

Logics, Categories and Colimits  
Lecture Notes  
DRAFT

A lecture by Till Mossakowski  
Notes taken by Florian Pommerening, Tilman Thiry and Robert Mattmüller



# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Logics</b>	<b>7</b>
2.1	Propositional Logics . . . . .	7
2.1.1	Foundations . . . . .	7
2.1.2	Proofs for Propositional Logic . . . . .	13
2.1.3	Conservative Extensions . . . . .	18
2.1.4	Freeness . . . . .	23
2.2	Description Logics . . . . .	28
2.2.1	Foundations . . . . .	28
2.2.2	Extensions of $\mathcal{ALC}$ . . . . .	33
2.2.3	Signature morphisms . . . . .	34
2.2.4	Freeness . . . . .	35
2.2.5	Conservative Extensions . . . . .	39
2.3	First-Order Logic . . . . .	43
2.3.1	Foundations . . . . .	43
2.3.2	Signature Morphisms . . . . .	45
2.3.3	Conservative Extensions . . . . .	47
2.3.4	Sort generation constraints . . . . .	49
2.3.5	Proofs . . . . .	50
<b>3</b>	<b>Category theory</b>	<b>51</b>
3.1	Satisfaction Systems . . . . .	51
3.2	Categories . . . . .	53
3.2.1	Functors . . . . .	57
3.2.2	Institutions . . . . .	58
3.2.3	Structured specifications . . . . .	61
3.2.4	Institutions with proofs . . . . .	65
3.3	Colimits . . . . .	66
3.3.1	Coproducts . . . . .	66
3.3.2	Semi-exactness . . . . .	69
3.3.3	Colimits in general . . . . .	75
3.4	Natural transformations . . . . .	77

3.4.1	Institution comorphisms . . . . .	79
3.5	Borrowing . . . . .	81
3.5.1	Borrowing for structured specifications . . . . .	83
3.6	Free specifications . . . . .	85
3.6.1	Model homomorphisms . . . . .	85
3.7	Adjoint functors . . . . .	89
<b>4</b>	<b>Outlook</b>	<b>93</b>
4.1	Modal logic . . . . .	93
4.1.1	Correspondence theory . . . . .	93
4.2	Coalgebraic logic . . . . .	94
4.3	Higher-order logic . . . . .	94
4.4	Substructural logics . . . . .	95
4.4.1	Linear logic . . . . .	95
4.4.2	Paraconsistent logic . . . . .	95
4.5	Institutional model theory . . . . .	95
4.5.1	Logic programing . . . . .	95
4.6	Heterogenous specifications . . . . .	96
<b>5</b>	<b>Have fun</b>	<b>97</b>

# Chapter 1

## Introduction

This course material has been initiated during the course *Logics, Categories, and Colimits for Artificial Intelligence* given by Till Mossakowski in 2008/09 at the university of Freiburg, and has also been used for the course *Logics and categories for software engineering and artificial intelligence* given by Till Mossakowski and Lutz Schröder at the university of Bremen in 2009. It provides material for one semester course and contains numerous exercises for deepening the understanding. Worked-out solutions are available for all exercises<sup>1</sup>, however, they have deliberately not been included into the text. However, they are available upon request from the authors. Although the material is basically self-contained, some prior exposition to logic and mathematical notation is needed.

While many textbooks and courses are centered around some particular logical system, in modern applications, a variety of logics are used and applied. The emphasis of this text is therefore to introduce a range of well-known and important logics in such a way that the common principles behind logic become clear, and the perspective to the theory of abstract logical systems (formalised as *institutions*, using category theory) is opened. The only available textbook on institutions so far [2] starts at a rather sophisticated level and is based on a high level of abstraction - typical examples are particular logics and mathematical constructions related to these. By contrast, the current text starts with a detailed discussion of individual logics, while providing many example theories and arguments in these logics, and only then moves on to the more abstract notions of institution theory. Moreover, we start with the simple and well-known propositional logic, which is very suitable to illustrate many logical concepts that are important also for more complex logics. We then proceed to description logics, which have become important in the field on ontologies and ontology engineering, and the move on the classical first-order logic (and some extensions), which provides more expressive power. Some discussion of modal logic is provided in the outlook.

---

<sup>1</sup>Many thanks to Robert Mattmüller for preparing this!

The presentation of logics not only discusses the classical logical notions like logical theory, logical consequence, proof, consistency, soundness, completeness and so on. We also give attention to modularity and structuring, and for example discuss conservative extension of theories from the very beginning, while later on, a language for structured specifications is introduced in an institution independent way. This is particularly important from a computer science perspective: logical theories used in computer science are often too large to be handled in a flattened unmodular way.

Logic is better learned if you can experiment with tools from the very start. Therefore, this text should be used with accompanying tools. The central tool that we use is the *Heterogeneous tool set* HETS [5, 4, 3], which provides a uniform input notation and tool support for a variety of logics. HETS is freely available at <http://www.dfki.de/sks/hets>, and there is an easy GUI-based installer. We urge the reader to install HETS and try out the examples presented in the text.

It should be noted that the current text is an unfinished draft; while most of the definitions, theorems and exercises are there, especially in the later chapters, explanatory text is still missing.

# Chapter 2

## Logics

Logic has been characterised as the study of sound reasoning, of what follows from what. Hence, the notion of logical consequence is central. In this chapter, we will introduce several logics. We begin with the simplest one, propositional logic. Although many readers will already know about propositional logic, it is nevertheless recommended to at least skim through this chapter and to do the exercises and try out HETS. In particular, the notions of signature morphism, model reduct, conservative, free and cofree extension are not so well-known, and using the HETS truth table prover, these notions can be illustrated very well.

We then proceed to various description logics, centered around the web ontology language OWL. Finally, first-order logic and some extensions for datatypes, subsorting and partial functions (as integrated in the language CASL) will be covered.

### 2.1 Propositional Logics

#### 2.1.1 Foundations

Propositional logic is one of the simplest logics that one can think of; still, it poses many non-trivial problems, is linked intimately to one the greatest open problems of computer science (the P=NP-problem), and has many applications, ranging from circuit design over planning to diagnostic systems. Since propositional logic is well supported with tools, it is often used as a kind of assembly language, that is, problems formulated in other formalisms are translated to propositional logic and then solved there.

The language of logic can be divided into the *logical* and the *non-logical* part. While logical symbols are provided by the logic itself and are fixed, non-logical symbols are provided by the user, tailored towards a particular application. The non-logical symbols are collected in a *signature*. In propositional logic, these are just propositional letters:

**Definition 2.1.1** (Signature). A propositional signature  $\Sigma$  is a set (of propositional letters, or propositional symbols, or propositional variables<sup>1</sup>).

A signature provides us with the basic material to form logical expressions, called formulas or sentences.

**Definition 2.1.2** (Sentence). Given a propositional signature  $\Sigma$ , a propositional sentence over  $\Sigma$  is one produced by the following grammar

$$\phi ::= p \mid \perp \mid \top \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \phi \leftrightarrow \phi$$

with  $p \in \Sigma$ .  $\text{Sen}(\Sigma)$  is the set of all  $\Sigma$ -sentences

Logical sentences provide a formalisation of the grammatical conjunctions “and”, “or”, etc. However note that the formalisation of a natural language sentence can be a non-trivial task. For example, the word “but” often needs to be formalised as logical conjunction, which also means that part of the meaning is lost during formalisation. For a more detailed treatment of this topic, see [1].

*Truth valuations*, or *models* (as we prefer to call them for the reason of achieving uniformity with other logics) provide an interpretation of propositional sentences. Each propositional letter is interpreted as a truth value:

**Definition 2.1.3** (Model). Given a propositional signature  $\Sigma$ , a  $\Sigma$ -model (or  $\Sigma$ -valuation) is a function in  $\Sigma \rightarrow \{T, F\}$ .  $\text{Mod}(\Sigma)$  is the set of all  $\Sigma$ -models.

Models interpret not only the propositional letters, but all sentences. A  $\Sigma$ -model  $M$  can be extended to

$$M^\# : \text{Sen}(\Sigma) \rightarrow \{T, F\}$$

using truth tables (see Figure 2.1 for the truth tables of propositional logic). We now can define what it means for a sentence to be satisfied in a model:

**Definition 2.1.4.**  $\phi$  holds in  $M$  (or  $M$  satisfies  $\phi$ ), written  $M \models_\Sigma \phi$  iff

$$M^\#(\phi) = T$$

We now arrive at the notion of logical consequence, which is central to logic. It formalises the notion of *valid logical argument*. Note that logic is *not* about the truth of individual sentences. In some circumstances (=models), a sentence may be true, in others, it may be false. This is subject to empirical observation, specific sciences, subjective preference etc., and is beyond the scope of logic. What logic provides is a mechanism to describe what valid logical arguments are. Such arguments need to *preserve* truth, but they cannot guarantee truth: if we start with false premises, then a valid logical argument may still lead us to false conclusions.

---

<sup>1</sup>Note that the term “propositional constant” would be more appropriate for reasons that will become clear later on.



$M^\#(p) = M(p)$	$M^\#(\phi) \parallel M^\#(\neg\phi)$
$M^\#(\top) = T$	$\frac{T}{\parallel} F$
$M^\#(\perp) = F$	$\frac{F}{\parallel} T$
(a) base cases	(b) not

$M^\#(\phi)$	$M^\#(\psi)$	$M^\#(\phi \wedge \psi)$	$M^\#(\phi \vee \psi)$	$M^\#(\phi \rightarrow \psi)$	$M^\#(\phi \leftrightarrow \psi)$
$T$	$T$	$T$	$T$	$T$	$T$
$T$	$F$	$F$	$T$	$F$	$F$
$F$	$T$	$F$	$T$	$T$	$F$
$F$	$F$	$F$	$F$	$T$	$T$

(c) and, or, implication, bimplication

Figure 2.1: Truth tables

**Definition 2.1.5** (Logical consequence). *Given  $\Gamma \subseteq \text{Sen}(\Sigma)$  and  $\phi \in \text{Sen}(\Sigma)$ ,  $\phi$  is a logical consequence of  $\Gamma$  (written as  $\Gamma \models \phi$ ), if for all  $M \in \text{Mod}(\Sigma)$*

$$M \models \Gamma \text{ implies } M \models \phi.$$

$\Gamma$  is called the set of premises, and  $\phi$  the conclusion.<sup>1</sup>

**Example 2.1.6.** *An argument in natural language is tested for validity by translating it into propositional logic.*

<p><i>John plays tennis, if it's a sunny weekend day. If John plays tennis, then Mary goes shopping. It is Saturday. It is sunny.</i></p>	<p><i>sunny <math>\wedge</math> weekend <math>\rightarrow</math> tennis tennis <math>\rightarrow</math> shopping saturday sunny</i></p>
<p><i>Mary goes shopping</i></p>	<p><i>shopping</i></p>

*Note that we first have to formalise the argument. Actually, we then can see that shopping is not a logical consequence.*

*saturday  $\rightarrow$  weekend*

---

<sup>1</sup>By contrast, in an implication  $\varphi \rightarrow \psi$ ,  $\varphi$  is called the *antecedent*, and  $\psi$  the *consequent*. Note the difference between implication (a sentence-forming operator) and logical consequence (a relation between sentences).

<i>sunny</i>	$\wedge$	<i>weekend</i>	$\rightarrow$	<i>tennis</i>
<i>tennis</i>	$\rightarrow$	<i>shopping</i>		
<i>saturday</i>				
<i>sunny</i>				
<i>saturday</i>	$\rightarrow$	<i>weekend</i>		
<i>shopping</i>				

The set of premises has the sentence *shopping* as a logical consequence

**Exercise 1**

Show the logical consequence in Exercise 2.1.6 with HETS, using the truth-table prover.<sup>1</sup>

EdNote(1)

**Definition 2.1.7.** Two sentences  $\phi$  and  $\psi$  are logically equivalent,  $\phi \models \psi$  if

$$\{\psi\} \models \phi \text{ and } \{\phi\} \models \psi.$$

**Proposition 2.1.8.** Some logical equivalences.

- $\neg(\phi \wedge \psi) \models (\neg\phi) \vee (\neg\psi)$  (“De Morgan’s law 1”)
- $\neg(\phi \vee \psi) \models (\neg\phi) \wedge (\neg\psi)$  (“De Morgan’s law 2”)
- $\phi \wedge (\psi \vee \chi) \models (\phi \wedge \psi) \vee (\phi \wedge \chi)$  (“Distributivity of  $\wedge$  over  $\vee$ ”)
- $\phi \vee (\psi \wedge \chi) \models (\phi \vee \psi) \wedge (\phi \vee \chi)$  (“Distributivity of  $\vee$  over  $\wedge$ ”)
- $\phi \rightarrow \psi \models \neg\phi \vee \psi$
- $\phi \leftrightarrow \psi \models (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$

The statement  $\phi \models \psi$  is a meta level statement and is sometimes also written as ‘ $\phi \Leftrightarrow \psi$ ’ or ‘ $\phi \equiv \psi$ ’

**Exercise 2**

Show some of the logical equivalences in Prop. 2.1.8 with HETS, using the truth-table prover.

**Definition 2.1.9** (Conjunctive normal form (CNF)). For each sentence, there is an equivalent conjunction of disjunctions of literals. A literal is of the form  $p$  or of form  $\neg p$  with  $(p \in \Sigma)$ .

Dually there is an equivalent disjunction of conjunctions (disjunctive normal form (DNF)).

*Proof sketch.* Use an algorithm that recursively eliminates implication and equivalences, pulls negations inwards and shifts conjunctions outwards, using Prop. 2.1.8 and the following equivalences:

- $\neg\neg\phi \models \phi$

---

<sup>1</sup>EDNOTE: TODO: explain CASL notation

- $\neg\top \models \perp$
- $\neg\perp \models \top$
- $\top \vee \phi \models \top$
- $\perp \wedge \phi \models \perp$
- $\top \wedge \phi \models \phi$   $\top$  is the empty conjunction
- $\perp \vee \phi \models \phi$   $\perp$  is the empty disjunction

□

**Definition 2.1.10** (Theories). A theory is a pair  $T = (\Sigma, \Gamma)$  where  $\Sigma$  is a signature and  $\Gamma \subseteq \text{Sen}(\Sigma)$ . A model of a theory  $T = (\Sigma, \Gamma)$  is a  $\Sigma$ -model  $M$  with  $M \models \Gamma$ . Also

$$T \models \phi \text{ iff } \Gamma \models_{\Sigma} \phi.$$

**Definition 2.1.11.** A theory  $T$  is satisfiable, if it has a model.

**Example 2.1.12.** The theory  $T_1 = (\{p, q\}, \{p \vee q, \neg p\})$  is satisfiable, because it has a model  $M = \{p \mapsto F, q \mapsto T\}$ .

The theory  $T_2 = (\{p, q\}, \{p \vee q, \neg p, \neg q\})$  is not satisfiable, because there is no  $\{p, q\}$ -model for  $\{p \vee q, \neg p, \neg q\}$

### Exercise 3

Show the results of Example 2.1.12 using HETS, using the truth-table prover.

### Proposition 2.1.13.

$$(\Sigma, \Gamma) \models \phi \text{ iff } (\Sigma, \Gamma \cup \{\neg\phi\}) \text{ is unsatisfiable.}$$

*Proof.*

$\Rightarrow$ : Assume  $M$  were a model of  $(\Sigma, \Gamma \cup \{\neg\phi\})$ . Then  $M \models \Gamma$  and  $M \models \neg\phi$ , hence  $M \not\models \phi$ . This is a contradiction to  $(\Sigma, \Gamma) \models \phi$ .

$\Leftarrow$ : Assume  $M \models \Gamma$ ,  $M$  cannot satisfy  $\neg\phi$ , hence  $M \models \phi$ .

