ORM Abstract Syntax and Semantics: normative specifications

ORM.net Proposed Recommendation

Version (draft): 29 February 2020
Editors: Enrico Franconi, Terry Halpin
Contributors: Jan Hidders, Alessandro Mosca, Francesco Sportelli

Abstract

Object-Role Modeling (ORM) is a rigorous approach to modeling and querying at the conceptual level the information semantics of arbitrary domains. This document defines an abstract syntax for ORM conceptual models together with its formal semantics. An ORM conceptual model comprises an ORM conceptual schema plus a population of (object and fact) instances. In addition to type and constraint declarations, an ORM schema may include derivation rules. The semantics of an ORM conceptual model is defined by transforming the model to first-order logic axioms, whose finite models denote the legal abstract information structures of the conceptual specification.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of the revisions of this technical report can be found in the ORM.net Technical Recommendations index at <https://gitlab.com/orm-syntax-and-semantics/orm-syntax-and-semantics-docs.git>.

This document is part of the ORM document suite. It is the central ORM normative abstract syntax and semantics specification and it formally defines the core ORM concepts. The companion document "ORM abstract syntax and semantics: non-normative glossary" summarizes the abstract syntax of the main graphical symbols used in ORM by means of examples. Both documents of the ORM document suite can be found at <https://gitlab.com/orm-syntax-and-semantics/orm-syntax-and-semantics-docs.git>.

This document is published on ORM.net as a Proposed Recommendation. If you wish to make comments regarding this document, please send them to <orm-semantics@googlegroups.com>, after having registered at <https://groups.google.com/group/orm-semantics>. All comments are welcome.

Once this document becomes an ORM.net Recommendation, it will be a stable document and may be used as reference material or cited from other documents. ORM.net's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of data models based on ORM or other fact-based modeling approaches.

Change History

None.

Introduction

Object-Role Modeling (ORM) is a conceptual approach to modeling and querying the information semantics of business domains in terms of the underlying facts of interest, where facts, constraints and derivation rules may be verbalized in language readily understood by non-technical users of those business domains. Unlike Entity-Relationship (ER) modeling and the Unified Modeling Language (UML) class diagrams, but similar to the Web Ontology Language (OWL), ORM treats all facts as relationships. How facts are grouped into structures (e.g. attribute-based entity types, classes, relation schemes, XML schemas) is considered a design level, implementation issue that is irrelevant to the capturing of essential business semantics. Avoiding attributes in the base model enhances semantic stability, populatability, and natural verbalization, facilitating communication with all stakeholders. For information modeling, fact-oriented graphical notations are typically far more expressive than those provided by other notations. ORM includes graphical and textual languages for modeling and querying information at the conceptual level, as well as procedures for designing conceptual models, transforming between different conceptual representations, forward engineering ORM schemas to implementation schemas (e.g. relational database schemas, ontologies, object-oriented schemas, XML schemas) and reverse engineering implementation schemas to ORM schemas.

The latest version of ORM (ORM 2) is thoroughly described in the 2008 book by T. Halpin and T. Morgan ("Information Modeling and Relational Databases", second edition, Morgan Kaufmann), and in the newer books by T. Halpin ("Object-Role Modeling Fundamentals", Technics Publications, 2015) and its companion ("Object-Role Modeling Workbook", Technics Publications, 2016) providing an up-to-date coverage of the latest enhancements to ORM and its conceptual schema design procedure. This document refers to the ORM 2 version of ORM. With respect to ORM 2, this version of the document does not include the specification of constraints with deontic modality; the specification of these features is postponed to a later version of the document.

ORM Conceptual Model

The semantics of an ORM conceptual model is given with a transformation to first-order logic formulas: the *finite* first-order models of the obtained first-order logical theory are the legal populations of the ORM conceptual specification.
After having introduced *ORM Conceptual Model Signatures* and the corresponding *First-Order Logic Signatures*, the following Sections describe the abstract syntax and semantics of ORM's main conceptual model constructs, namely declarations, constraints, and derivation rules. Each ORM conceptual model construct is described in a boxed text subdivided in three parts, including its syntactic expression in purple (based on generic terms from the ORM signature), possibly the syntactic constraints over the syntactic expression, and, in green, its corresponding semantics specified as a closed first-order logic formula or as a rewriting using other conceptual model constructs (denoted as a MACRO).

