University of Applied Sciences Northwestern Switzerland
School of Business

Semantic and Logical Foundations for Business Vocabulary and Rules

http://www.omg.org/spec/SBVR/1.0

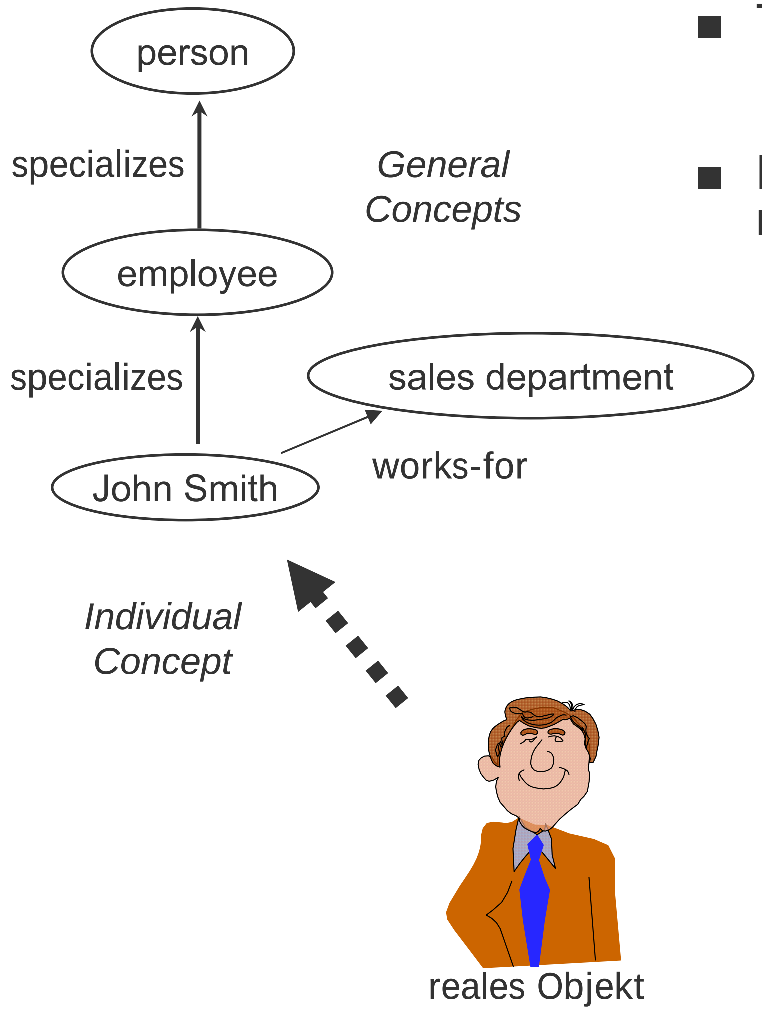
SBVR Clause 10: Semantic and Logical Foundations

- The SBVR initiative is intended to capture business facts and business rules that may be expressed either informally or formally.
  - Formal statements of rules may be transformed into logical formulations that are used for exchange with other rule-based software tools.
  - Informal statements of rules may be exchanged as uninterpreted comments.
- Business rule expressions are classified as formal only if they are expressed purely in terms of
  - fact types in the pre-declared schema for the business domain, and
  - logical/ mathematical operators, quantifiers, etc.
- The following discussion of business rule semantics is confined to formal statements of business rules.

SBVR: Conceptual Schema

- For any given business, the “universe of discourse” indicates those aspects of the business that are of interest.
- A “model,” in the sense used here, is a structure intended to describe a business domain, and is composed of
  - a conceptual *schema* (fact structure) and
  - a *population* of ground facts
- A *fact* is a proposition taken to be true by the business.
  - Instantiated roles of facts refer to individuals (such as “Employee 123”, "John Smith" or “the sales department”).
  - Individuals are considered as being of a particular type (such as “Employee” or “Department”) where *type* denotes “set of possible individuals.”
- The conceptual schema declares the *fact types* (kinds of facts, such as “Employee works for Department”) and *rules* relevant to the business domain.