□

**Definition 2.1.14.**  $\phi$  is valid (also: “ $\phi$  is a tautology”), if for all  $\Sigma$ -models  $M$

$$M \models \phi.$$

**Proposition 2.1.15.**  $\{\phi\}$  is unsatisfiable iff  $\neg\phi$  is valid

Representation	Arbitrary	CNF	DNF	ROBDD <sup>1</sup>
compact	often	sometimes	sometimes	often
satisfiability	NP-compl.	NP-compl.	linear	quite easy
validity	CoNP-compl.	linear <sup>2</sup>	CoNP-compl.	quite easy
$\wedge$	easy	easy	hard	medium
$\vee$	easy	hard	easy	medium
$\neg$	easy	hard	hard	easy

Table 2.1: Comparison between different representations of formulae

Props. 2.1.13 and 2.1.15 together show that the important questions for propositional theories (logical consequence and validity) can be reduced to satisfiability. This explains the central role that SAT-solvers play.

**Exercise 4** (Propositional Logic I)

- (a) Use the equivalence rules introduced in Prop. 2.1.8 above to push all occurrences of the negation symbol “ $\neg$ ” next to the atoms in the following expressions:

- (i)  $\neg((A \rightarrow B) \vee ((A \rightarrow C) \wedge \neg A))$   
(ii)  $\neg(A \wedge \neg B) \rightarrow A$

- (b) Below are two arguments in English. Translate each argument into logic using an appropriate dictionary, and check whether the argument is logically valid.

- (i) If the king is in the room, then the courtiers laugh only if he laughs. The courtiers always laugh when the jester is in the room. The king never laughs when the jester is in the room. *Therefore*, either the king or the jester is not in the room.  
(ii) If Jones did not meet Smith last night, then either Smith was a murderer, or Jones is telling a lie. If Smith was not a murderer, then Jones did not meet Smith last night, and the murder happened after midnight. If the murder happened after midnight, then either Smith was a murderer, or Jones is telling a lie, but not both. *Therefore*, Smith was a murderer.

**Exercise 5** (Propositional Logic II)

- (a) (i) Find formulae  $A$ ,  $B$ , and  $C$  such that  $\{A, B\}$ ,  $\{A, C\}$ , and  $\{B, C\}$  are consistent, while  $\{A, B, C\}$  is not.  
(ii) For any  $n$ , find an inconsistent set of  $n$  formulae, of which every  $n - 1$  formulae are consistent.

<sup>1</sup>Reduced Ordered Binary Decision Diagrams

<sup>2</sup>A CNF is valid iff all conjuncts are valid. A conjunct is valid if it contains T or if it contains  $p$  and  $\neg p$ .

- (b) (i) Find four pairwise inconsistent non-contradictory formulae.  
(ii) State the maximal number of pairwise inconsistent non-contradictory formulae with two atomic propositions  $p$  and  $q$ .
- (c) Check the validity of the following rules:

$$\frac{A \wedge B}{A} \text{ (\wedge elimination)} \qquad \frac{A \quad B}{A \wedge B} \text{ (\wedge introduction)}$$

$$\frac{\begin{array}{c} [A] \quad [B] \\ \vdots \quad \vdots \\ C \quad C \end{array}}{C} A \vee B \text{ (\vee elimination)} \qquad \frac{A \quad \neg A}{B} \text{ (\neg elimination)} \quad \frac{A \wedge B}{A} \text{ (\wedge elimination)}$$

**Exercise 6** (Propositional Logic III)

Suppose the engine of a car does not perform properly. We want to decide whether we should replace the engine, repair the engine, or replace auxiliary equipment. For the diagnosis, the following symptoms, intermediate conclusions and final decisions or diagnoses should be considered.

Variable	Meaning
<i>black_exhaust</i>	Engine fumes are black
<i>blue_exhaust</i>	Engine fumes are blue
<i>low_power</i>	Engine has low power
<i>overheat</i>	Engine overheats
<i>ping</i>	Engine emits a pinging sound under load
<i>incorrect_timing</i>	Ignition timing is incorrect
<i>low_compression</i>	Compression of engine is low
<i>carbon_deposits</i>	Cylinders have carbon deposits
<i>clogged_filter</i>	Air filter is clogged
<i>clogged_radiator</i>	Radiator is clogged
<i>defective_carburetor</i>	Carburetor is defective
<i>worn_rings</i>	Piston rings are worn
<i>worn_seals</i>	Valve seals are worn
<i>replace_auxiliary</i>	Replace auxiliary equipment
<i>repair_engine</i>	Repair engine
<i>replace_engine</i>	Replace engine

The following facts relate symptoms to intermediate conclusions (facts (i) through (vi)) and intermediate conclusions to final decisions (facts (vii) through (ix)).

- (i) If the engine overheats and the ignition is correct, then the radiator is clogged.  
(ii) If the engine emits a pinging sound under load and the ignition timing is correct, then the cylinders have carbon deposits.  
(iii) If power output is low and the ignition timing is correct, then the piston rings are worn, or the carburetor is defective, or the air filter is clogged.  
(iv) If the exhaust fumes are black, then the carburetor is defective, or the air filter is clogged.

- (v) If the exhaust fumes are blue, then the piston rings are worn, or the valve seals are worn.
- (vi) The compression is low if and only if the piston rings are worn.
- (vii) If the piston rings are worn, then the engine should be replaced.
- (viii) If carbon deposits are present in the cylinders or the carburetor is defective or valve seals are worn, then the engine should be repaired.
- (ix) If the air filter or radiator is clogged, then that auxiliary equipment should be replaced.

Suppose the car owner complains that the engine overheats. Due to a recent engine check, it is known that the ignition timing is correct. What should be done to eliminate the problem?

Answer this question by translating the given information into a propositional CASL specification and checking with HETS which of the final decisions (diagnoses) follow from the symptoms.

### 2.1.2 Proofs for Propositional Logic

There are different proof methods for propositional logic. We will describe them in the subsequent sections.

TODO: always present shopping example.

#### Truth tables

Truth tables directly evaluate the logical consequence relation. For small signatures, truth tables are easy to construct and provide a good overview for humans. You can construct truth tables easily using the truth table prover of HETS. However, since their size grows exponentially with the size of the signature, truth tables are not feasible even for medium-sized signatures. Therefore, HETS limits the size to 16 propositional symbols. Another limitation of truth tables is that they do not generalise to first-order logic.

TODO: describe how they work

#### Natural deduction

In contrast to truth tables, this is really a proof calculus. That is, from a given set of premises, new theorems (logical consequences) are derived using proof rules. The proof rules for natural deduction closely follow human reasoning. For each propositional connective, there is an introduction and an elimination rule. The program Fitch [?] can be used for the construction and verification of natural deduction proofs. Natural deduction is also available for first-order logic and other logics.

$$\begin{array}{c}
\frac{p \quad q}{p \wedge q} \wedge\text{-I} \quad \frac{p \wedge q}{p} \wedge\text{-E1} \quad \frac{p \wedge q}{q} \wedge\text{-E2} \\
\\
\frac{p}{p \vee q} \vee\text{-I1} \quad \frac{q}{p \vee q} \vee\text{-I2} \quad \frac{p \vee q \quad \begin{array}{c} [p] \\ \vdots \\ r \end{array} \quad \begin{array}{c} [q] \\ \vdots \\ r \end{array}}{r} \vee\text{-E} \\
\\
\frac{\begin{array}{c} [p] \\ \vdots \\ \perp \end{array}}{\neg p} \neg\text{-I} \quad \frac{\neg\neg p}{p} \neg\text{-E} \quad \frac{\begin{array}{c} [p] \\ \vdots \\ q \end{array}}{p \rightarrow q} \rightarrow\text{-I} \quad \frac{p \rightarrow q \quad p}{q} \rightarrow\text{-E} \\
\\
\frac{p \rightarrow q \quad q \rightarrow p}{p \leftrightarrow q} \leftrightarrow\text{-I} \quad \frac{p \leftrightarrow q}{p \rightarrow q} \leftrightarrow\text{-E1} \quad \frac{p \leftrightarrow q}{q \rightarrow p} \leftrightarrow\text{-E2} \\
\\
\frac{p \quad \neg p}{\perp} \perp\text{-I} \quad \frac{\perp}{p} \perp\text{-E} \quad \frac{}{\top} \top\text{-I}
\end{array}$$

Figure 2.2: Natural deduction rules for propositional logic

**Definition 2.1.16.**  $T = (\Sigma, \Gamma) \vdash \phi$ , if  $\phi$  can be obtained from  $\Gamma$  by successfully applying the rules in Figure 2.2.

### Resolution

TODO: machine efficient, unnatural SPASS, Vampire

### Davis-Putnam-Logemann-Loveland algorithm.

TODO: expand

DPLL is a backtracking algorithm for testing satisfiability. zChaff, minisat, darwin

It can

- select a literal,
- assign a truth value to it,
- simplify the formula,
- recursively check if the simplified formula is satisfiable
  - if this is the case, the original formula is satisfiable;
  - otherwise, do the recursive check with the opposite truth value.

Implementations: mChaff, zChaff, darwin Crucial: design of the literal selection function.

Example without saturday axiom.

### Tableaux

machine efficient, provides countermodels. Tableau proof methods are also available for first-order logic and other logics. Especially for modal logics and description logics, tableau calculi are very popular.

Isabelle's blast tactic

Interactive: Jitpro <http://ps.uni-sb.de/jitpro/prover.php>

**Theorem 2.1.17** (Soundness).

$$T \vdash \phi \text{ implies } T \models \phi$$

**Theorem 2.1.18** (Completeness).

$$T \models \phi \text{ implies } T \vdash \phi$$

**Definition 2.1.19** (Inconsistency).

(a) (Aristotle): A theory  $T$  is Aristotle-inconsistent, if there is a formula  $\phi$  with

$$T \vdash \phi \text{ and } T \vdash \neg\phi.$$

(b) (Hilbert): A theory  $T$  is  $\perp$ -inconsistent, if

$$T \vdash \perp$$

(c) (Hilbert): A theory  $T$  is absolutely inconsistent, if

$$T \vdash \phi \text{ for an arbitrary } \phi$$

**Proposition 2.1.20.** In propositional logic, all three definitions are equivalent.

*Proof.*

- (a)  $\Rightarrow$  (b) by  $\perp$ -Introduction
- (b)  $\Rightarrow$  (c) by  $\perp$ -Elimination
- (c)  $\Rightarrow$  (a) clear



□

**Example 2.1.21.**

- *Harry: John tells the truth:  $harry \leftrightarrow john$*
- *John: Harry lies:  $john \leftrightarrow \neg harry$*

1	$harry \leftrightarrow john$	
2	$john \leftrightarrow \neg harry$	
3	$harry$	
4	$harry \rightarrow john$	$\leftrightarrow -E (1)$
5	$john$	$\rightarrow -E (3,4)$
6	$john \rightarrow \neg harry$	$\leftrightarrow -E (2)$
7	$\neg harry$	$\rightarrow -E (3,6)$
8	$\perp$	$\perp -I(3,7)$
9	$\neg harry$	$\neg -I (3 - 8)$
10	$\neg harry \rightarrow john$	$\leftrightarrow -E (2)$
11	$john$	$\rightarrow -E (9,10)$
12	$john \rightarrow harry$	$\leftrightarrow -E (1)$
13	$harry$	$\rightarrow -E (11,12)$
14	$\perp$	$\perp -I (9,13)$

*Recall:* By Proposition 2.1.13  $(\Sigma, \Gamma) \models \phi$  iff  $(\Sigma, \Gamma \cup \{\neg\phi\})$  unsatisfiable

**Theorem 2.1.22.**  $(\Sigma, \Gamma) \vdash \phi$  iff  $(\Sigma, \Gamma \cup \{\neg\phi\})$  inconsistent

*Proof.*

$\Rightarrow :$

$\Gamma \vdash \phi$  implies

$$\left. \begin{array}{l} \Gamma \cup \{\neg\phi\} \vdash \phi \\ \Gamma \cup \{\neg\phi\} \vdash \neg\phi \end{array} \right\} \Gamma \cup \{\neg\phi\} \vdash \perp$$

$\Leftarrow :$

$$\begin{array}{l} \Gamma \cup \{\neg\phi\} \vdash \perp \\ \text{By } \neg -I: \Gamma \vdash \neg\neg\phi \\ \text{By } \neg -E: \Gamma \vdash \phi \end{array}$$

□

*Proof.* We prove soundness by induction on the length of the proof. Each rule needs to be proven sound. To show completeness, we use that the following three statements are equivalent:

- completeness
- every consistent theory is satisfiable
- every unsatisfiable theory is inconsistent

$\Rightarrow$  : Assume completeness and let  $T$  be unsatisfiable. Hence  $T \models \perp$ . By completeness we conclude  $T \vdash \perp$

$\Leftarrow$  : Assume that every unsatisfiable theory is inconsistent. Let  $T \models \phi$ . Then  $T \cup \{\neg\phi\}$  is unsatisfiable. By assumption we know, that  $T \cup \{\neg\phi\}$  is inconsistent. Hence:  $T \vdash \phi$ .

□

**Theorem 2.1.23.** *Every consistent theory is satisfiable*

*Proof sketch:*

Let  $T$  be consistent.

Extend  $T$  to a maximal consistent theory  $T'$

Define a model  $M$  by:  $M(\phi) = \top$  if  $T' \vdash \phi$

Then show the truth lemma:  $M \models \phi$  iff  $T' \vdash \phi$

From this it is clear, that  $M$  is a  $T'$ -model, and hence also a  $T$ -model. □

**Exercise 7** (Logical consequence or not?)

Evaluate the validity of the following argument. If it is a logical consequence, use the programs *SPASS*, *Fitch* and *Jitpro* to construct formal (resolution, natural deduction, tableau) proofs to show this. Otherwise, use *Tarski's World* to construct a counterexample.<sup>1</sup>

1	Cube(a) $\vee$ (Cube(b) $\rightarrow$ Tet(c))
2	Tet(c) $\rightarrow$ Small(c)
3	(Cube(b) $\rightarrow$ Small(c)) $\rightarrow$ Small(b)
4	<div style="border-top: 1px solid black; padding-top: 2px;"><math>\neg</math>Cube(a) <math>\rightarrow</math> Small(b)</div>

**Exercise 8** (Inconsistency)

Consider the set  $\mathcal{T} = \{(A \wedge B) \rightarrow \neg A, C \vee A, \neg A \rightarrow A, B\}$ . Use *SPASS*, *Fitch* and *Jitpro* to construct formal proofs showing that  $\mathcal{T} \vdash \perp$ .

**Exercise 9** (New connectives)

Consider the following truth table for the ternary connective  $\diamond$ .

---

<sup>1</sup>SPASS is available within Hets, see <http://www.dfki.de/sks/hets>. Fitch and Tarski's World can be downloaded from an internal web page shown in the lecture. Jitpro is available under <http://ps.uni-sb.de/jitpro/prover.php>.

$P$	$Q$	$R$	$\diamond(P, Q, R)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Express  $\diamond$  using only the connectives  $\vee$ ,  $\wedge$ , and  $\neg$ . Can you simplify the result such that the simplified sentence has no more than two occurrences each of  $P$ ,  $Q$ , and  $R$ , and no more than six occurrences of the Boolean connectives  $\vee$ ,  $\wedge$ , and  $\neg$ ?

### 2.1.3 Conservative Extensions

```

1 specc Animals =
2   props bird, penguin, living
3     .penguin => bird
4     .bird => living
5   then % cons
6     prop animal
7     .bird => animal
8     .animal => living
9 end

```

```

1 specc penguin =
2   props bird, penguin
3     .penguin => bird
4   then
5     prop can_fly
6     .bird => can_fly
7     .penguin => not can_fly
8 end

```

Listing 2.1: Example for conservative extensions in CASL

**Definition 2.1.24.** *Given two signatures  $\Sigma_1, \Sigma_2$  a signature morphism is a function  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  (Note that signatures are sets).*

**Definition 2.1.25.** *A signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  induces a sentence translation  $\sigma : \text{Sen}(\Sigma_1) \rightarrow \text{Sen}(\Sigma_2)$ , defined inductively by*

- $\sigma(\perp) = \perp$
- $\sigma(\top) = \top$

- $\sigma(\phi_1 \wedge \phi_2) = \sigma(\phi_1) \wedge \sigma(\phi_2)$
- *etc.*

**Definition 2.1.26.** A signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  induces a model reduction  $\_|\sigma : Mod(\Sigma_2) \rightarrow Mod(\Sigma_1)$ . Given  $M \in Mod(\Sigma_2)$  i.e.  $M : \Sigma \rightarrow \{T, F\}$ , then  $M|_\sigma \in Mod(\Sigma_1)$  is defined as  $M|_\sigma(\phi) := M(\sigma(\phi))$  i.e.  $M|_\sigma = M \circ \sigma$

**Theorem 2.1.27** (Satisfaction condition). Given a signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$ ,  $M_2 \in Mod(\Sigma_2)$  and  $\phi_1 \in Sen(\Sigma_1)$ , then:

$$M_2 \models_{\Sigma_2} \sigma(\phi_1) \text{ iff } M_2|_\sigma \models_{\Sigma_1} \phi_1$$

(“truth is invariant under change of notation.”)