An ORM conceptual model is any set of well-formed declarations and constraints from a given ORM signature, such that it includes exactly one `FactType` declaration for each canonical predicate name appearing in the ORM conceptual model.

Naming conventions

An *ORM conceptual model* is formally composed, following precise syntactic rules, by declarations and constraints, built from terms of different syntactic categories (*object type names*, *predicate names, role identifiers, data elements)* taken from a *signature*.
The ORM graphical notation depicts a *fact type* as a left-to-right top-to-bottom ordered sequence of *role* boxes, each of which is attached to exactly one *object type* shape. A fact type is bijectively associated to a *canonical* predicate.
Fact types (resp. object types) appearing in the ORM graphical notation as distinct are associated in the signature to distinct *predicate names* (resp. *object type names*). A role is uniquely identified in ORM graphical notation by the fact type in which it appears together with its relative position (left-to-right top-to-bottom) within the fact type; each role is given in the signature the *role identifier* obtained by concatenating the canonical predicate name it belongs to and the relative position within it.
In this document we assume that the signature of an ORM conceptual model (e.g., the choice of the canonical predicate name associated to a fact type) has been specified without ambiguity. This document does not focus on the pre-logical or linguistic means necessary in order to get the formal signature.
Predicate readings, as normally introduced in the ORM methodology, are used simply for readability and compactness of display on the graphical notation of ORM schema diagrams, and are not necessarily identifying, since distinct fact types may have the same predicate reading. On the other hand, distinct predicate names in the signature identify necessarily distinct fact types. Alternate predicate readings for the same fact type (e.g., useful to verbalise differently a predicate with different role orderings) are all obviously denoting the same fact type, and therefore will be given the same predicate name in the signature.

ORM Conceptual Model Signature

An *ORM conceptual model signature* is composed by the elements $< \mathcal{T}, \mathcal{V}, \mathcal{P}, \mathcal{R}, \mathcal{D}, \beta, \mathcal{F}, \alpha >$ denoting the following:

$\mathcal{T}$	a finite countable set of *domain object type names*
$\mathcal{V}$	a set $\mathcal{V} \subseteq \mathcal{T}$ of *domain value type names*
$\mathcal{P}$	a finite countable set of *predicate names*
$\mathcal{R}$	a finite countable set of *role names*
$\mathcal{D}$	a finite countable set of *domain values*
β	a total function $\beta: \mathcal{V} \rightarrow 2^{\mathcal{D}}$, the *domain value type extension*
$\mathcal{F}$	A finite countable set of *value function names*
α	a total function $\alpha: \mathcal{P} \cup \mathcal{F} \rightarrow \mathbb{N}^+$ specifying the *predicate* or the *function arity*

We adopt the following syntactic conventions in the ORM conceptual model:

- the letters `h,i, j, k,l,m,n` denote positive integer numbers;
- the letters `p,q` denote integer numbers;
- `T` denotes a domain object type name $\in \mathcal{T}$;
- `V` denotes a domain value type name $\in \mathcal{V}$;
- `P` denotes a predicate name $\in \mathcal{P}$;
- `r` denotes a role name $\in \mathcal{R}$;
- `d` denotes a domain value $\in \mathcal{D}$;
- `f` denotes a value function name $\in \mathcal{F}$.

- `P.i` denotes the i-th *role identifier of* `P`, with $1 \leq i \leq \alpha(P)$;
- `?v` denotes a *variable symbol* within a derivation rule, with `v` $\in$ STRINGS.