The Conceptual Schema as a Semantic Net



- Two types of concepts:
  - **general concept** (class)
  - **individual concept** (instance)
- From this distinction we have at least three kinds of relations (binary **fact types**)
  - structural relations:
    - Relation between individual and general concepts (also called instance-of)

      John Smith ***specializes*** **employee**
    - Relation between general concepts (also called is-a or SubClass-Of)

      employee ***specializes*** **person**

    Unfortunately, SBVR uses the same name for both kinds of structural relations
  - non-structural relations
    - arbitrary relations, e.g.
      John Smith ***works-for*** **sales department**

Conceptual Schema in Predicate Logic

- Concepts:
  - The general concepts correspond to unary predicates
  - The individual concepts correspond to constants
- Structural Relations
  - Instance-of is an atomic formula with the general concept as Predicate and the individual concept as term

    Employee(john_smith) — <u>John Smith</u> ***specializes*** <u>employee</u>

  - Subclass relationship corresponds to an implication

    $\forall x\ \text{Employee}(x) \rightarrow \text{Person}(x)$ — <u>employee</u> ***specializes*** <u>person</u>

  - Binary fact types correspond to binary predicates

    Works_for(john_smith, sales_department)

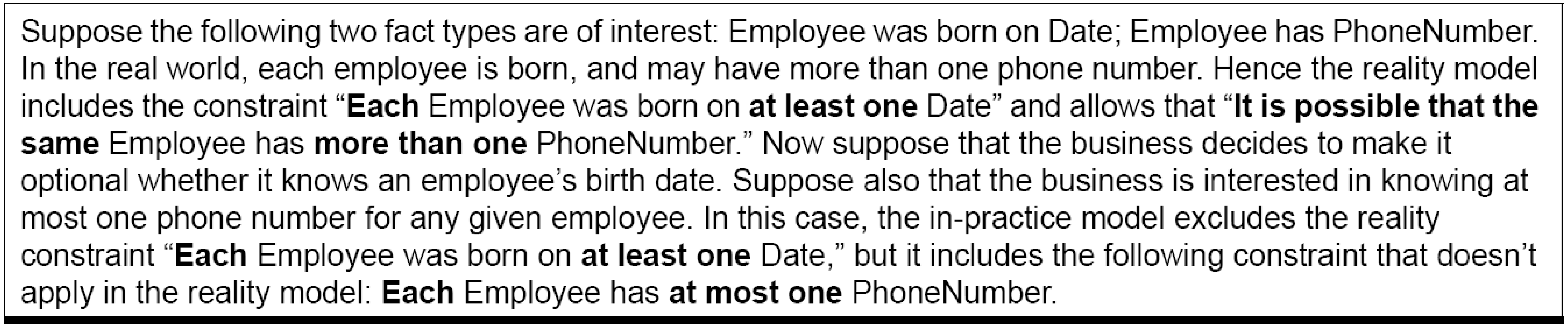
    <u>John Smith</u> ***works-for*** <u>sales department</u>

Facts

- *Facts* are either elementary or existential.
  - Elementary fact: declaration that an individual has a property
    - Example: Country(australia) $\wedge$ Large(australia)
      "The Country named 'Australia' is large "
  - Existential fact: assert the existence of an individual
    - Example: $\exists$x Country(x) $\wedge$ Country_code(x, us)
      "There is a Country that has the Country Code 'US' "

Reality model vs. in-practice model

- A *reality model* of a business domain is intended to reflect the constraints that actually apply to the business domain in the real world.
- An *in-practice model* of a business domain reflects the constraints that the business chooses in practice to impose on its knowledge of the business domain.



Suppose the following two fact types are of interest: Employee was born on Date; Employee has PhoneNumber. In the real world, each employee is born, and may have more than one phone number. Hence the reality model includes the constraint "**Each** Employee was born on **at least one** Date" and allows that "**It is possible that the same** Employee has **more than one** PhoneNumber." Now suppose that the business decides to make it optional whether it knows an employee's birth date. Suppose also that the business is interested in knowing at most one phone number for any given employee. In this case, the in-practice model excludes the reality constraint "**Each** Employee was born on **at least one** Date," but it includes the following constraint that doesn't apply in the reality model: **Each** Employee has **at most one** PhoneNumber.

Static Constraints

- *Constraints* are used to define bounds, borders, or limits on fact populations, and may be static or dynamic.
- A *static constraint* imposes a restriction on what fact populations are possible or permitted, for each fact population taken individually.
- Example:
  - $\forall x \; \exists y \; Employee(x) \wedge Date(y) \wedge born\_on(x,y)$
  - **Each** Employee has a date of birth

Derivation Rules / Inference Rules

- *Derivation rules* indicate how the population of a fact type may be derived from the populations of one or more fact types or how a type of an individual may be defined in terms of other types of individuals and fact types.
- Example 1:
  - Person1 is an uncle of Person2 **if** Person1 is a brother of **some** Person3 **who** is a parent of Person2,

$\forall x,y,z\ \ \text{Brother}(x,y) \wedge \text{Parent}(y,z) \rightarrow \text{Uncle}(x,z)$

- Example 2:
  - **Each** person **is a** employee **if the** person works for **a** company