*Proof.* By induction on  $\phi_1$ .

- Atomic Sentences :  $M_2 \models \sigma(\phi)$  iff  $M_2(\sigma(\phi)) = \top$  iff  $(M_2 \circ \sigma)(\phi) = \top$  iff  $M_2|_\sigma(\phi) = \top$  iff  $M_2|_\sigma \models \phi$
- Negations :  $M_2 \models \sigma(\neg\phi)$  iff  $M_2 \models \neg\sigma(\phi)$  iff  $M_2 \not\models \sigma(\phi)$  iff  $M_2|_\sigma \not\models \phi$  iff  $M_2|_\sigma \models \neg\phi$ .
- ...

□

TODO: example views in propositional logic.

```

1
2 spec Circular_Reasoning =
3   props p, q, r
4   . p <=> q
5   . p <=> r
6   . p <=> r
7 end
8
9 spec JohnMaryHarry =
10  props john, mary, harry
11  . john => harry
12  . not harry \/\ mary
13  . not (mary /\ not john)
14 end
15
16 view r : Circular_Reasoning to JohnMaryHarry =
17   p |-> john, q |-> harry, r |-> mary
18 end

```

<pre> 1 <b>spec</b> Sp = 2   Σ<sub>1</sub> 3   Γ<sub>1</sub> 4 <b>then</b> 5   Σ<sub>Δ</sub> 6   Γ<sub>Δ</sub> 7 <b>end</b> </pre>	<pre> 1 <b>spec</b> Animals = 2   <b>props</b> bird, penguin 3   . penguin =&gt; bird 4 <b>then</b> 5   <b>prop</b> can_fly 6   . penguin =&gt; not can_fly 7 <b>end</b> </pre>
--	---

Listing 2.2: Theory extensions in HetCasl.

**Definition 2.1.28.** A theory morphism  $(\Sigma_1, \Gamma_1) \rightarrow (\Sigma_2, \Gamma_2)$  is a signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  such that for  $M_2 \in \text{Mod}(\Sigma_2, \Gamma_2)$  we have  $M_2|_{\sigma} \in \text{Mod}(\Sigma_1, \Gamma_1)$

It is model-theoretically conservative, if for each  $M_1 \in \text{Mod}(\Sigma_1, \Gamma_1)$  there is  $M_2 \in \text{Mod}(\Sigma_2, \Gamma_2)$  with  $M_2|_{\sigma} = M_1$

Extensions (casl keyword **then**; cf. Listing 2.2) always lead to a theory morphism (by definition). Semantics for the Casl code are defined as follows: Theory morphism  $\sigma : (\Sigma_1, \Gamma_1) \rightarrow (\Sigma_2, \Gamma_2)$ , where  $\Sigma_2 = \Sigma_1 \cup \Sigma_{\Delta}$  and  $\Gamma_2 = \Gamma_1 \cup \Gamma_{\Delta}$ , such that  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  is the inclusion.

**Definition 2.1.29.** Let  $T_i = (\Sigma_i, \Gamma_i)$  for  $i \in \{1, 2\}$ . A theory morphism  $\sigma : T_1 \rightarrow T_2$  is model-theoretically-conservative, if any  $M_1 \in \text{Mod}(T_1)$  has a  $\sigma$ -expansion to a  $\Sigma_2$ -model that is a model

$$M_2 \in \text{Mod}(T_2), \text{ with } M_2|_{\sigma} = M_1.$$

**Definition 2.1.30.** Let  $T_i = (\Sigma_i, \Gamma_i)$  for  $i \in \{1, 2\}$ . A theory morphism  $\sigma : T_1 \rightarrow T_2$  is consequence-theoretically conservative<sup>1</sup>, if for each  $\phi_1 \in \text{Sen}(\Sigma_1)$

$$T_2 \models \sigma(\phi_1) \text{ implies } T_1 \models \phi_1.$$

**Theorem 2.1.31** (Compactness theorem).

If  $\Gamma \models_{\Sigma} \phi$ , then  $\Gamma' \models_{\Sigma} \phi$  for some finite  $\Gamma' \subseteq \Gamma$

*Proof.* Let  $\Gamma \models_{\Sigma} \phi$ . By completeness  $\Gamma \vdash_{\Sigma} \phi$ . Since the proof rules all have finitely many premises, only finitely many premises can be used in the proof of  $\Gamma \vdash_{\Sigma} \phi$ . Hence,  $\Gamma' \vdash_{\Sigma} \phi$  for some finite  $\Gamma' \subseteq \Gamma$ . By soundness  $\Gamma' \models_{\Sigma} \phi$ .  $\square$

**Definition 2.1.32.** Let  $M \in \text{Mod}(\Sigma)$ . Then

$$\text{Th}(M) := \{\phi \in \text{Sen}(\Sigma) \mid M \models_{\Sigma} \phi\}$$

<sup>1</sup>In the literature this is sometimes defined as *proof-theoretical conservativity*. We use the term *consequence-theoretical conservativity* because it is defined over  $\models$  and not over  $\vdash$ .

**Theorem 2.1.33.**  $\sigma : T_1 \rightarrow T_2$  is model-theoretically conservative iff it is consequence-theoretically conservative.

*Proof.*

$\Rightarrow$  : Assume that  $\sigma : T_1 \rightarrow T_2$  is model-theoretically conservative. Let  $\phi_1$  be a formula, such that  $T_2 \models_{\Sigma_2} \sigma(\phi_1)$ . Let  $M_1$  be a Model  $M_1 \in \text{Mod}(T_1)$ . By assumption there is a Model  $M_2 \in \text{Mod}(T_2)$  with  $M_2|_{\sigma} = M_1$ . Since  $T_2 \models_{\Sigma_2} \sigma(\phi_1)$ , we have  $M_2 \models \sigma(\phi_1)$ . By the satisfaction condition  $M_2|_{\sigma} \models_{\Sigma_1} \phi_1$ . Hence  $M_1 \models \phi_1$ . Altogether  $T_1 \models_{\Sigma_1} \phi_1$ .

$\Leftarrow$  : Assume that  $\sigma : T_1 \rightarrow T_2$  is consequence-theoretically conservative. Let  $M_1$  be a Model  $M_1 \in \text{Mod}(T_1)$ . Assume that  $M_1$  has no  $\sigma$ -extension to a  $T_2$ -model. This means that  $T_2 \cup \sigma(\text{Th}(M_1)) \models \perp^1$ . Hence by compactness we have  $T_2 \cup \sigma(\Gamma) \models \perp$  for a finite  $\Gamma \subseteq \text{Th}(M_1)$ . Let  $\Gamma = \{\phi_1, \dots, \phi_n\}$ . Thus  $T_2 \cup \sigma(\{\phi_1, \dots, \phi_n\}) \models \perp$  and hence  $T_2 \models \sigma(\phi_1) \wedge \dots \wedge \sigma(\phi_n) \rightarrow \perp$ . This means  $T_2 \models \sigma(\phi_1 \wedge \dots \wedge \phi_n \rightarrow \perp)$ . By assumption  $T_1 \models \phi_1 \wedge \dots \wedge \phi_n \rightarrow \perp$ . Since  $M_1 \in \text{Mod}(T_1)$  and  $M_1 \models \phi_i$ , ( $1 \leq i \leq n$ ), also  $M_1 \models \perp$ . Contradiction to the assumption that  $M_1$  has no  $\sigma$ -extension to a  $T_2$  model. □

**Theorem 2.1.34.** If  $T_1 \xrightarrow{\sigma_1} T_2 \xrightarrow{\sigma_2} \dots \xrightarrow{\sigma_{n-1}} T_n$  are model-theoretically conservative, and  $T_1$  is satisfiable, then  $T_n$  is satisfiable.

*Proof.* Clear. □

**Exercise 10** (Conservative Extensions)

Consider your solution to Exercise 1.3 from the last exercise sheet, and consider the theory morphism  $\sigma : (\Sigma_1, \Gamma_1) \rightarrow (\Sigma_2, \Gamma_2)$ , where

$$\begin{aligned} \Sigma_1 &= \{black\_exhaust, blue\_exhaust, low\_power, overheat, ping, \\ &\quad incorrect\_timing, clogged\_filter, low\_compression, carbon\_deposits, \\ &\quad clogged\_radiator, defective\_carburetor, worn\_rings, worn\_seals\}, \\ \Sigma_2 &= \Sigma_1 \cup \{replace\_auxiliary, repair\_engine, replace\_engine\}, \end{aligned}$$

$\Gamma_1$  contains all the axioms corresponding to the symptoms (the overheating engine and the fact that the ignition timing is correct) as well as all the axioms describing diagnostic rules (i.e., the formalizations of facts (i) through (vi) in the informal description in Exercise 1.3).  $\Gamma_2$  contains all axioms from  $\Gamma_1$  plus the three rules corresponding to facts (vii) through (ix). The morphism  $\sigma$  is the inclusion mapping from  $\Sigma_1$  into  $\Sigma_2$  mapping each proposition to itself.

---

<sup>1</sup>This step is only possible in propositional logic. The other direction of the proof can also be done in any other logic.

- (a) Show that  $\sigma$  is a model-theoretically conservative theory morphism.
- (b) Reformulate your HETS specification such that  $(\Sigma_2, \Gamma_2)$  is specified as an extension to  $(\Sigma_1, \Gamma_1)$  using the **then** keyword. Additionally, indicate that the extension is supposed to be conservative using **%cons**. Use HETS to prove that this is indeed the case (you will need the latest nightly build of HETS to do that<sup>1</sup>).

**Exercise 11** (Conservative extensions)

Consider the following specifications.

<p>(i)</p> <pre> <b>logic</b> PROPOSITIONAL <b>spec</b> BLOCKSHAPES =   <b>props</b> <i>cube tetrahedon</i>   • <i>cube</i> <math>\vee</math> <i>tetrahedon</i>   • <math>\neg</math>(<i>cube</i> <math>\wedge</math> <i>tetrahedon</i>) <b>then</b>                               %%cons?   <b>prop</b> <i>dodecahedron</i>   • <i>cube</i> <math>\vee</math> <i>tetrahedon</i> <math>\vee</math> <i>dodecahedron</i>   • <math>\neg</math>(<i>cube</i> <math>\wedge</math> <i>dodecahedron</i>)   • <math>\neg</math>(<i>tetrahedon</i> <math>\wedge</math> <i>dodecahedron</i>)   • <math>\neg</math><i>cube</i> <math>\Rightarrow</math> <i>dodecahedron</i> <b>end</b> </pre>	<p>(ii)</p> <pre> <b>logic</b> PROPOSITIONAL <b>spec</b> IMPLICATIONS =   <b>props</b> <i>a b c</i>   • <i>a</i> <math>\Rightarrow</math> <i>b</i>   • <i>b</i> <math>\Rightarrow</math> <i>c</i> <b>then</b>                               %%cons?   <b>prop</b> <i>d</i>   • <math>\neg</math>(<i>d</i> <math>\Rightarrow</math> <i>a</i>) <b>end</b> </pre>
--	--

- (a) Decide whether the extensions in (i) and (ii) are conservative. If they are not, provide models that cannot be expanded.
- (b) In case of lacking conservativity, use the theorem from the lecture to construct from the model a sentence that can be proven in the extended theory, but not in the base theory.

**2.1.4 Freeness**

Freeness and cofreeness constraints are a powerful mechanism at the level of structured specifications. They work for any logic. Propositional logic is a good starting point for learning about freeness and cofreeness, since things are much less complicated here when compared with other logics.

Consider the following two statements:

Harry: John tells the truth.

John: If Mary is right, then Harry does not tell the truth.

---

<sup>1</sup>You can download the new HETS library from the lecture wiki (Resources/Software) and follow the installation instructions provided there.

Let us formalise these statements and look at the logical consequences. We introduce three propositions telling us whether Harry, John, resp. Mary tell the truth.

```

1 spec Liar0 =
2   prop mary
3   props harry, john
4     . harry => john           %(whenjohn)%
5     . john => (mary => not harry) %(whenharry)%
6 then %implies
7   . harry %(harry)%
8   . john  %(john)%
9   . mary  %(mary)%
10  . not harry %(notharry)%
11  . not john  %(notjohn)%
12  . not mary  %(notmary)%
13 end

```

Actually, when calling HETS with the truth table prover, we get e.g. for the first goal:

```

1 Legend:
2 M = model of the premises
3 + = OK, model fulfills conclusion
4 - = not OK, counterexample for logical consequence
5 o = OK, premises are not fulfilled, hence conclusion is irrelevant
6
7 || harry | john | mary || whenjohn | whenharry || harry
8 =====+=====+=====+=====+=====+=====
9 M- ||      F |   F |   F ||         T |         T ||      F
10 M- ||      F |   F |   T ||         T |         T ||      F
11 M- ||      F |   T |   F ||         T |         T ||      F
12 M- ||      F |   T |   T ||         T |         T ||      F
13 o  ||      T |   F |   F ||         F |         T ||      T
14 o  ||      T |   F |   T ||         F |         T ||      T
15 M+ ||      T |   T |   F ||         T |         T ||      T
16 o  ||      T |   T |   T ||         T |         F ||      T

```

The other goal cannot be proved either. So this theory cannot decide the truth of the propositional letters, and it leaves open whether Harry, John or Mary tell the truth or lie, and indeed, we have five possible cases (indicated by the five models, i.e. those rows marked with "M"). A semantics that admits many possible interpretations and only constrains them by logical formulas is called *open world semantics*.

By contrast, a *closed world semantics* assumes some default, e.g. any propositional letter whose truth value cannot be determined is assumed to be false. Indeed, *free* or *initial semantics* imposes this kind of constraints. As a prerequisite, we need to define a partial order on propositional models:

**Definition 2.1.35.** *Given a propositional signature  $\Sigma$  and two  $\Sigma$ -models  $M$  and  $M'$ , then  $M \leq M'$  if  $M(p) = \text{true}$  implies  $M'(p) = \text{true}$  for all  $p \in \Sigma$*



Then, a free (or initial) specification, written  $\text{free}\{SP\}$ , selects the least model of a specification:

$$\text{Mod}(\text{free}\{SP\}) = \{M \in \text{Mod}(SP) \mid M \text{ least model in } \text{Mod}(SP)\}$$

Note that a least model need not exist; in this case, the model class is empty, hence the free specification inconsistent. Coming back to our example:

```

1 spec Liar1 =
2   free {
3     prop mary
4     props harry, john
5       . harry => john           %(whenjohn)%
6       . john => (mary => not harry) %(whenharry)%
7   }
8   then %implies
9     . not harry %(notharry)%
10    . not john  %(notjohn)%
11    . not mary  %(notmary)%
12 end

```

With the HETS truth table prover, we now get:

	harry	john	mary	notharry	notjohn	free	notmary
M+	F	F	F	T	T	T	T
o	F	F	T	T	T	F	F
o	F	T	F	T	F	F	T
o	F	T	T	T	F	F	F
o	T	F	F	F	T	F	T
o	T	F	T	F	T	F	F
o	T	T	F	F	F	F	T
o	T	T	T	F	F	F	F

That is, Harry, John and Mary all are lying! (We are not forced by the specification to think that they tell the truth, so by minimality of the initial model, the propositional letters are all assigned false.)