First-Order Logic Signature

The *First-Order Logic (FOL) signature* of an ORM conceptual model reuses the same symbols from $\mathcal{T}, \mathcal{V}, \mathcal{P}, \mathcal{D}, \mathcal{F}$ of the ORM conceptual model signature, and it is composed by the elements $< \mathcal{T}, \mathcal{V}, \mathcal{P}, \mathcal{L}, \mathcal{D}, \mathcal{G}, \mathcal{F} >$ denoting the following:

$\mathcal{T}$	a finite countable set of *unary predicate symbols*
$\mathcal{V}$	a set $\mathcal{V} \subseteq \mathcal{T}$
$\mathcal{P}$	a finite countable set of *predicate symbols*, each $\mathtt{P} \in \mathcal{P}$ with arity $\alpha(\mathtt{P})$
$\mathcal{L}$	a family of injective and well-founded *objectification functions*, one for each $\mathtt{P} \in \mathcal{P}$ and with arity $\alpha(\mathtt{P})$
$\mathcal{D}$	a finite countable set of *constant symbols*
$\mathcal{G}$	A family of *domain value to data value injective functions* $\gamma_{\mathtt{V}}$ for each $\mathtt{V} \in \mathcal{V}$, such that for each $\mathtt{d} \in \beta(\mathtt{V})$ and $\mathtt{T} \in \mathcal{T}$, it holds that $\mathtt{V}(\mathtt{d})$ and $\forall x.\, T(x) \rightarrow x \neq \gamma_{\mathtt{V}}(\mathtt{d})$
$\mathcal{F}$	A family of *functions over data values*, namely with domain and range over the range of the function γ

We adopt the following syntactic conventions in the FOL formulas:

- the precedence of Boolean operators is: "~" > "&" > "∨" > "→";
- ℓ_P denotes a function $\in \mathcal{L}$ associated to the predicate $\mathtt{P} \in \mathcal{P}$ and with arity $\alpha(\mathtt{P})$;
- $\gamma_{\mathtt{V}}$ denotes a function $\in \mathcal{G}$ associated to the domain value type $\mathtt{V}$.

First-order Logic extensions needed for the semantics of ORM Conceptual Models

The translation of derivation rules is given in first-order logic extended with *lambda expressions*. In the specification as a first order logic formula of the semantics of a derivation rule containing a *PATH* expression, a *PATH* expression corresponds to an open first order formula with one free variable. Such an open formula is built inductively from the parse tree of the *PATH* expression using its grammar specification in a way similar to Montague grammars. The composition among steps in the induction makes use of *lambda expressions* and their application using variable bindings and substitutions: if φ is a formula with a free variable x, and t is a term, an application of the lambda calculus *β-reduction rule* $((\lambda x.\, \varphi)(t)) \dashrightarrow (\varphi_{[x/t]})$ replaces the occurrences of the bound variable x within the body φ of the lambda expression with the term t.

The translation of ring constraints is given in first-order logic extended with the *transitive closure* operator $(\cdot)^*$ over binary predicates. First-order logic extended with the transitive closure operator is strictly more expressive than first-order logic. The transitive closure of a binary predicate $\mathtt{P}$ can be expressed in first-order logic with least fixpoints as follows:
$\forall x'y'.\, \mathtt{P}^* x'y' \leftrightarrow \mathbf{lfp}_{\mathtt{Q},xy}(\mathtt{P}xy \vee (\exists z.\, \mathtt{Q}xz \,\&\, \mathtt{P}zy))x'y'$.

The translation of identification constraints is given in first-order logic extended with *well-founded* binary relations. A binary predicate $\mathtt{P}$ is well-founded, *well-founded*($\mathtt{P}$), if its interpretation contains no countable infinite descending chains: that is, in the interpretation of $\mathtt{P}$ there is no infinite sequence $a_0, a_1, a_2, \ldots$ of non necessarily distinct elements such that $\mathtt{P}\ a_n\ a_{n+1}$ for every natural number n.

Declarations

`FactType(P (`T_1 … $T_{\alpha(P)}$`))`	P does not appear as an AlternatePredicate
$\forall x_1 \ldots x_{\alpha(P)}.\, Px_1 \ldots x_{\alpha(P)} \rightarrow T_1x_1 \,\&\, \ldots \,\&\, T_{\alpha(P)}x_{\alpha(P)}$	

`AlternatePredicate(P, Pa (P.`i_1 … `P.`$i_{\alpha(P)}$`))`	$\mathtt{P} \neq \mathtt{P}_a$ $\alpha(\mathtt{P}) = \alpha(\mathtt{P}_a)$ $\{i_1 \ldots i_{\alpha(P)}\} = \{1 \ldots \alpha(P)\}$
(MACRO) Replace all occurrences of $\mathtt{P}_a.j$ in the ORM conceptual model with `P.`i_j, and then all occurrences of $\mathtt{P}_a$ with P	

