
\begin{array}{cccccc}
P_i & - & P^1_i & - & \ldots & - & P^k_i & - & P_{i+1} \\
\end{array}
\]

is a unique direct path in the \( \gamma \)-diagram (i.e., the \( P^j_i \) are all different from each other and \( P_{i+1} \notin \{ P^1_i, \ldots, P^k_i \} \)).

"Front end-" and "back end-gaps" \( \hat{x}P_1 \) and \( P_n \ldots A \), resp., are treated analogously.

If no or more than one completion exists for any "gap", \( \text{transform}^4_\gamma \) has an undefined result.

Additional contextual conditions in \( L^A_\gamma \) are recording the different variants of the "existence-of-a-unique-completion"-condition for incomplete path/indirect selective terms. They are testable by inspection of \( \gamma \) and therefore augment the set of admissibility conditions.
3.5 Universal quantifiers and multiple occurrences of variables with identical type

All syntactical features introduced in the previous sections aim at queries with special properties: there are chains of existentially quantified variables connected by means of associative formulas; the chains describe paths that are composed of uniquely identifiable connections between the participating entity types. In addition, each "chain type" occurs only once in such a query. If the wide class of queries without these properties is considered, the abbreviation facilities do hardly apply. Especially queries that do not make use of any associative formula at all but interconnect variables by means of comparative formulas are not supported by the offered abbreviative tools. The same applies to queries where only universal quantifiers occur. In the following a few further examples are demonstrating how the non-supported syntactical constructs are embedded in our language hierarchy.

A. Queries involving universal quantifiers

Let \( U \) be a universally quantified formula. If \( U \) is nested inside of existentially quantified formulas or is itself containing further (possibly nested) formulas of that kind, the existential quantifiers can be saved by application of the usual conventions independent of the "disturbing" universal quantifier in between. Consider

\[
(Q_8^1) \quad \text{get } x.\text{TYPE} \text{ where } PART(x) \quad \exists y \ st \ EMPL(y)( \text{ PRODUCTION(PRODUCER:y,PRODUCT:x)} \wedge ( \forall z \ st \ DEPT(z)( \text{ MEMBERSHIP(MEMBER:y,EMPLOYER:z)} \Rightarrow ( \exists v \ st \ COMPANY(v)( \text{ SUPPLY(ORDERER:z,SUPPLIER:v)} \wedge v.CITY='Paris' )))
\]

and its \( L_s^4 \) version

\[
(Q_8^4) \quad \text{get } x.\text{TYPE} \text{ where } ( \forall z \ st \ x..\text{EMPLOYER}(z))( z..\text{CITY='Paris'} )
\]

A comparison with \( Q_7 \) - which is closely related to \( Q_8 \) - shows that the single term \( x..\text{EMPLOYER..CITY} \) in \( Q_7^4 \) has been split into two parts by the insertion of the universally quantified variable. In \( Q_8^1 \) the connection of the universally quantified variable, \( z \), and the "outer" variable \( y \) appeared in front of the implication symbol.
In this case the respective assoc. formula could be absorbed by an indirect type formula for z. If, on the other hand, the assoc. formula appears in the conclusion of an implication, two type formulas are required:

\[(Q_9^4) \quad \text{get } x.\text{TYPE where } \ldots \text{ (like in } Q_8^4) \ldots (\forall z \text{ st DEPT}(z) \implies
(\exists v \text{ st COMPANY}(v)(\ldots) \Rightarrow MEMBERSHIP(MEMBER:y, \text{EMPLOYER}:z))!\]

abbreviated by

\[(Q_9^4) \quad \text{get } x.\text{TYPE where } (\forall z \text{ st DEPT}(z))(z..\text{CITY}='\text{Paris}') \Rightarrow x..\text{EMPLOYER}(z))!\]

B. Queries involving multiple occurrences of variables with identical type

When resolving a path term \(\hat{x}.P_1\) occurring on the position of a variable, it was assumed that each occurrence of \(\hat{x}.P_1\) should be replaced by the same new variable. There are, however, numerous examples where it is desired to have different variables which are interconnected with some other (common) variable in identical manner:

\[(Q_{10}^1) \quad \text{get } x.\text{SALARY where } \text{EMPL}(x) \land
(\exists y \text{ st DEPT}(y))(\text{MEMBERSHIP}(\text{MEMBER}:x, \text{EMPLOYER}:y)
\land (\exists z_1 \text{ st PART}(z_1))(\text{SALES}(\text{SELLER}:y, \text{SOLD ITEM}:z_1)
\land z_1.\text{COLOR}='\text{red}')
\land (\exists z_2 \text{ st PART}(z_2))(\text{SALES}(\text{SELLER}:y, \text{SOLD ITEM}:z_2)
\land z_2.\text{COLOR}='\text{green}') )!\]

Because of the convention just mentioned
\[(Q_{10}^4) \quad \text{get } x.\text{SALARY where } x..\text{COLOR}='\text{red}' \land x..\text{COLOR}='\text{green}' !\]
would not be reduced to \(Q_{10}^1\).