Of course, the assumption that propositional letters are false by default is somewhat arbitrary. We could have taken the opposite assumption. Indeed, this exactly is what final (or cofree) specifications do:

$$\text{Mod}(\text{cofree}\{SP\}) = \{M \in \text{Mod}(SP) \mid M \text{ greatest model in } \text{Mod}(SP)\}$$

However, no greatest model exists, hence the cofree specification is inconsistent:

We can also mix the open and closed world assumptions. Assume that we want to be unspecific about Mary, but use closed world assumption for Harry and John. Then we write:

```

1 spec Liar2 =
2   cofree {
3     prop mary
4     props harry, john
5     . harry => john           %(whenjohn)%
6     . john => (mary => not harry) %(whenharry)%
7   }
8 then %implies
9   . false %(false)%
10 end

```

	harry	john	mary	whenjohn	whenharry	cofree	false
3	o	F	F	F	T	T	F
4	o	F	F	T	T	T	F
5	o	F	T	F	T	T	F
6	o	F	T	T	T	T	F
7	o	T	F	F	F	T	F
8	o	T	F	T	F	T	F
9	o	T	T	F	T	T	F
10	o	T	T	T	T	F	F

```

1 spec Liar3 =
2   prop mary
3 then
4   free {
5     props harry, john
6     . harry => john           %(whenjohn)%
7     . john => (mary => not harry) %(whenharry)%
8   }
9 then %implies
10  . not harry %(harry)%
11  . not john  %(john)%
12 end

```

The semantics is as follows:

$$\begin{aligned} \text{Mod}(SP_1 \text{ then free}\{SP_2\}) = \\ \{M \in \text{Mod}(SP_1 \text{ then } SP_2) \mid M \text{ is the least model in} \\ \text{Mod}(SP_1 \text{ then } SP_2) \text{ with the same } \sigma\text{-reduct as } M\} \end{aligned}$$

and as a result, we obtain that both Harry and John lie (independently of what Mary concerns!):

		harry		john		mary		whenjohn		whenharry		free		harry	
1															
2		====+		====+		====+		====+		====+		====+		====+	
3	M+		F		F		F		T		T		T		T
4	M+		F		F		T		T		T		T		T
5	o		F		T		F		T		T		F		T
6	o		F		T		T		T		T		F		T
7	o		T		F		F		F		T		F		F
8	o		T		F		T		F		T		F		F
9	o		T		T		F		T		T		F		F
10	o		T		T		T		T		F		F		F

The dual is cofree with mixed open and closed world semantics:

```

1 spec Liar4 =
2   prop mary
3   then
4     cofree {
5       props harry, john
6         . harry => john           %(whenjohn)%
7         . john => (mary => not harry) %(whenharry)%
8     }
9   then %implies
10    . harry \ / mary %(harrymary)%
11    . john   %(john)%
12 end

```

$$\begin{aligned} \text{Mod}(SP_1 \text{ then cofree}\{SP_2\}) = \\ \{M \in \text{Mod}(SP_1 \text{ then } SP_2) \mid M \text{ is the greatest model in} \\ \text{Mod}(SP_1 \text{ then } SP_2) \text{ with the same } \sigma\text{-reduct as } M\} \end{aligned}$$

with the result that John tells the truth, and at least either of Harry and Mary as well:

**Exercise 12** (Logelei)

Consider the following Logelei:

When I recently took a train, three teenagers boarded in Bremen, a boy and two girls. The boy asked his companions (named Olga

		harry		john		mary		whenjohn		whenharry		cofree		harrymary	
1															
2	====+	=====	+	=====	+	=====	+	=====	+	=====	+	=====	+	=====	
3	o		F		F		F		T		T		F		F
4	o		F		F		T		T		T		F		T
5	o		F		T		F		T		T		F		F
6	M+		F		T		T		T		T		T		T
7	o		T		F		F		F		T		F		T
8	o		T		F		T		F		T		F		T
9	M+		T		T		F		T		T		T		T
10	o		T		T		T		T		F		F		T

and Petra) who from their class would come to the party that was planned. It was an interesting subject for them, although they made only indirect statements.

Olga started: “If neither Bernd nor Christian come, then Nobert won’t come either.”

Then Petra: “If Dieter and Norbert come, then so will Elgar.”

Olga: “If neither Axel nor Lars come, so won’t Christian.”

Petra: “If Fabian does not come to the party, then Jürgen won’t, provided Martin comes.”

Olga: “If both Haug and Axel come, then Christian won’t appear.”

Petra: “If Martin joints the party, then Bernd will (if Lars does not come) join the party as well.”

The trains reached Osnabrück, where the three teenagers left. The boy, who left last, just turned to me and whispered: “Our conversation will have confused you a bit. You have to notice that one of the girls always tells the truth, while the other one never utters a true sentence.”

Before I could ask him who the liar was, the boy has left the train. Since then, I speculate who would come to the party and who wouldn’t. Who??!

Specify this problem, using alternatively ordinary, free, or cofree specifications. Note that free and cofree specifications require a different style of axioms. Discuss the differences.

## 2.2 Description Logics

Description Logics emerged from the field of Knowledge Representation. Semantic Networks (e.g. KL-ONE) were used to reason about concepts, subclassing and relations between these concepts. In such networks (cf. Figure 2.3a) the meaning of edges was not always well defined and in some cases the formalization relied on undecidable First-Order Logics. Description Logics

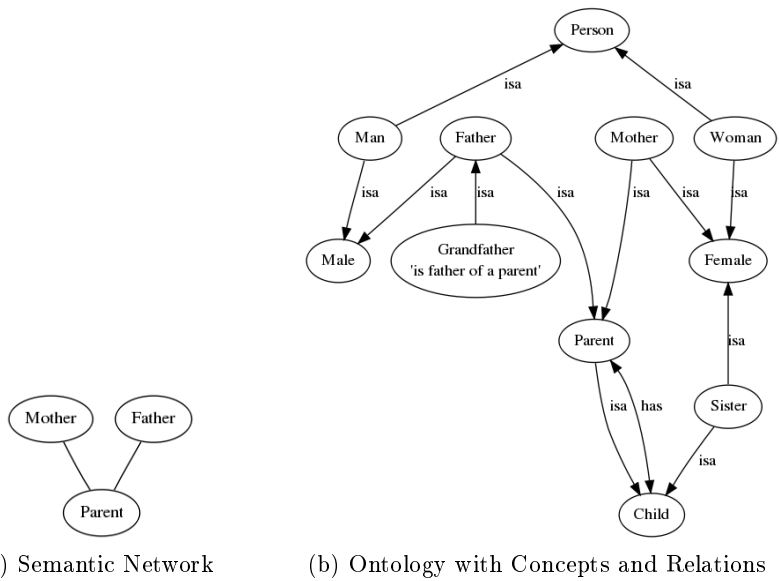


Figure 2.3: Examples of Knowledge Representation in Semantic Networks

are (more or less efficiently, depending on the intended use) decidable fragments of First-Order Logic.

**Applications:**

- Ontologies (cf. Figure 2.3b), semantic web, swoogle
- Software Engineering
- Configurations (e.g. of cars, computer systems, etc.)
- Medicine
- Natural language processing
- Database management
- OWL (Web Ontology Language)

**Syntactic elements** Figure 2.3b already shows the basic elements of Description Logics:

- *Concepts* (in OWL: *classes*) (Mother, Father, etc.)
- *Subsumption*  $C \sqsubseteq D$  (read: “ $C$  is subsumed by  $D$ ”) means that each  $C$  is a  $D$

- $Woman \sqsubseteq Person$
- $Father \sqsubseteq Male$
- ...
- To relate concepts, we need *roles* (in OWL: *properties*) like 'hasChild'.
  - $Parent \sqsubseteq \exists hasChild. \top$  ( $\top$ : top concept, includes everything. In OWL: *Thing*)
  - $Parent \sqsubseteq \exists hasChild. Child$
  - $Child \sqsubseteq \exists hasParent. \top$  (Bad, because *hasChild* is converse to *hasParent* which is not expressed here)
  - $Child \sqsubseteq \exists hasChild^-. \top$  (Better formalization)
  - $hasParent \equiv hasChild^-$  (Alternative, not possible in every DL)
  - $hasGrandfather \equiv (\exists hasChild. \exists hasChild. \top) \sqcap Male$  ( $C \equiv D$  is an abbreviation for  $C \sqsubseteq D$  and  $D \sqsubseteq C$ )
  - $hasGrandfather \equiv (\exists hasChild. Parent) \sqcap Father$  (Alternative formalization)

These axioms are generally split up in two sets. The *TBox* contains a set of subsumptions and definitions involving concepts and roles. The *ABox* contains individuals and their membership in concepts and roles (e.g.  $john : Father, hasChild(john, harry)$ ).

### 2.2.1 Foundations

**Definition 2.2.1.** A DL-signature  $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$  consists of

- a set  $\mathbf{C}$  of concept names,
- a set  $\mathbf{R}$  of role names,
- a set  $\mathbf{I}$  of individual names,

**Definition 2.2.2.** For a signature  $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$  the set of  $\mathcal{ALC}$ -concepts<sup>1</sup> over  $\Sigma$  is defined by the following grammar:

$C ::=$	$A$ for $A \in \mathbf{C}$	<i>(Hets) Manchester syntax</i> <i>a concept name</i>
	$\top$	<i>Thing</i>
	$\perp$	<i>Nothing</i>
	$\neg C$	<i>not C</i>
	$C \sqcap C$	<i>C and C</i>
	$C \sqcup C$	<i>C or C</i>
	$\exists R. C$ for $R \in \mathbf{R}$	<i>R some C</i>
	$\forall R. C$ for $R \in \mathbf{R}$	<i>R only C</i>

**Definition 2.2.3.** The set of  $\mathcal{ALC}$ -Sentences over  $\Sigma$  ( $Sen(\Sigma)$ ) is defined as

---

<sup>1</sup> $\mathcal{ALC}$  stands for “attributive language with complement”

- $C \sqsubseteq D$ , where  $C$  and  $D$  are  $\mathcal{ALC}$ -concepts over  $\Sigma$ .

*Class: C SubclassOf: D*

- $a : C$ , where  $a \in I$  and  $C$  is a  $\mathcal{ALC}$ -concept over  $\Sigma$ .

*Individual: a Types: C*

- $R(a_1, a_2)$ , where  $R \in \mathbf{R}$  and  $a_1, a_2 \in I$ .

*Individual: a1 Facts: R a2*

**Definition 2.2.4.** Given  $\Sigma = (\mathbf{C}, \mathbf{R}, I)$ , a  $\Sigma$ -model  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ , where

- $\Delta^{\mathcal{I}}$  is a non-empty set
- $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$  for each  $A \in \mathbf{C}$
- $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$  for each  $R \in \mathbf{R}$
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$  for each  $a \in I$

**Definition 2.2.5.** We can extend  $\cdot^{\mathcal{I}}$  to all concepts as follows:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\perp^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}. (x, y) \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}. (x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}
\end{aligned}$$

**Definition 2.2.6** (Satisfaction of sentences in a model).

$$\begin{aligned}
\mathcal{I} \models C \sqsubseteq D &\quad \text{iff} \quad C^{\mathcal{I}} \subseteq D^{\mathcal{I}}. \\
\mathcal{I} \models a : C &\quad \text{iff} \quad a^{\mathcal{I}} \in C^{\mathcal{I}}. \\
\mathcal{I} \models R(a_1, a_2) &\quad \text{iff} \quad (a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \in R^{\mathcal{I}}.
\end{aligned}$$

**Definition 2.2.7.** For  $\Gamma \subseteq \text{Sen}(\Sigma)$ ,  $\phi \in \text{Sen}(\Sigma)$ ,  $\phi$  is a logical consequence of  $\Gamma$  (written:  $\Gamma \models_{\Sigma} \phi$ ), if for each  $\Sigma$ -model  $\mathcal{I}$

$$\mathcal{I} \models \Gamma \text{ implies } \mathcal{I} \models \phi.$$

If  $\Gamma$  contains only subsumptions,  $\Gamma$  is written as  $\mathcal{T}$  (TBox).

If  $\Gamma$  contains only sentences  $a : C$  and  $R(a_1, a_2)$ ,  $\Gamma$  is written as  $\mathcal{A}$  (ABox).

## Derived sentences (syntactic sugar)

- $C \equiv D$  for  $\{C \sqsubseteq D, D \sqsubseteq C\}$

Class: C EquivalentTo: D

- $\text{disjoint}(C, D)$  for  $C \sqcap D \sqsubseteq \perp$

Class: C DisjointTo: D

- $\text{domain}(R) \equiv C$  for  $\exists R.T \sqsubseteq C$

ObjectProperty: R Domain: C

- $\neg a : C$  for  $a : \neg C$

- $\text{unsat}(C)$  for  $C \equiv \perp$

## TBox reasoning

$\mathcal{T} \models C \sqsubseteq D$  iff  $\mathcal{T} \models \text{unsat}(C \sqcap \neg D)$

$\mathcal{T} \models \text{unsat}(C)$  iff  $\mathcal{T} \models C \sqsubseteq \perp$

Complexity of TBox reasoning for  $\mathcal{ALC}$ :

- general TBoxes: EXPTIME complete
- empty or acyclic TBoxes: PSPACE complete<sup>1</sup>.

Acyclic TBoxes contain only definitions  $A \equiv C$ , such that concept dependency is acyclic ( $A$  depends on all concepts occurring in  $C$ ).

Description Logics use *open world semantics*, i.e. semantics of a theory (TBox, ABox) is its class of models. Typically there are many models of a theory interpreting concepts, roles and individuals in different ways. In contrast *closed world semantics* would assume, that unspecified facts are false:

### Example 2.2.8.

1	<i>VegetarianPizza</i>	$\sqsubseteq$	<i>Pizza</i>
2	<i>MagheritaPizza</i>	$\sqsubseteq$	<i>Pizza</i>
3	<i>TomatoTopping</i>	$\sqsubseteq$	<i>VegetableTopping</i>
4	<i>MozzarellaTopping</i>	$\sqsubseteq$	<i>CheeseTopping</i>
5	<i>VegetarianPizza</i>	$\equiv$	$\forall \text{hasTopping} (\text{VegetableTopping} \sqcup \text{CheeseTopping})$
6	<i>MagheritaPizza</i>	$\sqsubseteq$	$\exists \text{hasTopping} \text{MozzarellaTopping} \sqcap$
7			$\exists \text{hasTopping} \text{TomatoTopping} \sqcap$
8			$\forall \text{hasTopping} (\text{MozzarellaTopping} \sqcup \text{TomatoTopping})$

<sup>1</sup>We know that  $P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$  and that  $P \subset EXPTIME$ , so it is possible that  $PSPACE \subset EXPTIME$ .



**ABox-Reasoning:** for example: Instance checking:

$$\mathcal{T}, \mathcal{A} \models a : C \text{ iff } \mathcal{T} \cup \mathcal{A} \cup \{ \text{not } a : C \} \text{ inconsistent}$$

Complexity of deciding ABox consistency may be harder than TBox reasoning, but it usually is not. For  $\mathcal{ALC}$  it is PSpace/ExpTime complete.

**Exercise 13** (Description Logics)

Familiarize yourself with the *pizza ontology*.

It can be found at <http://www.co-ode.org/ontologies/pizza/>.

**Exercise 14** (Specification extensions)

Construct a specification extension such that the basis specification has models with 0, 1, 2 and 3 different expansions.

**Exercise 15** (Pizza ontology)

Formalize the following statements from the pizza ontology in Hets:<sup>1</sup>

- Pizza is food.
- Pizza base is food.
- Pizza topping is food.
- Pizzas, pizza bases, and pizza toppings are disjoint sets of things.
- A fish topping is a pizza topping.
- A meat topping is a pizza topping.
- Pizzas have pizza toppings.
- Pizzas have unique pizza bases.
- Thin and crispy pizza bases are pizza bases.
- A thin and crispy pizza is a pizza that only has a thin and crispy base.
- An interesting pizza is a pizza that has at least three toppings.
- A vegetarian pizza is a pizza which has neither a meat topping nor a fish topping.

**Exercise 16** (Deductive ontology)

Download and read the document describing the deductive ontology introduced in the lecture.<sup>2</sup>

Formalize the part of the deductive ontology describing the concepts

- Satisfiable,
- Theorem,
- WeakerTheorem,
- Equivalent,

---

<sup>1</sup>The input format follows the Manchester syntax. To get an idea of what such a formalization might look like, you can review the formalization of the family ontology from the lecture at <http://www.informatik.uni-freiburg.de/~ki/teaching/ws0809/lccai/family.het>.