`RoleNaming(P.i r)`	
(MACRO) Replace all occurrences of the role name **r** in the ORM conceptual model with the role identifier `P.i`	

Constraints

Mandatory(T $P_1.i_1$ … $P_m.i_m$)	for j≠k and j,k≤m: $P_j \neq P_k$
$\forall x.Tx \rightarrow \exists y_1 \dots y_{\alpha(P_1)}.\left(P_1 y_1 \dots y_{\alpha(P_1)} \,\&\, x = y_{i_1}\right) \vee \dots \vee \exists y_1 \dots y_{\alpha(P_m)}.\left(P_m y_1 \dots y_{\alpha(P_m)} \,\&\, x = y_{i_m}\right)$	

Unique(P.i_1 … P.i_m)	for j≠k and j,k≤m: $i_j \neq i_k$
(MACRO) Frequency(P.i_1 … P.i_m (1, 1))	

Identification(T P.i_{m+1} (P.i_1 … P.i_m))	for j≠k and j,k≤m+1: $i_j \neq i_k$
(MACRO) Unique(P.i_{m+1}) Mandatory(T (P.i_{m+1})) Unique(P.i_1 … P.i_m) FactTypeRule(B_1 (P.i_{m+1} ➤ (P.i_1 ⋈ ?x)) (?x)) … FactTypeRule(B_m (P.i_{m+1} ➤ (P.i_m ⋈ ?x)) (?x)) *well-founded*(B_1) … *well-founded*(B_m)	

ExternalUnique($P_1.i_1$ … $P_m.i_m$)	for j≠k and j,k≤m: α(P_j)=2 , $P_j \neq P_k$, if i_j=1 then l_j=2 else l_j=1; and P fresh predicate name of arity m+1
(MACRO) FactTypeRule(P ($P_1.i_1$ ➤ ($P_1.l_1$ ⋈ ?x)) … ($P_m.i_m$ ➤ ($P_m.l_m$ ⋈ ?x)) (?x)) Unique(P.1 … P.m)	

ExternalIdentification(T ($P_1.i_1$ … $P_m.i_m$))	for j≠k and j,k≤m+1: α(P_j)=2 , $P_j \neq P_k$, if i_j=1 then l_j=2 else l_j=1; and P fresh predicate name of arity m+1
(MACRO) FactTypeRule(P ($P_1.i_1$ ➤ ($P_1.l_1$ ⋈ ?x)) … ($P_m.i_m$ ➤ ($P_m.l_m$ ⋈ ?x)) (?x)) Unique(P.1 … P.m) Unique(P.m+1) Mandatory(T (P.m+1)) *well-founded*(P_1) … *well-founded*(P_m)	

Frequency(P.i_1 … P.i_m $\underline{F}$)	for j≠k and j,k≤m: $i_j \neq i_k$. p,q≥1 (1) $\underline{F}$:=(p..) (2) $\underline{F}$:=(..q) (3) $\underline{F}$:=(p..q) (4) $\underline{F}$:=(p)
(1) $\left(\forall x_1 \dots x_{\alpha(P)}.Px_1 \dots x_{\alpha(P)} \rightarrow \exists^{\geq p} y_1 \dots y_{\alpha(P)}.Py_1 \dots y_{\alpha(P)} \,\&\, x_{i_1} = y_{i_1} \,\&\, \dots \,\&\, x_{i_m} = y_{i_m}\right)$ (2) $\left(\forall x_1 \dots x_{\alpha(P)}.Px_1 \dots x_{\alpha(P)} \rightarrow \exists^{\leq q} y_1 \dots y_{\alpha(P)}.Py_1 \dots y_{\alpha(P)} \,\&\, x_{i_1} = y_{i_1} \,\&\, \dots \,\&\, x_{i_m} = y_{i_m}\right)$ (3) $\left(\forall x_1 \dots x_{\alpha(P)}.Px_1 \dots x_{\alpha(P)} \rightarrow \exists^{\geq p} y_1 \dots y_{\alpha(P)}.Py_1 \dots y_{\alpha(P)} \,\&\, x_{i_1} = y_{i_1} \,\&\, \dots \,\&\, x_{i_m} = y_{i_m}\right) \,\&$ $\left(\forall x_1 \dots x_{\alpha(P)}.Px_1 \dots x_{\alpha(P)} \rightarrow \exists^{\leq q} y_1 \dots y_{\alpha(P)}.Py_1 \dots y_{\alpha(P)} \,\&\, x_{i_1} = y_{i_1} \,\&\, \dots \,\&\, x_{i_m} = y_{i_m}\right)$ (4) $\left(\forall x_1 \dots x_{\alpha(P)}.Px_1 \dots x_{\alpha(P)} \rightarrow \exists^{=p} y_1 \dots y_{\alpha(P)}.Py_1 \dots y_{\alpha(P)} \,\&\, x_{i_1} = y_{i_1} \,\&\, \dots \,\&\, x_{i_m} = y_{i_m}\right)$	