$\forall x,y\ \ \text{Person}(x) \wedge \text{Company}(y) \wedge \text{Works\_for}(x,y) \rightarrow \text{Employee}(x)$

Modalities

- Structural and operational rules
- Rule enforcement

Modalities

- In SBVR every constraint has an associated modality
- Alethic modality – Structural Rules

□	necessity
◊	possibility

- Deontic modality – Operative Rules

O	obligation
P	permission
F	forbidden

Transformation Rules for Alethic Modalities

Possibility statements can be transformed into necessity statements (and vice versa) using the following transformation rules:

Modality		Modal Formula		applying modal negation rules ... = (Logically Equivalent) Modal Formula	
		Formula	Reading (Verbalized as):	Formula	Reading (Verbalized as):
alethic	necessity	$\Box p$	It is necessary that *p*	$\sim\Diamond\sim p$	It is not possible that not *p*
	the negation of necessity: **non-necessity**	$\sim\Box p$	It is not necessary that *p*	$\Diamond\sim p$	It is possible that not *p*
	possibility	$\Diamond p$	It is possible that *p*	$\sim\Box\sim p$	It is not necessary that not *p*
	the negation of possibility: impossibility	$\sim\Diamond p$	It is not possible that *p* It is impossible that *p*	$\Box\sim p$	It is necessary that not *p*
	contingency	$\Diamond p$ & $\sim\Box p$	It is possible but not necessary that *p*	$\sim(\sim\Diamond p \vee \Box p)$	It is neither impossible nor necessary that *p*

Other transformation rules: $\forall x \Box Fx \equiv \Box \forall x Fx$

$\exists x \Diamond Fx \equiv \Diamond \exists x Fx$

Transformation Rules for Deontic Modalities

Obligatory statements can be transformed into prohibition statements (and vice versa) using the following transformation rules:

Modality		Modal Formula		applying modal negation rules ... = (Logically Equivalent) Modal Formula	
		Formula	Reading (Verbalized as):	Formula	Reading (Verbalized as):
deontic	obligation	**O**p	It is obligatory that *p*	~***P***~*p*	It is not permitted that not *p*
	the negation of obligation: non-obligation	~**O**p	It is not obligatory that *p*	***P***~*p*	It is permitted that not *p*
	permission	**P**p	It is permitted that *p*	~***O***~*p*	It is not obligatory that not *p*
	the negation of permission: prohibition	~**P**p **F**p	It is not permitted that *p* It is prohibited that *p* It is forbidden that *p*	***O***~*p*	It is obligatory that not *p*
	optionality	***P****p* & ~***O****p*	It is permitted but not obligatory that *p*	~ (~***P****p* v ***O****p*)	It is neither prohibited nor obligatory that *p*

Modalities and Predicate Logic

- Rules with modalities can easily be represented in predicate logic by „tagging“
- Rules usually have just one modal operator.
- These rules can be transformed so that the modal operator is moved to the front using negation and tranformation rules
  - The rule (without modality operator) is represented in predicate logic
  - The rule is "tagged" with the modality of the main operator

Interpretation of Alethic Modality

- If no modality is explicitly specified, an alethic modality of necessity is often assumed:

  C1 Each **Person** ***was born in*** at most one **Country**

- may be explicitly verbalized with an alethic modality

  C1' It is necessary that each **Person** ***was born in*** at most one **Country**

- For the model theory, we omit the necessity operator from the formula. The version without modal operator can be represented in standard predicate logic

Represent the following rule in predicate logic

It is necessary that a person *that rents* a car *has* a driver license

Modalities and Rule Enforcement

- Simply tagging rules with the modality might not be enough because modalities are important for rule enforcement.
- The tagging of a rule as a necessity or obligation impacts the rule enforcement policy.
  - Necessity rules (alethic modality) do not need enforcement – what is derived is valid.
  - Enforcement of an obligation rule (deontic modality) should allow states that do not satisfy the obligation rule

Deontic Modality in Predicate Logic

- To distinguish rules that need enforcement, we can represent the deontic modality operator explicitly as a special predicate
- To do this, we transform rules with deontic operators into a form using the „forbidden" modality

  It is obligatory that each <u>person</u> ***that drives*** **a** <u>car</u> ***is older than*** <u>18 years.</u>

  or It is forbidden that a **<u>Person</u>** ***that drives*** **a** <u>car</u> ***is younger than*** <u>18 years</u>
- Deontic Modality can be represented in Predicate Logic:
  1. Normalize the formula by moving the modal operator to the front
  2. Replace the modal operators by a special predicate (e.g. forbidden).
- Example: It is forbidden that a **<u>car driver</u>** ***is less than*** **<u>18 years</u>**

  can be represented as

  $$\forall x\ \forall y\ (Car\_driver(x) \wedge Age(x,y) \wedge y < 18\ \rightarrow forbidden$$
- In the Structured Englisch verbalization, forbidden is a modal operator while in the logic representation forbidden is a predicate. This predicates is treated like any other predicate, except that it has a reserved name.