<sup>2</sup><http://www.cs.miami.edu/~tptp/cgi-bin/SeeTPTP?Category=Documents&File=SZSontology>

- TautologousConclusion,
- EquivalentTheorem,
- Tautology,
- ContradictoryAxioms,
- SatisfiableConclusionContradictoryAxioms,
- TautologousConclusionContradictoryAxioms, and
- NoConsequence,

i.e., the left half of the graphic depicting the deductive ontology, using Manchester syntax. Follow these steps:

- Introduce basis concepts describing the status of the axioms (valid, satisfiable, unsatisfiable), the status of the conjecture (valid, satisfiable, unsatisfiable), and the possible entailment relations between the axioms and the conjecture (all models of the axioms are models of the conjecture, some models of the axioms are models of the conjecture, etc).
- For these basis concepts, formalize all subsumption, equivalence and disjointness relations that you are aware of.
- Define the eleven concepts listed above as intersections of (complements of) basis concepts. Follow the definitions of the concepts given in the Section *Deductive Statuses* of the document describing the deductive ontology.

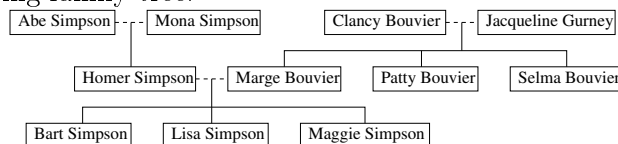
### Exercise 17 (Family ontology)

Download the family ontology from the lecture website<sup>1</sup>.

- Define roles *hasParent*, *siblingOf*, *relativeOf*, and *ancestorOf*.
- Define cousin relations (cf. <http://en.wikipedia.org/wiki/Cousin>) up to second degree cousins and two removes.
- Where applicable, state role hierarchies, reflexivity and transitivity of roles, inverse roles, and role compositions.

### Exercise 18 (Reasoning with Protégé and Pellet)

- Extend the family ontology from Exercise 4.1 with the individual and role assertions (only marriage and parent-child relations) depicted in the following family tree.



<sup>1</sup><http://www.informatik.uni-bremen.de/agbkb/lehre/ss09/logcat/family.net>  
 (note that the ontology differs from the one mentioned on the last exercise sheet)

- (b) Additionally, state some implied facts (persons being grandparents, implied relations, etc.) and mark them with `%implies`.
- (c) Use Hets (latest build) to prove the implied facts. It is also possible to translate the Hets input file into an OWL file that can be parsed by Protégé by running `hets -o owl Family.het`.

## 2.2.2 Extensions of $\mathcal{ALC}$

**Definition 2.2.9** (The logic  $\mathcal{ALCQ}$ ).  $\mathcal{ALCQ}$  extends the sentences of  $\mathcal{ALC}$  with three cases:

*(Hets) Manchester syntax*

$$\begin{aligned}
C ::= & \dots \\
& | \leq nR.C \text{ for } R \in \mathbf{R} \quad R \text{ max } n \ C \\
& \quad \{x \in \Delta^{\mathcal{I}} \mid n \geq |\{y \in \Delta^{\mathcal{I}} \mid (x, y) \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\}|\} \\
& | \geq nR.C \text{ for } R \in \mathbf{R} \quad R \text{ min } n \ C \\
& \quad \{x \in \Delta^{\mathcal{I}} \mid n \leq |\{y \in \Delta^{\mathcal{I}} \mid (x, y) \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\}|\} \\
& | = nR.C \text{ for } R \in \mathbf{R} \quad R \text{ exactly } n \ C \\
& \quad \{x \in \Delta^{\mathcal{I}} \mid n = |\{y \in \Delta^{\mathcal{I}} \mid (x, y) \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\}|\}
\end{aligned}$$

**Definition 2.2.10** (The logic  $\mathcal{ALCO}$ ).  $\mathcal{ALCO}$  extends  $\mathcal{ALC}$  by nominals.

*(Hets) Manchester syntax*

$$\begin{aligned}
C ::= & \dots \\
& | \{a_1, \dots, a_n\} \quad a_1, \dots, a_n
\end{aligned}$$

Complexity of  $\mathcal{ALCO}$ : PSpace/ExpTime-complete

**Definition 2.2.11** (The logic  $\mathcal{S}$ ).  $\mathcal{S}$  extends  $\mathcal{ALC}$  by transitivity of roles

*(Hets) Manchester syntax*

$$\begin{aligned}
\phi ::= & \dots \\
& | \text{Tra}(R) \quad R \text{ Transitive} \\
& \quad \text{Semantics: } R^{\mathcal{I}} \text{ transitive} \\
& \quad \text{Example: } \text{Tra}(\text{Ancestor})
\end{aligned}$$

**Definition 2.2.12** (The logic  $\mathcal{SI}$ ).  $\mathcal{SI}$  extends  $\mathcal{S}$  by inverse roles

$$\begin{aligned}
R ::= & T \dots T \in R \\
& | R^{-} \\
& \quad \text{Semantics: } (R^{-})^{\mathcal{I}} = \{(y, x) \mid (x, y) \in R^{\mathcal{I}}\}
\end{aligned}$$

Complexity of  $\mathcal{SI}$ : PSpace/ExpTime-complete

**Definition 2.2.13** (The logic  $\mathcal{SH}$ ).  $\mathcal{SH}$  extends  $\mathcal{S}$  by role hierarchies

$$\begin{aligned}
R ::= & \dots \\
& | R_1 \sqsubseteq R_2
\end{aligned}$$

Complexity of  $\mathcal{SH}, \mathcal{SHI}, \mathcal{SHIQ}, \mathcal{SHOI}, \mathcal{SHOQ}$ : ExpTime/ExpTime-complete

Within  $\mathcal{SH}$  TBoxes can be internalized:

- Introduce a role  $U$  the “universal role”

- For any Role  $R$ :  $R \sqsubseteq U$  and  $R^- \sqsubseteq U$
- $\text{Tra}(U)$

$C \sqsubseteq D$  iff  $T \sqsubseteq \neg C \sqcup D$

A TBox  $\mathcal{T}$  can be coded into a single concept

$$C_T ::= \bigcap_{D \sqsubseteq E_s} \neg D \sqcup E$$

TBox-Reasoning:  $T \models \text{sat}(D)$  iff  $\text{sat}(D \sqcap C_T \sqcap \forall U.C_T)$

**Definition 2.2.14.** *SR<sub>Q</sub>IQ introduces the following new Sentences.*

	(Hets) Manchester syntax	Semantics
$\phi ::= \dots$		
$R \sqsubseteq S$	ObjectProperty: $R$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
	SubPropertyOf: $S$	
$R_1; \dots; R_n \sqsubseteq R$	ObjectProperty: $R$	$R_1^{\mathcal{I}} \circ \dots \circ R_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
	SubPropertyChain $R_1 \circ \dots \circ R_n$	
$\text{Dis}(R_1, R_2)$	ObjectProperty: $R_1$ Disjoint $R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}} = \emptyset$
$\text{Ref}(R)$	ObjectProperty: $R$ Reflexive	$\forall x \in \Delta^{\mathcal{I}}. R^{\mathcal{I}}(x, x)$
$\text{Irr}(R)$	ObjectProperty: $R$ Irreflexive	$\forall x \in \Delta^{\mathcal{I}}. \neg R^{\mathcal{I}}(x, x)$
$\text{Asy}(R)$	ObjectProperty: $R$ Asymmetric	$\forall x, y \in \Delta^{\mathcal{I}}$ $R(x, y) \rightarrow R(y, x)$

where  $R \circ S = \{(x, z) \mid \exists y. (x, y) \in R, (y, z) \in S\}$

the following new concept

$$C ::= \dots \quad \mid \exists R. \text{Self} \quad (\exists R. \text{Self})^{\mathcal{I}} = \{x \mid x \in \Delta^{\mathcal{I}}, (x, x) \in R^{\mathcal{I}}\} \quad \text{and the univer-}$$

$$\text{sal role } R ::= \dots \quad \mid U \quad U^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$$

Axioms about roles are collected in the RBox.

Complexity of SR<sub>Q</sub>IQ: N2ExpTime-complete (nondeterministic runtime  $O(2^{2^n})$ ), det. runtime  $O(2^{2^{2^n}})$

### 2.2.3 Signature morphisms

**Definition 2.2.15.** *Given two DL signatures  $\Sigma_1 = (C_1, R_1, I_1)$  and  $\Sigma_2 = (C_2, R_2, I_2)$  a signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  consists of three functions*

- $\sigma^C : C_1 \rightarrow C_2$ ,
- $\sigma^R : R_1 \rightarrow R_2$ ,
- $\sigma^I : I_1 \rightarrow I_2$ .

**Definition 2.2.16.** *Given a signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  and a  $\Sigma_1$ -sentence  $\phi$ , the translation  $\sigma(\phi)$  is defined by inductively replacing the symbols<sup>1</sup> in  $\phi$  along  $\sigma$ .*

<sup>1</sup>Note that  $\phi$  and  $\sigma(\phi)$  live in the same DL.

**Definition 2.2.17.** Given a signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  and a  $\Sigma_2$ -model  $\mathcal{I}_2$ , the  $\sigma$ -reduct of  $\mathcal{I}_2$  along  $\sigma$   $\mathcal{I}_2|_\sigma$  is the  $\Sigma_1$ -model  $\mathcal{I}_1$  defined by

- $\Delta^{\mathcal{I}_1} = \Delta^{\mathcal{I}_2}$
- $A^{\mathcal{I}_1} = \sigma^C(A)^{\mathcal{I}_2}$ , for  $A \in \mathbf{C}_1$
- $R^{\mathcal{I}_1} = \sigma^R(R)^{\mathcal{I}_2}$ , for  $R \in \mathbf{R}_1$
- $a^{\mathcal{I}_1} = \sigma^I(a)^{\mathcal{I}_2}$ , for  $a \in \mathbf{I}_1$

**Proposition 2.2.18** (satisfaction condition). Given  $\sigma : \Sigma_1 \rightarrow \Sigma_2$ ,  $\phi_1 \in \text{Sen}(\Sigma_1)$  and  $\mathcal{I}_2 \in \text{Mod}(\Sigma_2)$

$$\mathcal{I}_2|_\sigma \models \phi_1 \quad \text{iff} \quad \mathcal{I}_2 \models \sigma(\phi_1)$$

*Proof.* Induction over the structure of  $\phi_1$  □

## 2.2.4 Freeness

**Definition 2.2.19.** Given  $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$  and  $\Sigma$ -models  $\mathcal{I}$  and  $\mathcal{I}'$  we say that  $\mathcal{I} \leq \mathcal{I}'$  if

- $\Delta^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}'}$
- $A^{\mathcal{I}} \subseteq A^{\mathcal{I}'}$  for  $A \in \mathbf{C}$
- $R^{\mathcal{I}} \subseteq R^{\mathcal{I}'}$  for  $R \in \mathbf{R}$
- $a^{\mathcal{I}} = a^{\mathcal{I}'}$  for  $a \in \mathbf{I}$

**Definition 2.2.20.** Given a theory  $T = (\Sigma, \Gamma)$ ,

$$\text{Mod}(\text{free}\{T\}) = \{M \in \text{Mod}(T) \mid M \text{ is the least model in } \text{Mod}(T)\}$$

$$\text{Mod}(\text{cofree}\{T\}) = \{M \in \text{Mod}(T) \mid M \text{ is the greatest model in } \text{Mod}(T)\}$$

Given theories  $SP_1 = (\Sigma_1, \Gamma_1)$  and  $SP_2 = (\Sigma_2, \Gamma_2)$  and a signature morphism  $\sigma : SP_1 \rightarrow SP_2$ ,

$$\begin{aligned} \text{Mod}(SP_1 \text{ then } \text{free}\{SP_2\}) = \\ \{M \in \text{Mod}(SP_1 \text{ then } SP_2) \mid M \text{ is the least model in} \\ \text{Mod}(SP_1 \text{ then } SP_2) \text{ with the same } \sigma\text{-reduct as } M\} \end{aligned}$$

$$\begin{aligned} \text{Mod}(SP_1 \text{ then } \text{cofree}\{SP_2\}) = \\ \{M \in \text{Mod}(SP_1 \text{ then } SP_2) \mid M \text{ is the greatest model in} \\ \text{Mod}(SP_1 \text{ then } SP_2) \text{ with the same } \sigma\text{-reduct as } M\} \end{aligned}$$

**Example 2.2.21.** *Defintion of a Man with only male descendant (Momd).*

```

1 spec Momd =
2   Class Man
3   ObjectProperty hasChild
4 then free {
5   Class Momd
6     EquivalentTo Man and hasChild only Momd
7 }
8 end

```

By the closed world assumption Momd only contains those individuals that are forced to belong to Momd by the axioms:

- (a) All men without children (in  $M|_{SP_1} = M|_{\sigma}$ )
- (b) All men having only children that belong to (1)
- (c) All men having only children that belong to (2)
- (d) ...

**Example 2.2.22.** *Transitive closure.*

```

1 spec Ancestor =
2   ObjectProperty hasParent
3 then free {
4   ObjectProperty ancestor
5     Characteristics: Transitive
6     SuperPropertyOf: hasParent
7 }
8 end

```

Without free, ancestor can be interpreted as the union of two different hasParent relations or even as the universal role. With free, ancestor is always the transitive closure of hasParent.

**Example 2.2.23.** *Defintion of Momd (without) free.*

```

1 spec Momd2 =
2   Class Man
3   ObjectProperty hasChild
4 then free {
5   ObjectProperty decendant
6     Characteristics: Transitive
7     SuperPropertyOf: hasChild
8 }
9 then
10  Class Momd
11    EquivalentTo Man and decendant only Man
12 end

```

It is not possible to define the transitive closure of a role without free (or without defining it in the logic).

## Encoding of Free and Cofree

Logic	How to support free and cofree
Propositional Logic	Quantified Boolean Formulae
$\mathcal{ALC}$	$\mu\mathcal{ALC}$
FOL	$\mu\text{FOL}$ , HOL

Table 2.2: Support for free and cofree

As Table 2.2 shows, more complex logics are needed to encode free and cofree in some logics.

**Example 2.2.24.** *Encoding of free in Propositional Logic.*

```

1 spec Liar =
2   prop mary
3 then free {
4   props harry, john
5     . harry => john
6     . john & mary => harry
7 }
8 end

```

This can be encoded in the following QBF:

$$\forall \text{john}', \text{harry}' . ((\text{harry}' \rightarrow \text{john}') \wedge (\text{john}' \wedge \text{mary} \rightarrow \text{harry}')) \\ \rightarrow (\text{harry} \rightarrow \text{harry}') \wedge (\text{john} \rightarrow \text{john}')$$

The cofree variant can be encoded like this:

$$\forall \text{john}', \text{harry}' . ((\text{harry}' \rightarrow \text{john}') \wedge (\text{john}' \wedge \text{mary} \rightarrow \text{harry}')) \\ \rightarrow (\text{harry}' \rightarrow \text{harry}) \wedge (\text{john}' \rightarrow \text{john})$$

**Definition 2.2.25.**  $\mu\mathcal{ALC}$  (*ExpTime-complete*)

$$C ::= \dots \\ | X \\ | \mu X.C \\ | \nu X.C$$

**Example 2.2.26.** *Encoding of Example 2.2.21 in  $\mu\mathcal{ALC}$ :*

$$\text{Momd} = \mu X. \text{Man} \sqcap \forall \text{hasChild}. X$$

**Definition 2.2.27.**  $\mathcal{EL}$  (*Reasoner Hyb can solve cyclic TBoxes in P with greatest fixpoint semantic.*)

$$C ::= \top \\ | C \sqcap C \\ | \exists R.C \\ | \neg A$$

**Example 2.2.28** ((co)inductive datatypes).

```
1 Node  $\sqsubseteq$   $\exists$ hasValue.T
2 then free {
3   e : EmptyTree
4   Tree  $\equiv$  EmptyTree  $\sqcup$  (Node  $\sqcap$  ( $\leq 1$  child.T)  $\sqcap$   $\exists$ child.T  $\sqcap$   $\forall$ child.Tree)
5 }
```

*Trees are inductive datatypes that are defined bottom-up (from leaves to root) and allow induction over their structure. If this would be cofree instead of free Tree could be interpreted as the universal concept.*

```
1 Node  $\sqsubseteq$   $\exists$ hasValue.T
2 then cofree {
3   Stream  $\equiv$  Node  $\sqcap$  ( $\leq 1$  succ.T)  $\sqcap$   $\exists$ succ.Stream
4 }
```

*Streams (infinite lists) are coinductive datatypes that are defined top-down (coinduction instead of induction). If this would be free instead of cofree Stream could be interpreted as the empty concept.*

**Exercise 19** (Cyclic TBoxes)

Specify the following statements, possibly using cyclic TBoxes. If you use a cyclic TBox, be careful to correctly choose least or greatest fixpoint semantics, according to what is needed in the examples.