ExternalFrequency($P_1.i_1$ … $P_m.i_m$ $\underline{F}$)	for j≠k and j,k≤m: α(P_j)=2 , $P_j \neq P_k$, if i_j=1 then l_j=2 else l_j=1; and P fresh predicate name.

	p,q≥1 (1) $\underline{F}$ =(p..) (2) $\underline{F}$ =(..q) (3) $\underline{F}$ =(p..q) (4) $\underline{F}$ =(p)
(MACRO) FactTypeRule(P (P_1.i_1 ➤ (P_1.l_1 ⋈ ?x)) … (P_m.i_m ➤ (P_m.l_m ⋈ ?x)) (?x)) Frequency(P.l … P.m $\underline{F}$)	

Subtype((T_1 … T_m) T)	
$(\forall x.T_1x \rightarrow Tx) \,\&\, \ldots \,\&\, (\forall x.T_mx \rightarrow Tx)$	

ExclusiveSubtypes((T_1 … T_m) T)	
$(\forall x.T_1x \rightarrow Tx \,\&\, {\sim}T_2x \,\&\, \ldots \,\&\, {\sim}T_mx) \,\&\, \ldots \,\&\, (\forall x.T_{m-1}x \rightarrow Tx \,\&\, {\sim}T_mx) \,\&\, (\forall x.T_mx \rightarrow Tx)$	

ExhaustiveSubtypes((T_1 … T_m) T)	
$\big((\forall x.T_1x \rightarrow Tx) \,\&\, \ldots \,\&\, (\forall x.T_mx \rightarrow Tx)\big) \wedge (\forall x.Tx \rightarrow T_1x \vee \ldots \vee T_mx)$	

Subset((P_1.i_1 P_2.h_1) … (P_1.i_m P_2.h_m))	$P_1 \neq P_2$ and for j≠k and j,k≤m: P_1.$i_j \neq P_1$.i_k and P_2.$h_j \neq P_2$.h_k
$\forall x_1 \ldots x_{\alpha(P_1)}.P_1x_1 \ldots x_{\alpha(P_1)} \rightarrow \exists y_1 \ldots y_{\alpha(P_2)}.P_2y_1 \ldots y_{\alpha(P_2)} \,\&\, x_{i_1} = y_{h_1} \,\&\, \ldots \,\&\, x_{i_m} = y_{h_m}$	

Exclusive((P_1.i_1 P_2.h_1) … (P_1.i_m P_2.h_m))	$P_1 \neq P_2$ and for j≠k and j,k≤m: P_1.$i_j \neq P_1$.i_k and P_2.$h_j \neq P_2$.h_k
$\forall x_1 \ldots x_{\alpha(P_1)}.P_1x_1 \ldots x_{\alpha(P_1)} \rightarrow \exists y_1 \ldots y_{\alpha(P_2)}.{\sim}P_2y_1 \ldots y_{\alpha(P_2)} \,\&\, x_{i_1} = y_{h_1} \,\&\, \ldots \,\&\, x_{i_m} = y_{h_m}$	

Equal((P_1.i_1 P_2.h_1) … (P_1.i_m P_2.h_m))	$P_1 \neq P_2$ and for j≠k and j,k≤m: P_1.$i_j \neq P_1$.i_k and P_2.$h_j \neq P_2$.h_k
(MACRO) Subset((P_1.i_1 P_2.h_1) … (P_1.i_m P_2.h_m)) Subset((P_2.h_1 P_1.i_1) … (P_2.h_m P_1.i_m))	