Open and Closed World

- Negation
- Incomplete Information

Open/Closed World Semantics

Dealing with missing Information

- The closed world assumption (CWA) is the presumption that what is not currently known to be true is false.
  - Under the CAW, if a proposition cannot be proved true, it is false.
- The open world assumption (OWA) states that lack of knowledge does not imply falsity.
  - Under the OWA, if a proposition cannot be proved true and its negation cannot be proved true, the truth of the proposition is *unknown*

Database Example for Open/Closed World

Suppose we have the following sample database with the employee number and name of each employee, as well as the cars they drive (if any):

Does it include all employees?

Employee	
empNr	*empName*
1	John Smith
2	Ann Jones
3	John Smith

Drives	
empNr	*carRegNr*
1	ABC123
2	AAA246
2	DEF001

Does it include all drivers?

- Users typically adopt the closed world assumption when interpreting data in databases.
  - If a data is not in the database, it is assume to be false
- Example: Select employee number of each employee who does not drive a car

  select empNr **from** Employee **where** empNr **not in** (**select** empNr **from** Drives).
- Correctness of the result depends on whether all employees with their car registry are in the database.

Open and Closed World in a Business Domain

- The distinction between open and closed world assumption can be made on the level of the fact model, based on different criteria:
- **Domain of interest:** In modeling any given business domain, attention can be restricted to propositions of interest to that domain. If a proposition is not relevant to that domain, it is not included as a fact there.
  - In this case we do not assume missing information as false; rather we simply dismiss it from consideration.
  - Example: We decide what information about customers shall be stored in a CRM system. We can decide not to store information about his/her marital status.
- **Incomplete information:** In a given business domain we might be unable to collect all information.
  - In this case, missing information does not imply falsity.
  - Example: Assume we collect the phone number of our customers in a CRM system. If we do not find the phone number of a customer, it does not mean, that he does not have a phone number

Open/Closed World and Negation

- The open or closed world semantics is important for negation
  - Closed World (CWA): a failure to find a fact implies its negation
    → **negation as failure**
  - Open World (OWA): lack of knowledge does not imply falsity. A proposition is false only if its negation can be proved.
    → **full negation**
- Example:
  - If the customer did not pay his goods he is reminded.

Open or closed world?

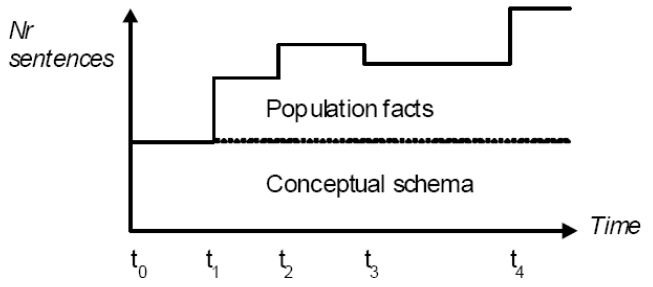
- A business might have complete knowledge about some parts and incomplete knowledge about other parts
- Thus, in practice a mixture of open and close world assumption may applied
- To cope with this situation, one might, for example,
  - assume open world semantic by default and
  - apply local closure to specific parts

  (or vice versa)
- Local closure means that for some parts of the overall DB schema the closed world assumption applies.
- Local closure can be asserted explicitly for individual and fact types, e.g.
  - **employee** ***is closed*** (all employees are known)
  - **has-name** ***is closed*** (all names of employees are known).

Adding and Deleting Facts

Fact Population

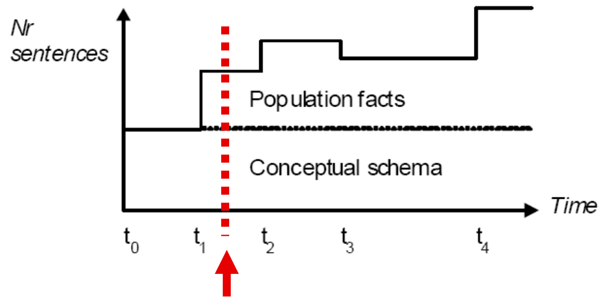
- Rules are applied to facts about the domain (i.e. data about customers, empoyees, etc.
- The fact model includes both
  - the conceptual schema and
  - the ground fact population
    (set of fact instances that instantiate the fact types in the schema)
- In contrast to the conceptual schema, the (domain-specific) fact population is typically highly variable.
- In treating a fact model as a set of facts that typically changes over time, we allow facts to be added or deleted



Constraints and Changing Fact Populations

Static Constraints **and** ***Derivation Rules*** **are applied at a single state**

- If the fact model changes there is a new state, for which the constraints and derivation rules are applied again without regard of previous states.