- (a) A chess fanatic is a chess player all of whose friends are chess fanatics.
- (b) A Gmail user has been invited by some Gmail user.
- (c) A folder is an inode that contains inodes, all of which are files, folders or devices.



## 2.2.5 Conservative Extensions

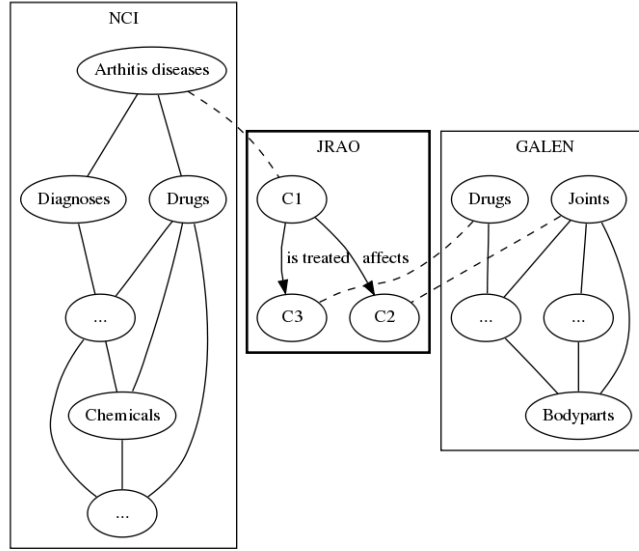


Figure 2.4: Building the Ontology JRAO using modules from NCI and GALEN

**Example 2.2.29.** *JRAO (cf Figure 2.4) is constructed using fragments of the two existing ontologies NCI and GALEN. NCI and GALEN are too large for a complete import. The Goal is to only import interesting modules (called ontology modules).*

**Definition 2.2.30.** *A theory morphism  $\sigma : T_1 \rightarrow T_2$ ,  $T_1 = (\Sigma_1, \Gamma_1)$ ,  $T_2 = (\Sigma_2, \Gamma_2)$  is a signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  such that  $M_2 \in \text{Mod}(T_2)$  implies  $M_2|_{\sigma} \in \text{Mod}(T_1)$*

**Example 2.2.31.**  *$T_1 \hookrightarrow T_1 \cup T_2$  is always a theory morphism.*

**Definition 2.2.32.** *A theory morphism  $\sigma : T_1 \rightarrow T_2$  is consequence-theoretically conservative<sup>1</sup>, if for all  $\phi \in \text{Sen}(\Sigma_1)$*

$$T_2 \models \sigma(\phi) \quad \text{implies} \quad T_1 \models \phi$$

**Definition 2.2.33.** *An ontology module  $T_1$  of an ontology  $T_2$  is a subtheory  $T_1 \subseteq T_2$ , such that  $\sigma : T_1 \hookrightarrow T_2$  is conservative.*

**Definition 2.2.34.** *A theory morphism  $\sigma : T_1 \rightarrow T_2$  is model-theoretically conservative, if each  $T_1$ -model  $M_1$  has a  $\sigma$  expansion to a  $T_2$ -model  $M_2$ , i.e.*

$$M_2|_{\sigma} = M_1$$

<sup>1</sup>This is the same definition as for Propositional Logic (cf. Definition 2.1.29) but with a different meaning of  $T, \phi, \sigma, \dots$

**Proposition 2.2.35.** *For any DL, model theoretical conservativity implies consequence theoretical conservativity.*

*Proof.* (cf. theorem 2.1.33) □

**Proposition 2.2.36.** *Let  $T_1, T_2$  be TBoxes in  $\mathcal{ALC}$ ,  $\mathcal{ALCQ}$ ,  $\mathcal{ALCO}$  or  $\mathcal{ALCOQ}$ . Then  $\sigma : T_1 \rightarrow T_2$  is consequence theoretically conservative iff the following implication holds: If  $C$  is satisfiable relative to  $T_1$  (i.e. there is a  $T_1$ -model making  $C$  non-empty) then  $\sigma(C)$  is satisfiable relative to  $T_2$*

*Proof.*

$\Rightarrow$  : Assume  $\sigma : T_1 \rightarrow T_2$  is consequence theoretically conservative (1) and  $C$  is satisfiable relative to  $T_1$  (2).

To show:  $\sigma(C)$  is satisfiable relative to  $T_2$ .

$$\begin{array}{ll} \text{By (2)} & T_1 \not\models C \sqsubseteq \perp \\ \text{By (1) we get} & T_2 \not\models \sigma(C \sqsubseteq \perp) \\ \text{and thus} & T_2 \not\models \sigma(C) \sqsubseteq \perp. \end{array}$$

Hence  $\sigma(C)$  is satisfiable relative to  $T_2$ .

$\Leftarrow$  : (a) Subsumption:  $T_2 \models \sigma(C \sqsubseteq D)$

$$\begin{array}{l} T_2 \models \sigma(C \sqsubseteq D) \\ \text{iff } T_2 \models \sigma(C) \sqsubseteq \sigma(D) \\ \text{iff } \sigma(C) \sqcap \neg\sigma(D) \text{ is unsatisfiable relative to } T_2 \\ \text{iff } \sigma(C \sqcap \neg D) \text{ is unsatisfiable relative to } T_2 \\ \text{implies } C \sqcap \neg D \text{ is unsatisfiable relative to } T_1 \\ \text{iff } T_1 \models C \sqsubseteq D \end{array}$$

(b) Individuals:  $T_2 \models \sigma(a : C)$

cases:  $\mathcal{ALCO}$ ,  $\mathcal{ALCOQ}$ :

$$\begin{array}{ll} & T_2 \models \sigma(a : C) \\ \text{iff} & T_2 \models \sigma(\{a\} \sqsubseteq C) \\ \text{implies (by (a))} & T_1 \models \{a\} \sqsubseteq C \\ \text{iff} & T_1 \models a : C \end{array}$$

cases:  $\mathcal{ALC}$ ,  $\mathcal{ALCQ}$ :

$$\begin{array}{ll} & T_2 \models \sigma(a : C) \\ \text{iff (since } a \text{ does not appear in } T_2) & T_2 \models \top \sqsubseteq \sigma(C) \\ \text{implies (by (a))} & T_1 \models \top \sqsubseteq \sigma(C) \\ \text{iff} & T_1 \models a : C \end{array}$$

(b) Roles:  $T_2 \models \sigma(R(a, b))$

cases:  $\mathcal{ALCO}$ ,  $\mathcal{ALCOQ}$ :

$$\begin{aligned}
& T_2 \models \sigma(R(a, b)) \\
& \text{iff } T_2 \models \sigma(\{a\} \sqsubseteq \exists R.\{b\}) \\
& \text{iff } T_2 \models \sigma(\{a\}) \sqsubseteq \sigma(\exists R.\{b\}) \\
& \text{implies (by (a)) } T_1 \models \{a\} \sqsubseteq \exists R.\{b\} \\
& \text{iff } T_1 \models R(a, b)
\end{aligned}$$

cases:  $\mathcal{ALC}$ ,  $\mathcal{ALCQ}$ :

$$\begin{aligned}
& T_2 \models \sigma(R(a, b)) \\
& \text{iff* } T_2 \models \top \sqsubseteq \perp \quad (\text{"}T_2 \text{ has no model.}") \\
& \text{iff } T_2 \models \sigma(\top) \sqsubseteq \sigma(\perp) \\
& \text{implies (by (a)) } T_1 \models \top \sqsubseteq \perp \\
& \text{iff* } T_1 \models R(a, b)
\end{aligned}$$

(\*) If  $T_2$  had a model, by the tree model property<sup>1</sup>, and since  $a$  and  $b$  do not occur in  $T_2$ , we can swap the interpretations of  $\sigma(a)$  and  $\sigma(b)$ , we can modify the  $T_2$ -model to a  $T_2$ -model that does not satisfy  $\sigma(R(a, b))$ . If  $T_2$  has no model, any sentence follows from it (in particular  $\sigma(R(a, b))$ ).

□

### Example 2.2.37.

```

1 spec  $T_2 =$ 
2    $WebService \sqsubseteq \exists provider.\top \sqcap \exists input.\top \sqcap \exists output.\top \text{ } \% (a) \%$ 
3 then
4    $WebService \sqsubseteq \exists input.Integer \sqcap \exists input.Array \text{ } \% (b) \%$ 
5    $Integer \sqsubseteq \neg Array \text{ } \% (c) \%$ 
6 end

```

$T_1$  is the theory consisting of the axiom (a);  $T_2$  is the complete theory with the axioms (a), (b) and (c).

$T_2$  is not consequence theoretically conservative over  $T_1$  in  $\mathcal{ALCQ}$ . This is shown by giving a witness concept that is satisfiable in  $T_1$  but not in  $T_2$ .

Here the witness concept is:  $WebService \sqcap (\leq_1 input.\top)$

However, the extension is consequence theoretically conservative in  $\mathcal{ALC}$ .

*Proof.* Let  $C$  be satisfiable relative to  $T_1$ , i.e. let  $\mathcal{I}$  be a  $T_1$ -model with  $C^{\mathcal{I}} \neq \emptyset$ . Let  $\mathcal{I}'$  be obtained from  $\mathcal{I}$  by

$$\bullet \Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}} \uplus \Delta^{\mathcal{I}}, \text{ say } \Delta^{\mathcal{I}} \xrightarrow{\Pi_1} \Delta^{\mathcal{I}} \amalg \Delta^{\mathcal{I}} \xleftarrow{\Pi_2} \Delta^{\mathcal{I}}$$

<sup>1</sup>Relations form a tree, so  $R(a, b)$  implies  $\neg R(b, a)$ . Shown only for  $\mathcal{ALC}$  and  $\mathcal{ALCQ}$ .

- Idea: create two copies of each individual. Interpret one copy as `Integer` and the other one as `Array`.
- $\Pi_i$  is an unambiguous injection like  $\Pi_i(x) = \langle x, i \rangle$
- $C^{\mathcal{I}'} = \Pi_1(C^{\mathcal{I}}) \cup \Pi_2(C^{\mathcal{I}})$ 
  - In set notation:  $C^{\mathcal{I}'} = \{\Pi_1(x) \mid x \in C^{\mathcal{I}}\} \cup \{\Pi_2(x) \mid x \in C^{\mathcal{I}}\}$
- $R^{\mathcal{I}'} = \bigcup_{i,j \in \{1,2\}} \Pi_i \times \Pi_j(R^{\mathcal{I}})$ 
  - In set notation:  $R^{\mathcal{I}'} = \bigcup_{i,j \in \{1,2\}} \{(\Pi_i(x), \Pi_j(y)) \mid (x, y) \in R^{\mathcal{I}}\}$

Then  $\mathcal{I}'$  is a  $T_1$ -model making  $C$  non-empty. Extend  $\mathcal{I}'$  to a  $T_2$ -model  $\mathcal{I}''$  by putting

- $\text{Integer}^{\mathcal{I}''} = \Pi_1(\Delta^{\mathcal{I}'})$
- $\text{Array}^{\mathcal{I}''} = \Pi_2(\Delta^{\mathcal{I}'})$

This way each webserver has at least one integer and one array input.  $C^{\mathcal{I}''} \neq \emptyset$  □

### Complexity:

- consequence-theoretical conservativity in
  - $\mathcal{ALCQI}$ : 2ExpTime-complete
  - $\mathcal{ALCQIO}$ : undecidable
  - $\mathcal{EL}$ : ExpTime-complete
- model-theoretical conservativity in
  - $\mathcal{EL}$ : undecidable
  - $\mathcal{ALC}$ :  $\Pi_1^1$ -complete

### Exercise 20 (Conservative extension)

Consider the following description logic theories.

- (a)
- |                       |               |  |
|-----------------------|---------------|--|
| <code>Lecture</code>  | $\sqsubseteq$ | $\exists \text{has\_subject. Subject} \sqcap \exists \text{given\_by. Lecturer}$ |
| <code>Intro_AI</code> | $\sqsubseteq$ | <code>Lecture</code>   |

Is the following a conservative extension?

- |   |               |   |
|---|---------------|---|
| <code>Intro_AI</code>                                   | $\sqsubseteq$ | $\exists \text{has\_subject. Logic}$          |
| <code>Intro_AI</code>                                   | $\sqsubseteq$ | $\exists \text{has\_subject. NeuralNetworks}$ |
| <code>Logic</code> $\sqcap$ <code>NeuralNetworks</code> | $\sqsubseteq$ | $\perp$                                       |

(b)

Penguin  $\sqsubseteq$  Bird  
Bird  $\sqsubseteq$  LivingBeing

Is the following a conservative extension?

Bird  $\sqsubseteq$  Animal  
Animal  $\sqsubseteq$  LivingBeing

(c)

Penguin  $\sqsubseteq$  Bird

Is the following a conservative extension?

Bird  $\sqsubseteq$  CanFly  
Penguin  $\sqsubseteq$   $\neg$ CanFly

In each case, consider different possible senses of “conservative extension”. If applicable, construct a witness concept.

## 2.3 First-Order Logic

- expressive general-purpose language
- roots are:
  - Aristotelian Syllogisms
  - Boole
  - Frege’s Begriffsschrift
- Axiomatic method: description of objects by their properties
  - Euclids geometry axioms
  - abstract algebra (groups, fields, vector spaces)
  - set theory (ZFC, VNBG) (foundation of mathematics and theoretical computer-science)
- specification of software
- language for upper ontologies
- logic programming
- Planning languages (PDDL)

### 2.3.1 Foundations

**Definition 2.3.1.** A Signature  $\Sigma = (S, F, P)$  of many-sorted-FOL consists of:

- a set  $S$  of sorts, where  $S^*$  is the set of words over  $S$
- for each  $w \in S^*$ , and each  $s \in S$  a set  $F_{w,s}$  of function symbols (here  $w$  are the argument sorts and  $s$  are the result sorts)
- for each  $w \in S^*$  a set  $P_w$  of predicate symbols

**Example 2.3.2.** “This spear is so sharp, that it can pierce through anything”  
 “This shield is so strong, that nothing can destroy it.” Formalised in Hets as:

```

1 sorts : Object
2 ops  : Spear: Object; Shield:Object
3 preds : pierce: Object x Object
4       destroy: Object x Object
5       . forall x: Object . pierce (Spear, x)
6       . not exists x: Object . destroy(x, Shield)
7       . forall x,y: Object . pierce (x,y) --> destroy (x,y)
  
```

**Definition 2.3.3.** Given a Signature  $\Sigma = (S, F, P)$  the set of ground  $\Sigma$ -terms is inductively defined by:  $f_{w,s}(t_1, \dots, t_n)$  is a term of sort  $s$ , if each  $t_i$  is a term of sort  $s_i$  ( $i = 1 \dots n, w = S_1 \dots S_n$ ) and  $f \in F_{w,s}$ .