Multiple occurrences have to be indicated explicitly by means of variables occurring in the query, so that only a lower-level representation is provided for \(Q_{10}^1\), namely

\[(Q_{10}^4) \quad \text{get } x.\text{SALARY where } (\exists z_1 \text{ st } x..\text{SOLD ITEM}(z_1))
(z_1.\text{COLOR}='\text{red}')
\land (\exists z_2 \text{ st } x..\text{SOLD ITEM}(z_2))
(z_2.\text{COLOR}='\text{green}') )!\]
4. Some features of a query language with English keywords based on $L^4_Y$

The last chapter is dedicated to example queries illustrating a formal query language which expresses formulas of $L^4_Y$ by means of various English keywords. For many queries of this language the semantics - defined by means of the transformations

\[ \text{transform}^4_{\frac{3}{2}}, \ldots, \text{transform}^1_0 \]

can immediately be recognized by the user.

The proposed syntactical framework requires that all roles in the underlying ER-scheme are English nouns or nouns combined with an adjective.

Each example query is given in combination with its $L^4_Y$-equivalent:

a) direct and indirect selective terms:

\[
\text{get } \text{NAME where } \text{AGE < '30' and } \text{SALARY < BUDGET} \]

in $L^4_Y$:

\[
\text{get } x.\text{NAME where } x.\text{AGE < '30'} \land x.\text{SALARY < x..BUDGET} \]

b) comparative formula including a path term:

\[
\text{get } \text{NAME where } \text{SALARY > SALARY of MANAGER} \]

in $L^4_Y$:

\[
\text{get } x.\text{NAME where } x.\text{SALARY > x.MANAGER.SALARY} \]

c) indexed path term and backward reference:

\[
\text{get } \text{NAME of DOCTOR where } \text{AGE of PATIENT of (this DOCTOR as SURGEON)} = '95 \]

in $L^4_Y$:

\[
\text{get } x.\text{NAME where } \text{DOCTOR(x) \land x[SURGEON].PATIENT.AGE='95'} \]

d) comparative formula inside a quantified formula including an outer variable:

\[
\text{get } \text{NAME where } \text{AGE < AGE of every EMPL with NAME='Maier'} \]

in $L^4_Y$:

\[
\text{get } x.\text{NAME where } (\forall y \text{ at EMPL}(y))( y.\text{NAME='Maier'} \Rightarrow x.\text{AGE < y.AGE} ) \]

- 26 -
e) parallel universally and negated quantified formulas:
\[
\text{get } \# \text{ where } \forall y \text{ st } x.\text{SOLD ITEM}(y)(y.\text{TYPE}='\text{car'}) \land \\
\text{no SUPPLIER has } \text{CITY}='\text{Paris}' !
\]
in \( L^4 \):
\[
\text{get } x. \# \text{ where } (\forall y \text{ st } x.\text{SOLD ITEM}(y)(y.\text{TYPE}='\text{car'}) \land \\
\neg(\exists z \text{ st } x.\text{SUPPLIER}(z)(z.\text{CITY}='\text{Paris}')) !
\]
f) multiple occurrences of variables with identical type:
\[
\text{get } \text{SALARY of } \text{MANAGER}
\text{ where some } \text{SOLD ITEM has } \text{COLOR}='\text{red}'
\text{ and }
\text{some other } \text{SOLD ITEM has } \text{COLOR}='\text{green}' !
\]
in \( L^4 \):
\[
\text{get } x.\text{SALARY where } \text{MANAGER}(x) \land (\exists z_1 \text{ st } x.\text{SOLD ITEM}(z_1))
\text{ ( } z_1.\text{COLOR}='\text{red}' )
\land (\exists z_2 \text{ st } x.\text{SOLD ITEM}(z_2))
\text{ ( } z_2.\text{COLOR}='\text{green}' ) !
\]

References:

[Cod 72] CODD, E.,
Relational Completeness of Data Base Sublanguages,

[C/K 76] CARLSON, C.R. and KAPLAN, R.S.,

[HOT 76] HALL, P., OWLETT, J. and TODD, S.,

[Che 76] CHEN, P.P.,
The Entity-Relationship Model - Towards a Unified View of Data, ACM ToDS 1,1 (Mar. 1976), pp. 9-36

[Fal 76] FALKENBERG, E.,
[L/P 77] LACROIX, M. and PIROTTE, A.,
Domain-oriented Relational Languages, Proc. 4th Int.
Conf. on VLDB, Tokyo, 1977, pp. 370-378

[Cod 79] CODD, E.F.,
Extending the Relational Database Model to Capture
More Meaning, ACM ToDS 4,4 (Dec. 1979), pp. 397-434

[Osb 79] OSBORN, S.L.,
Towards a Universal Relation Interface, Proc. 6th Int.
Conf. on VLDB, Rio d. J., 1979, pp. 52-60

[Ken 79] KENT, W.,
The Entity Join, Proc. 6th Int. Conf. on VLDB, Rio d. J.,
1979, pp. 232-238

[Ull 80] ULLMAN, J.D.,

[M/U 80] MAIER, D. and ULLMAN, J.D.,
Maximal Objects and the Semantics of Universal Rela-
tion Databases, TR 80-018, SUNY Stony Brook, N.Y.,
(Nov. 1980)

[B/K 81] BEERI, C. and KORTH, H.F.,
Compatible Attributes in a Universal Relation, IBM

[Jac 82] JACOBS, B.E.,
On Database Logic, JACM 29, 2 (Apr. 1982), pp. 310-332