TypeCardinality(T (p, q))	p,q≥0 and q possibly ∞
$\exists^{p..q}x.Tx$	

RoleCardinality(P.i (p, q))	p,q≥0 and q possibly ∞
$\forall x_1 \ldots x_n.\exists^{p..q}y.Px_1 \ldots x_n \,\&\, x_i = y$	

Objectifies(T P)	
$\forall x_1 \ldots x_n.Px_1 \ldots x_n \leftrightarrow T\big(\ell_P(x_1 \ldots x_n)\big)$	

ValuesOf(V (d₁ … dₘ))	
$\forall x. \mathrm{V}x \rightarrow (x = d_1) \vee \ldots \vee (x = d_m)$	

ValuesOf(P.i (d₁ … dₘ))	
$\forall x_1 \ldots x_n. Px_1 \ldots x_{\alpha(P)} \rightarrow (x_i = d_1) \vee \ldots \vee (x_i = d_m)$	

≶(P.i₁ P.i₂)	≶ ::=< | ≤ | > | ≥ | = | ≠ and V_1 is the value type associated to P.i₁ and V_2 is the value type associated to P.i₂
$\forall x_1 \ldots x_{\alpha(P)} y_1 \ldots y_{\alpha(P)}. Px_1 \ldots x_{\alpha(P)} \,\&\, Py_1 \ldots y_{\alpha(P)} \rightarrow \gamma_{V_1}(x_{i_i}) \lessgtr \gamma_{V_2}(y_{i_2})$	

RingConstraint(P.i P.j)	i≠j and P_b fresh predicate name with $\alpha(P_b) = 2$

(MACRO)
JoinPath(Pb (P.i P₁.j))

- LocallyReflexive
  - $\forall x_1 x_2.\ \mathrm{P_b} x_1 x_2 \rightarrow \mathrm{P_b} x_1 x_1$
- PurelyReflexive
  - $\forall x_1 x_2.\ \mathrm{P_b} x_1 x_2 \rightarrow x_1 = x_2$
- Irreflexive
  - $\forall x.\ \sim \mathrm{P_b} xx$
- Symmetric
  - $\forall x_1 x_2.\ \mathrm{P_b} x_1 x_2 \rightarrow \mathrm{P_b} x_2 x_1$
- Asymmetric
  - $\forall x_1 x_2.\ \mathrm{P_b} x_1 x_2 \rightarrow \sim \mathrm{P_b} x_2 x_1$
- Antisymmetric
  - $\forall x_1 x_2.\ \mathrm{P_b} x_1 x_2 \,\&\, x_1 \neq x_2 \rightarrow \sim \mathrm{P_b} x_2 x_1$
- Transitive
  - $\forall x_1 x_2 y_1 y_2. \mathrm{P_b} x_1 x_2 \,\&\, \mathrm{P_b} y_1 y_2 \,\&\, x_2 = y_1 \rightarrow \mathrm{P_b} x_1 y_2$
- Intransitive
  - $\forall x_1 x_2 y_1 y_2. \mathrm{P_b} x_1 x_2 \,\&\, \mathrm{P_b} y_1 y_2 \,\&\, x_2 = y_1 \rightarrow \sim \mathrm{P_b} x_1 y_2$
- StronglyIntransitive
  - $\forall x_1 x_2 y_1 y_2. \mathrm{P_b} x_1 x_2 \,\&\, \mathrm{P_b}^* y_1 y_2 \,\&\, x_2 = y_1 \rightarrow \sim \mathrm{P_b} x_1 y_2$
- Acyclic
  - $\forall x.\ \sim \mathrm{P_b}^* xx$

Derivation Rules

SubTypeRule(T PATH)	
$\forall x.Tx \leftrightarrow \exists var_1 \dots var_m.\,\mathrm{PATH}^{\mapsto}\ x$	$\{var_1 \dots var_m\}$ includes all the ?VAR variable symbols in $\mathrm{PATH}^{\mapsto}$