In particular (for  $n = 0$ ) this means, that constants  $w = \lambda^1$  are terms

**Definition 2.3.4.** Given a signature  $\Sigma = (S, F, P)$  the set of  $\Sigma$ -sentences is inductively defined by:

- $t_1 = t_2$  for  $t_1, t_2$  of the same sort
- $P_w(t_1, \dots, t_n)$  for  $t_i$   $\Sigma$ -term of sort  $s_i$ , ( $1 \leq i \leq n, w = s_1, \dots, s_n, p \in P_w$ )
- $\phi_1 \wedge \phi_2$  for  $\phi_1, \phi_2$   $\Sigma$ -formulae
- $\phi_1 \vee \phi_2$  for  $\phi_1, \phi_2$   $\Sigma$ -formulae
- $\phi_1 \rightarrow \phi_2$  for  $\phi_1, \phi_2$   $\Sigma$ -formulae
- $\phi_1 \leftrightarrow \phi_2$  for  $\phi_1, \phi_2$   $\Sigma$ -formulae
- $\neg \phi_1$  for  $\phi_1$   $\Sigma$ -formula
- $\top$

---

<sup>1</sup>empty word

- $\perp$
- $\forall x : s . \phi$  if  $s \in S$ ,  $\phi$  is a  $\Sigma + \{x : s\}$ -sentence where  $\Sigma + \{x : s\}$  is  $\Sigma$  enriched with a new constant  $x$  of sort  $s$
- $\exists x : s . \phi$  likewise

**Exercise 21** (Formalization in First-Order Logic)

Translate the following into First-Order Logic. Explain the meanings of the names, predicates, and functions you use, and comment on any shortcomings in your translations.

- There's a sucker born every minute.
- Whither thou goest, I will go.
- Soothsayers make a better living in the world than truthsayers.
- To whom nothing is given, nothing can be required.
- If you always do right, you will gratify some people and astonish the rest.

**Exercise 22** (First-Order Theorem Proving)

Use SPASS to prove that the following statements are tautologies:

- There is some  $x$  such that if  $x$  is a  $P$  then everything is a  $P$ .
- There cannot be any barber (who is a man) who shaves every man who does not shave himself.

**Exercise 23** (Satisfiability of Specifications)

Consider the specification of classical extensional parthood<sup>1</sup>. Assume you want to prove that this specification is consistent (right click on the node, node menu, check consistency, CASL  $\rightarrow$  SoftFOL (darwin)).

- Install the latest HETS binary<sup>2</sup>, the DARWIN THEOREM PROVER<sup>3</sup> and the E EQUATIONAL THEOREM PROVER<sup>4</sup>.
- Simplify the task of checking consistency by decomposing the specification in such a way that the whole specification can be seen as a conservative extension of a smaller specification and it is sufficient to prove consistency of the smaller specification and conservativity of the extension.
- Use DARWIN to find some models for the specification (say the three smallest models). For the smallest model, simply run DARWIN from HETS as described above. To find larger models of size  $n$ , add to your

---

<sup>1</sup><http://www.informatik.uni-freiburg.de/~ki/teaching/ws0809/ailogic/CEP.casl>  
<sup>2</sup><http://www.dfki.de/sks/hets/>  
<sup>3</sup><http://combination.cs.uiowa.edu/Darwin/>  
<sup>4</sup><http://www.e prover.org/>

specification axioms stating that there should be  $n$  pairwise distinct elements in the model. Can you come up with a conjecture regarding the general form of a model for `CLASSICAL_EXTENSIONAL_PARTHOOD`? How many elements can a model have?

### Exercise 24 (Induction Proofs)

Consider the definitions of natural numbers, lists, and trees in the file `Datatypes.casl`<sup>5</sup>. Use ISABELLE<sup>6</sup> or SPASS<sup>7</sup> to prove the implied sentences.

### 2.3.2 Signature Morphisms

**Definition 2.3.5.** Given a signature  $\Sigma = (S, F, P)$  a  $\Sigma$ -model  $M$  consists of

- a carrier set  $M_s \neq \emptyset$  for each sort  $s \in S$
- a function  $f_{w,s}^m : M_{s_1} \times \dots \times M_{s_n} \rightarrow M_s$  for each  $f \in F_{w,s}$ ,  $w = s_1, \dots, s_n$ .  
In particular, for a constant, this is just an element of  $M_s$
- a relation  $p_w^M \subseteq M_{s_1} \times \dots \times M_{s_n}$  for each  $p \in P_w$ ,  $w = s_1 \dots s_n$

**Definition 2.3.6.** Given signatures  $\Sigma = (S, F, P)$ ,  $\Sigma' = (S', F', P')$  a signature morphism  $\sigma : \Sigma \rightarrow \Sigma'$  consists of

- a map  $\sigma^S : S \rightarrow S'$
- a map  $\sigma_{w,s}^F : F_{w,s} \rightarrow F'_{\sigma^S(w), \sigma^S(s)}$  for each  $w \in S^*$  and each  $s \in S$
- a map  $\sigma_w^P : P_w \rightarrow P'_{\sigma^S(w)}$  for each  $w \in S^*$

**Definition 2.3.7.** Given a signature morphism  $\sigma : \Sigma \rightarrow \Sigma'$  and a  $\Sigma'$ -model  $M'$ , define  $M = M'|_{\sigma}$  as

- $M_s = M'_{\sigma^S(s)}$
- $f_{w,s}^M = \sigma_{w,s}^F(f)_{\sigma^S(w), \sigma^S(s)}^{M'}$
- $p_{w,s}^M = \sigma_w^P(p)_{\sigma^S(w)}^{M'}$

**Definition 2.3.8.** A  $\Sigma$ -term  $t$  is evaluated in a  $\Sigma$ -model  $M$  as follows:

$$M(f_{w,s}(t_1, \dots, t_n)) = f_{w,s}^M(M(t_1), \dots, M(t_n))$$

<sup>5</sup><http://www.informatik.uni-freiburg.de/~ki/teaching/ws0809/aillogic/Datatypes.casl>

<sup>6</sup><http://isabelle.in.tum.de/>

<sup>7</sup><http://www.spass-prover.org/>



**Definition 2.3.9** (satisfaction of sentences).

$$M \models t_1 = t_2 \text{ iff } M(t_1) = M(t_2)$$

$$M \models p_w(t_1 \dots t_n) \text{ iff } (M(t_1), \dots, M(t_n)) \in p_w^M$$

$$M \models \phi_1 \wedge \phi_2 \text{ iff } M \models \phi_1 \text{ and } M \models \phi_2$$

$$M \models \forall x : s. \phi \text{ iff for all } \sigma\text{-expansions } M' \text{ of } M, M' \models \phi$$

where  $\sigma : \Sigma \hookrightarrow \Sigma + \{x : s\}$  is the inclusion.

$$M \models \exists x : s. \phi \text{ iff there is a } \sigma\text{-expansion } M' \text{ of } M \text{ such that } M \models M'$$

**Definition 2.3.10.** Given  $\Gamma \subseteq \text{Sen}(\Sigma)$ ,  $\phi \in \text{Sen}(\Sigma)$ ,  $\Gamma \models \phi$  iff for all  $\Sigma$ -models  $M$

$$M \models \Gamma \text{ implies } M \models \phi$$

**Example 2.3.11.**  $\Gamma_{Chu} \models \perp$

That is,  $\Gamma_{Chu}$  from the last example has no models (it is unsatisfiable)

**Definition 2.3.12.** Given a signature morphism  $\sigma : \Sigma \rightarrow \Sigma'$  and  $\phi \in \text{Sen}(\Sigma)$  the translation  $\sigma(\phi)$  is defined inductively by:

$$\sigma(f_{w,s}(t_1 \dots t_n)) = \sigma_{w,s}^F(f_{\sigma(w),\sigma(s)})(\sigma(t_1) \dots \sigma(t_n))$$

$$\sigma(t_1 = t_2) = \sigma(t_1) = \sigma(t_2)$$

$$\sigma(p_w(t_1 \dots t_n)) = \sigma_w^P(p)_{\sigma^S(w)}(\sigma(t_1) \dots \sigma(t_n))$$

$$\sigma(\phi_1 \wedge \phi_2) = \sigma(\phi_1) \wedge \sigma(\phi_2)$$

$$\sigma(\forall x : s. \phi) = \forall x : \sigma^S(s). (\sigma + id)(\phi)$$

$$\sigma(\exists x : s. \phi) = \exists x : \sigma^S(s). (\sigma + id)(\phi)$$

where  $(\sigma + id) : \Sigma + \{x : s\} \rightarrow \Sigma' + \{x : s\}$  acts like  $\sigma$  on  $\Sigma$  and is the identity on  $x$ .

**Proposition 2.3.13** (satisfaction condition). For a signature morphism  $\sigma : \Sigma \rightarrow \Sigma'$ ,  $\phi \in \text{Sen}(\Sigma)$ ,  $M' \in \text{Mod}(\Sigma')$ :

$$M'|_{\sigma} \models \phi \text{ iff } M' \models \sigma(\phi)$$

*Proof.* This can easily be shown by induction □

**Definition 2.3.14.**  $\text{Mod}(SP \text{ with } \sigma : \Sigma \rightarrow \Sigma') = \{M' \in \text{Mod}(\Sigma') \mid M'|_{\sigma} \in \text{Mod}(SP)\}$

**Proposition 2.3.15.**  $\text{Mod}((\Sigma, \Gamma) \text{ with } \sigma : \Sigma \rightarrow \Sigma') = \text{Mod}(\Sigma', \sigma(\Gamma))$

*Proof.*

$$M'|_{\sigma} \in \text{Mod}(\Sigma, \Gamma) \text{ iff } M'|_{\sigma} \models \Gamma \text{ iff } M' \models \sigma(\Gamma)$$

□

**Exercise 25** (Theory morphisms)

- (a) Show with HETS that  $\text{Monoid} \rightarrow \text{Nat}$  is a theory morphism.
- (b) Specify that classical extensional parthood has models which use the powerset without the empty set.
- (c) Specify another theory morphism of your choice and prove it with HETS.

**2.3.3 Conservative Extensions**

**Definition 2.3.16.** A theory morphism  $\sigma : T_1 \rightarrow T_2$  is a consequence-theoretically conservative extension, if for all  $\phi \in \text{Sen}(\Sigma_1)$

$$T_2 \models \sigma(\phi) \text{ implies } T_1 \models \phi$$

**Definition 2.3.17.** A theory morphism  $\sigma : T_1 \rightarrow T_2$  is a model-theoretically conservative extension, if each  $T_1$ -model  $M_1$  has a  $\sigma$ -expansion to a  $T_2$ -model, i.e.  $M_2 \in \text{Mod}(T_2)$  with  $M_2|_{\sigma} = M_1$ .

**Proposition 2.3.18.** Model-theoretical conservativity implies consequence-theoretical conservativity.

*Proof.* Same as in propositional logic (cf. 2.1.3). □

**Example 2.3.19.** Theory  $T$ :

```
1 sort:   Nat
2 ops:   0:Nat
3        succ: Nat -> Nat
4 pred:  ___ < ___ : Nat x Nat
5        forall x,y,z : Nat
6          . x = 0 \\/ exists u:Nat . succ(u) = x
7          . x < succ(y) <=> (x<y \\/ x = y)
8          . not (x < 0)
9          . x < y => not (y < x)
10         . (x < y /\ y < z) => (x < z)
11         . x < y \\/ x = y \\/ y < x
```

*Models of  $T$ :*

- $\mathbb{N}$  (the standard model)
- $\mathbb{N} + \mathbb{Z}$

*Now extent  $T$  to the theory  $T'$*

**Proposition 2.3.20** (completeness). For any sentence  $\phi$ ,  $T \models \phi$  or  $T \models \neg\phi$ .

```

1 op: ___ + ___ : Nat x Nat -> Nat
2   forall x,y : Nat
3     . 0 + y = y
4     . succ(x) + y = succ(x + y)  %(+succ)%
5     . succ(x) + y > y            %(succ_great)%

```

**Proposition 2.3.21.**  $T'$  is a consequence theoretical conservative extension of  $T$

*Proof.* Suppose that  $T' \models \phi$  but  $T \not\models \phi$ .

By completeness,  $T \models \neg\phi$ , and since  $T'$  is an extension of  $T$   $T' \models \neg\phi$  hence  $T' \models \perp$ . But  $\mathbb{N}$  is a model of  $T'$ .  $\square$

**Proposition 2.3.22.**  $T'$  is no model-theoretically conservative extension of  $T$ .

*Proof.* Suppose that  $\mathbb{N} + \mathbb{Z}$  can be extended to a  $T'$ -model  $M'$

By (succ\_great) we have  $0' +^{M'} 0' >^{M'} 0'$

now let  $n = 0' +^{M'} 0'$

By (+succ) we know that:  $\text{succ}^{M'}(-1' +^{M'} 0') = 0' +^{M'} 0' = n'$

Hence  $-1 +^{M'} 0' = (n - 1)'$

analogous  $-2 +^{M'} 0' = (n - 2)'$

we then get  $-n +^{M'} 0' = 0'$

But, by (succ\_great) we obtain  $-n' +^{M'} 0' > 0'$   $\square$

### 2.3.4 Sort generation constraints

**Definition 2.3.23.** Given a signature  $\Sigma = (S, F, T)$  a  $\Sigma$ -sort generation constraint is a pair

$$(S_0, F_0) \text{ with } s_0 \subseteq S, (F_0)_{w,s} = F_{w,s} \text{ for } w \in s^*, s \in S$$

Intuitively:  $(S_0, F_0)$  expresses that the carriers for the sorts in  $S_0$  are generated by the operations in  $F_0$

**Definition 2.3.24.**  $M \models (S_0, F_0)$  iff for each  $x \in M_s, s \in S_0$  there is an extension  $\Sigma'$  of  $\Sigma$  with constants of sort in  $S \setminus S_0$ , a term  $t$  of sort  $s$  using only symbols from  $F_0$ , and a  $\Sigma'$  expansion of  $M$  such that  $M'(t) = X$ .

**Example 2.3.25.** Definition of a free datatype in Hets:

```

1 sort Elem
2 free type List ::= nil | cons(Elem; List)
3 ({List}, {nil:List, cons: Elem x List -> List})

```

The first two lines expand to:

```

1 generated {
2 sort List
3 ops nil: List
4 cons: Elem x List -> List
5 forall x;y : Elem
6     ll;l2 : List
7 . not nil = cons(x,y)
8 . cons(x,ll) = cons(y,l2) => (x=y & ll = l2)
9 }

```

Models  $M$  are such that  $M_{Elem}$  is any set and

- $M_{List} = M_{Elem}^*$  (words over  $M_{Elem}$ )
- $nil^M = \lambda$
- $cons^M(x, w) = x, w$  (concatenation of words)

If for example  $M_{Elem} = \mathbb{N}$ , then  $2, 20, 4 \in M_{List}$  since

- $\Sigma'$  extends  $\Sigma$  by constants  $a, b, c : Elem$
- $M'(a) = 2, \quad M'(b) = 20, \quad M'(c) = 4$
- $t = cons(a, cons(b, cons(c, nil)))$
- $M'(t) = 2, 20, 4$

### 2.3.5 Proofs

We can use the propositional calculus if we introduce new rules for quantifiers seen in Figure 2.5.

$$\begin{array}{c}
\frac{\forall x. \phi(x)}{\phi(t)} \forall\text{-E} \qquad \frac{\phi(c)}{\forall x. \phi(x)} \forall\text{-I}^* \qquad \frac{\phi(t)}{\exists x. \phi(x)} \exists\text{-I} \qquad \frac{\begin{array}{c} [\phi(c)] \\ \vdots \\ \psi \end{array}}{\exists x. \phi(x)} \exists\text{-E}^* \\
\psi
\end{array}$$

Figure 2.5: Natural deduction rules for quantifiers

The rules  $\forall\text{-I}$  and  $\exists\text{-E}$  have the side condition, that  $c$  must not occur freely in any premise of  $\phi(c)$ .

Sort generation constraints lead to induction proof rules, e.g. for  $\text{Nat}$  (cf. Example in section 2.3.3):

$$[P(0) \wedge \forall x : \text{Nat}. P(x) \rightarrow P(\text{suc}(x))] \rightarrow \forall x : \text{Nat}. P(x),$$

or for  $\text{Lists}$  (cf. Example in section 2.3.4):

$$[P(\text{nil}) \wedge \forall x : \text{Elem}, l : \text{List}. P(l) \rightarrow P(\text{cons}(x, l))] \rightarrow \forall l : \text{List}. P(l),$$

These rules are second order rules (because they quantify over a predicate) but can be instantiated to first order rules for any given (fixed) predicate.