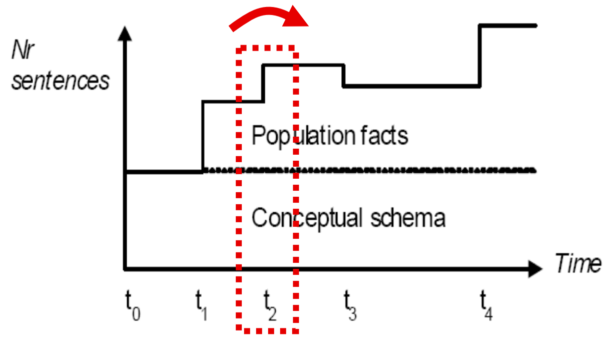


- Example:

  Assume that customers get a discount if their shopping exceeds 1'000 Fr. within 12 months. The calculation of the discount changes as soon as a shopping is made such that 1'000 Fr. are reached and may be reduced again if the customer does not buy enough within 12 months.

A ***dynamic constraint*** **imposes a restriction on transitions between states of fact populations.**

- Dynamic constraints compare one state to another state.



- Example:

  A person's marital status may change from single to married, but not from divorced to single

(The semantics of dynamic constraints is not defined in SBVR 1.0)

Adding and Deleting Facts

- If we allow deletion or changes of knowledge, it may occur that previous inferences become invalid.
  - We might delete a fact
    - because we revise our decision on whether it is true (correcting a failure)
    - or because we decide that a fact is no longer of interest
- With Negation as Failure (Closed World), also adding facts may make previous inferences invalid
  - Example: Every employee has a date of birth

    $\forall x \, \forall y \; Employee(x) \wedge Date(y) \wedge not\,(born\_on(x,y)) \rightarrow forbidden$

    If for an employee there is no date of birth in the database the constraint is violated (i.e. „forbidden“ is derived). If the date of birth is added, the rule should not not be applicable anymore.
- Making previous derivations invalid is called non-monotonic. To avoid this problem, static constraints are applied to individual state of the fact model.

Additional Quantifiers

Quantifiers in SBVR

In addition to ∀ and ∃ SBVR supports numeric quantifiers:

Symbol	*Example*	*Name*	*Meaning*
$\forall$	$\forall x$	Universal Quantifier	For each and every *x*, taken one at a time
$\exists$	$\exists x$	Existential Quantifier	At least one *x*
$\exists^1$	$\exists^1 x$	Exactly-one quantifier	There is exactly one (at least one and at most one) *x*
$\exists^{0..1}$	$\exists^{0..1} x$	At-most-one quantifier	There is at most one *x*
$\exists^{0..n}$ ($n \geq 1$)	$\exists^{0..2} x$	At-most-*n* quantifier	There is at most *n x* *Note:* *n* is always instantiated by a number ≥ 1. So this is really a set of quantifiers (*n* = 1, etc.)
$\exists^{n..}$ ($n \geq 1$)	$\exists^{2..} x$	At-least-*n* quantifier	There is at least *n x* *Note:* *n* is always instantiated by a number ≥ 1. So this is really a set of quantifiers (*n* = 1, etc.)
$\exists^{n}$ ($n \geq 1$)	$\exists^{2} x$	Exactly-*n* quantifier	There is at exactly (at least and at most) *n x* *Note:* *n* is always instantiated by a number ≥ 1. So this is really a set of quantifiers (*n* = 1, etc.)
$\exists^{n..m}$ ($n \geq 1, m \geq 2$)	$\exists^{2..5} x$	Numeric range quantifier	There is at least *n* and at most *m x*

Definition of Additional Quantifiers

- The additional existential quantifiers can easily be defined in terms of the standard quantifiers
- Example:

$$\forall y\ \exists^2 x\ \ \text{Parent}(x,y)$$

is equivalent to

$$\forall y\ \exists x_1\ \exists x_2\ \ (\text{Parent}(x_1,y) \land \text{Parent}(x_2,y) \land x_1 \neq x_2 \land \forall x\ (\text{Parent}(x,y) \rightarrow (x = x_1 \lor x = x_2)))$$