FactTypeRule(P PATH_1 … $\mathrm{PATH}_{\alpha(P)}$)	
$\forall x_1 \dots x_{\alpha(P)}.Px_1 \dots x_{\alpha(P)} \leftrightarrow \exists var_1 \dots var_m.\,\mathrm{PATH}_1^{\mapsto}\ x_1\ \&\ \dots\ \&\ \mathrm{PATH}_{\alpha(P)}^{\mapsto}\ x_{\alpha(P)}$	$\{var_1 \dots var_m\}$ includes the ?VAR variable symbols in $\mathrm{PATH}_1^{\mapsto}, \dots, \mathrm{PATH}_{\alpha(P)}^{\mapsto}$

JoinPath(P (P₁.i₁ P₁.j₁) … (Pₘ.iₘ Pₘ.jₘ))	$\alpha(P) = 2$ and for k≤m: $i_k \neq j_k$
(MACRO) FactTypeRule(P (P₁.i₁ ➤ (P₁.j₁ ⋈ (P₂.i₂ ➤ (P₂.j₂ ⋈ (… (Pₘ.iₘ ➤ (Pₘ.jₘ ⋈ ?x))))))) (?x))	

PATH ::= T | Pᵤ | P.i ➤ [P.i₁ ⋈ PATH₁] … [P.iₘ ⋈ PATHₘ] | PATH₁ ∨ PATH₂ | PATH₁ ∧ PATH₂ | PATH₁ \ PATH₂ | {d₁ … dₙ} | ?VAR | V ≦ TERM TERM ::= d | ?VAR | f(TERM₁ … TERM$_{\alpha(f)}$)	≦ ::= < | ≤ | > | ≥ | = | ≠ $\alpha(P_u) = 1$ for j≠k and j,k≤m: $i_j \neq i_k$

$$
\begin{aligned}
\mathtt{T} &\mapsto \lambda x.Tx \\
\mathtt{P_u} &\mapsto \lambda x.P_u x \\
\mathtt{P.i} \blacktriangleright [\mathtt{P.i_1} \bowtie \mathrm{PATH}_1] \dots [\mathtt{P.i_m} \bowtie \mathrm{PATH}_m] &\mapsto \lambda x.\exists x_1 \dots x_{\alpha(P)}.Px_1 \dots x_{\alpha(P)} \wedge \mathrm{PATH}_1^{\mapsto}\ x_{i_1}\ \&\ \dots\ \&\ \mathrm{PATH}_m^{\mapsto}\ x_{i_m} \wedge x = x_i \\
\mathrm{PATH}_1 \wedge \mathrm{PATH}_2 &\mapsto \lambda x.\mathrm{PATH}_1^{\mapsto}\ x\ \&\ \mathrm{PATH}_2^{\mapsto}\ x \\
\mathrm{PATH}_1 \vee \mathrm{PATH}_2 &\mapsto \lambda x.\mathrm{PATH}_1^{\mapsto}\ x \vee \mathrm{PATH}_2^{\mapsto}\ x \\
\mathrm{PATH}_1 \setminus \mathrm{PATH}_2 &\mapsto \lambda x.\mathrm{PATH}_1^{\mapsto}\ x\ \&\ {\sim}\mathrm{PATH}_2^{\mapsto}\ x \\
\{\mathtt{d_1} \dots \mathtt{d_n}\} &\mapsto \lambda x.x = \mathtt{d_1} \vee \dots \vee x = \mathtt{d_n} \\
\mathtt{?VAR} &\mapsto \lambda x.x = \mathtt{?VAR} \\
\mathtt{V} \leqq \mathrm{TERM} &\mapsto \lambda x.\mathtt{V}x \wedge \gamma_{\mathtt{V}}(x) \leqq \gamma_{\mathtt{V}}(\mathrm{TERM}^{\hookrightarrow}) \\
\mathtt{d} &\hookrightarrow \mathtt{d} \\
\mathtt{?VAR} &\hookrightarrow \mathtt{?VAR} \\
\mathtt{f}(\mathrm{TERM}_1 \dots \mathrm{TERM}_{\alpha^{\mathcal{F}}(f)}) &\hookrightarrow \mathtt{f}(\mathrm{TERM}_1^{\hookrightarrow} \dots \mathrm{TERM}_{\alpha^{\mathcal{F}}(f)}^{\hookrightarrow})
\end{aligned}
$$