# Chapter 3

## Category theory

### 3.1 Satisfaction Systems

- Notions, results, structuring mechanisms and even tools can be developed independently of the details of a particular logical system.
- Clarifications of the notion of “a logic” or “a logical system”.
- Abstract model- and proof-theory
- Morphisms between logics
- Multi-logic specifications and theories.

**Definition 3.1.1.** A satisfaction system (or information flow classification<sup>1</sup> or twisted relations or two-valued Chu space<sup>2</sup>) consists of

- a set  $S$  (of “sentences”)
- a set  $\mathcal{M}$  (of “Models”)
- a binary relation  $\models \subseteq \mathcal{M} \times S$

**Example 3.1.2.** Given a propositional signature  $\Sigma$ , then  $(\text{Sen}(\Sigma), \text{Mod}(\Sigma), \models_{\Sigma})$  is a satisfaction system. The same holds for any DL, FOL and CFOL (FOL with sort generation constraints).

**Definition 3.1.3.** Given a satisfaction system  $(S, \mathcal{M}, \models)$ , a theory is a set  $\Gamma \subseteq S$  and for a given  $M \in \mathcal{M}$

$$M \models \Gamma \text{ iff } M \models \psi \text{ for all } \psi \in \Gamma.$$

For a given  $\phi \in S$

$$\Gamma \models \phi \text{ iff } \forall M \in \mathcal{M}. M \models \Gamma \rightarrow M \models \phi.$$

$\Gamma$  is satisfiable, if there is some  $M \in \mathcal{M}$  with  $M \models \Gamma$ .

---

<sup>1</sup>Used in information flow theory.

<sup>2</sup>Used in Mathematics.

**Definition 3.1.4.** Given two satisfaction systems  $T_1 = (S_1, \mathcal{M}_1, \models_1)$  and  $T_2 = (S_2, \mathcal{M}_2, \models_2)$  a satisfaction system morphism  $(\alpha, \beta) : T_1 \rightarrow T_2$  is given by

- a map  $\alpha : S_1 \rightarrow S_2$  and
- a map  $\beta : \mathcal{M}_1 \rightarrow \mathcal{M}_2$ ,

such that for all  $\phi_1 \in S_1$  and  $M_2 \in \mathcal{M}_2$

$$\beta(M_2) \models_1 \phi_1 \text{ iff } M_2 \models_2 \alpha(\phi_1) \quad (\text{satisfaction condition})$$

**Example 3.1.5.** For any propositional signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$

$$(\text{Sen}(\Sigma_1), \text{Mod}(\Sigma_1), \models_{\Sigma_1}) \xrightarrow{\alpha, \beta} (\text{Sen}(\Sigma_2), \text{Mod}(\Sigma_2), \models_{\Sigma_2}), \text{ with}$$

- $\alpha(\phi) = \sigma(\phi)$
- $\beta(M) = M|_{\sigma}$

is a satisfaction system morphism, since

$$M_2|_{\sigma} \models \phi_1 \text{ iff } M_2 \models \sigma(\phi_1) \quad (\text{satisfaction condition in Prop}).$$

The same holds for any DL and FOL but not for CFOL.

The satisfaction condition does not hold for CFOL. Consider the following two signatures

1	$\Sigma_1 =$ sorts $s, t$
2	op $c : s$
3	op $f : t \rightarrow t$
4	
5	$\Sigma_2 =$ sort $\text{Nat}$
6	op $0 : \text{Nat}$
7	op $\text{suc} : \text{Nat} \rightarrow \text{Nat}$

Then the signature morphism is uniquely defined as

- $\sigma^S(s) = \sigma^S(t) = \text{Nat}$ ,
- $\sigma_s^F(c) = 0$ ,
- $\sigma_{t,t}^F(f) = \text{suc}$ .

Now consider the sort generation constraint  $\phi_1 = (\{s, t\}, \{c : s, f : t \rightarrow t\})$  and the standard model  $M_2$  for  $\Sigma_2$ ,  $M_2 = (\mathbb{N}, 0, +1)$ . Then  $\sigma(\phi_1) = (\{\text{Nat}\}, \{0 : \text{Nat}, \text{suc} : \text{Nat} \rightarrow \text{Nat}\})$  and

- $(M_2|_{\sigma})_s = (M_2|_{\sigma})_t = \mathbb{N}$ ,
- $c^{M_2|_{\sigma}} = 0$ ,

- $f^{M_2|_\sigma}(x) = x + 1$ .

$M_2 \models \sigma(\phi_1)$  (because the natural numbers are generated by 0 and +1), but  $M_2|_\sigma \not\models \phi_1$ , since the only type correct term is  $c$ , which does not generate the sort  $s$ . Also  $f$  does not generate anything, in particular not  $t$ ).

To save the satisfaction condition we regard sort generation constraints as pairs  $(\sigma : \Sigma' \rightarrow \Sigma, (S_0, F_0))$ , where  $\Sigma' = (S', F', P')$ ,  $S_0 \subseteq S'$ ,  $(F_0)_{w,s} \subseteq F'_{w,s}$ , etc.

Given a  $\Sigma$ -Model  $M$  define

$$M \models (\sigma, (S_0, F_0)) \quad \text{iff} \quad M|_\sigma \models (S_0, F_0) \text{ in the sense defined above}$$

Translation of a  $\Sigma$ -constraint  $(\sigma, (S_0, F_0))$  along  $\theta : \Sigma \rightarrow \bar{\Sigma}$  is defined as

$$\theta((\sigma, (S_0, F_0))) = (\theta \circ \sigma, (S_0, F_0))$$

$$(S_0, F_0) \subset \Sigma' \xleftarrow{\sigma} \Sigma \xleftarrow{\theta} \bar{\Sigma}$$

The satisfaction condition holds since:

$$\begin{aligned} M_2|_\sigma \models (\sigma, (S_0, F_0)) &\text{ iff } (M_2|_\theta)|_\sigma \models (S_0, F_0) \\ &\text{ iff } M_2|_{\theta \circ \sigma} \models (S_0, F_0) \\ &\text{ iff } M_2 \models (\theta \circ \sigma, (S_0, F_0)) \end{aligned}$$

## 3.2 Categories

- Objects are not described by their construction, but by their relation to other objects (“social lives”).
- Provides useful abstraction mechanism.
- Transfer of notions and results from one area to others.
- Visualisation of situations with diagrams.

**Definition 3.2.1.** *A Category  $\mathbf{C}$  consists of*

- a class  $|\mathbf{C}|$  of objects
- for any two objects  $A, B \in |\mathbf{C}|$  a set  $\mathbf{C}(A, B)$  of morphisms from  $A$  to  $B$ .  $f \in \mathbf{C}(A, B)$  is written as  $f : A \rightarrow B$  (not necessarily a function).
- for any object  $A \in |\mathbf{C}|$  an identity morphism

$$id A \in \mathbf{C}(A, A),$$

*i.e.*  $id A : A \rightarrow A$

- for any  $A, B, C \in |\mathbf{C}|$  a composition operation

$$\circ : \mathbf{C}(B, C) \times \mathbf{C}(A, B) \rightarrow \mathbf{C}(A, C),$$

i.e. for  $f : A \rightarrow B, g : B \rightarrow C$ , we have  $g \circ f : A \rightarrow C$ . Alternatively, the notion  $f; g : A \rightarrow C$ .

$$\begin{array}{ccc} & B & \\ f \nearrow & & \searrow g \\ A & \xrightarrow{f; g = g \circ f} & C \end{array}$$

such that

- identities are neutral elements for composition, i.e.  $f \circ id_A = f = id_B \circ f$

$$\begin{array}{ccc} & A & \xrightarrow{f} & B \\ id_A \nearrow & & \searrow f & \\ A & \xrightarrow{f} & B & \nearrow id_B \end{array}$$

- composition is associative, i.e.  $(f \circ g) \circ h = f \circ (g \circ h)$

$$\begin{array}{ccccc} & & & & \\ & \overset{f \circ (g \circ h)}{\curvearrowright} & & & \\ & \overset{g \circ h}{\curvearrowright} & & & \\ A & \xrightarrow{h} & B & \xrightarrow{g} & C & \xrightarrow{f} & D \\ & \underset{(f \circ g) \circ h}{\curvearrowleft} & & & \\ & \underset{f \circ g}{\curvearrowleft} & & & \end{array}$$

### Example 3.2.2.

- **Set**: Sets and functions.
  - $|\mathbf{Set}| = \{M \mid M \text{ is a set}\}$
  - $\mathbf{Set}(A, B) = \{f \mid f : A \rightarrow B \text{ is a function}\} \subseteq \mathcal{P}(A \times B)$
  - Compositions and identities of functions.
- **Sign<sup>Prop</sup>**: Propositional signatures and signature morphisms (= **Set**).
- **Sign<sup>DL</sup>**: DL-signatures and signature morphisms.
- **Sign<sup>FOL</sup>**: FOL-signatures and signature morphisms.
- **Th<sup>Prop</sup>, Th<sup>DL</sup>, Th<sup>FOL</sup>**: Theories and theory morphisms. For DL, we have to choose a particular DL.



- *Theories and mod.-th.cons. theory morphisms. Same objects as  $\mathbf{Th}^*$  but fewer morphisms.*
- **SAT**: *satisfaction systems and morphisms.*
- **Mod<sup>Prop</sup>( $\Sigma$ )**: *Propositional models, where a unique morphism  $f : M \rightarrow M'$  exists iff*

$$M \leq M', \text{ i.e. } \forall p \in \Sigma. M(p) = \top \Rightarrow M'(p) = \top.$$

- *Similarly any pre-order (i.e. reflexive, transitive relation) is a category.*
- *Sentences and logical entailment  $\phi \models \psi$  is a pre-order, hence a category.*
- *Pre-orders and monotone maps*
- **Graph**: *Graphs and homomorphisms*
- *Moore automata and simulations*
- *Petri nets and simulations*
- *Parallel systems and superpositions*
- *Monoids and homomorphisms*
- *Groups and homomorphisms*
- *Rings and homomorphisms*
- *vector spaces and linear maps*
- *metric spaces and non-expansive maps*
- *topological spaces and continuous maps*

**Definition 3.2.3.**  $f : A \rightarrow B$  is an isomorphism if there is a morphism  $g : B \rightarrow A$  such that  $f \circ g = id_B$  and  $g \circ f = id_A$

$$\begin{array}{ccc}
 & A & \\
 g \nearrow & & \searrow f \\
 B & \xrightarrow{id_B} & B
 \end{array}
 \quad
 \begin{array}{ccc}
 & B & \\
 f \nearrow & & \searrow g \\
 A & \xrightarrow{id_A} & A
 \end{array}$$

**Definition 3.2.4.** An object  $0 \in |\mathbf{C}|$  is called initial if for any  $A \in |\mathbf{C}|$  there exists a unique morphism  $0_A = 0 \rightarrow A$ .

**Example 3.2.5.**

- **Set**:  $\emptyset$ .

- **Graph**: the empty graph.
- $\text{Mod}^{Prop}(\Sigma)$ : the everywhere false model.
- $\text{Mod}^{Prop}(\Sigma, \Gamma)$ : the model of  $\text{free}\{\Sigma, \Gamma\}$

**Definition 3.2.6.** Analogously, an object  $1 \in |\mathbf{C}|$  is called terminal, if for each  $A \in |\mathbf{C}|$  there exists a unique morphism  $1_A = A \rightarrow 1$ .

**Example 3.2.7.**

- **Set**: any singleton set.
- $\text{Mod}^{Prop}(\Sigma, \Gamma)$ : the model of  $\text{cofree}\{\Sigma, \Gamma\}$

**Definition 3.2.8.** Given a Category  $\mathbf{C}$  its dual  $\mathbf{C}^{op}$  is given by just inverting all arrows. Formally:

$$\begin{aligned} |\mathbf{C}^{op}| &= |\mathbf{C}| \\ \mathbf{C}^{op}(A, B) &= \mathbf{C}(A, B) \\ f \circ^{\mathbf{C}^{op}} g &= g \circ^{\mathbf{C}} f \\ (id_A)^{\mathbf{C}^{op}} &= (id_A)^{\mathbf{C}} \end{aligned}$$

**Proposition 3.2.9.**

$$(\mathbf{C}^{op})^{op} = \mathbf{C}$$

**Proposition 3.2.10.** An object  $A$  is initial in  $\mathbf{C}$  iff it is terminal in  $\mathbf{C}^{op}$

*Proof.* There exists a unique morphism  $A \rightarrow B$  in  $\mathbf{C}$ , iff there is a unique morphism  $B \rightarrow A$  in  $\mathbf{C}^{op}$   $\square$

That is: the notion “terminal object” is the dual of the notion “initial object”.

**Theorem 3.2.11.** *Meta-Theorem: If a theorem holds for all categories, it also holds for all categories, when all notions are replaced by their duals and all arrows are reversed.*

**Definition 3.2.12** (Products). Given two categories  $\mathbf{C}_1$  and  $\mathbf{C}_2$  their product  $\mathbf{C}_1 \times \mathbf{C}_2$  is defined as follows:

$$\begin{aligned} |\mathbf{C}_1 \times \mathbf{C}_2| &= |\mathbf{C}_1| \times |\mathbf{C}_2| \\ \mathbf{C}_1 \times \mathbf{C}_2((A_1, A_2), (B_1, B_2)) &= \mathbf{C}_1(A_1, B_1) \times \mathbf{C}_2(A_2, B_2) \\ (f_1, f_2) \circ (g_1, g_2) &= (f_1 \circ g_1, f_2 \circ g_2) \\ id_{(A_1, A_2)} &= (id_{A_1}, id_{A_2}) \end{aligned}$$

**Exercise 26** (Generalization of surjectiveness)

Let  $A$  and  $B$  be two arbitrary sets and  $f : A \rightarrow B$  a function. Show that the following statements are equivalent:

- (a)  $f$  is *surjective*, i.e., for all  $b \in B$  there exists  $a \in A$  such that  $b = f(a)$ .
- (b)  $f$  is an *epimorphism*, i.e., for all sets  $C$  and functions  $g, g' : B \rightarrow C$ , if  $g \circ f = g' \circ f$ , then  $g = g'$ .
- (c)  $f$  is a *retraction*, i.e., there is a function  $g : B \rightarrow A$  such that  $f \circ g = id_B$ .

Here,  $\circ$  is the function composition, and  $id_B$  is the identity function on  $B$ .

**Exercise 27** (Uniqueness of identities)

Show that in any category, identities are unique.

**Exercise 28** (Initial and terminal objects)

What are the initial and terminal objects in the following categories?

- (a) The category of propositional signatures and signature morphisms
- (b) The category of propositional theories and theory morphisms
- (c) The category of propositional theories and conservative theory morphisms
- (d) The category of propositional models for the signature  $\{p, q\}$
- (e) The category of satisfaction systems

**Exercise 29** (Uniqueness of initial objects)

Prove that any two initial objects are isomorphic and that any two terminal objects are isomorphic.

**Exercise 30** (Isomorphic theories)

What does it mean that two theories are isomorphic?

**Exercise 31** (Graphic depiction of categories)

There are different ways to realise a category with two objects and four arrows. Draw all the possible shapes that such a category can have (in the drawing, please use a different style for the identity arrows, or even omit them). Specify at least two different composition laws (if existing) for each shape.

### 3.2.1 Functors

Functors are morphisms between categories.

**Definition 3.2.13.** *Given two categories  $\mathbf{C}_1$  and  $\mathbf{C}_2$ , a functor  $F : \mathbf{C}_1 \rightarrow \mathbf{C}_2$  is defined by:*

- *its action on objects:*

$$F : |\mathbf{C}_1| \rightarrow |\mathbf{C}_2|$$

- its action on morphisms

$$F : \mathbf{C}_1(A, B) \rightarrow \mathbf{C}_2(FA, FB)$$

i.e. for  $f : A \rightarrow B \in \mathbf{C}_1$ ,  $Ff : FA \rightarrow FB \in \mathbf{C}_2$

such that

- $Fid_A = id_{FA}$  (preservation of identities)
- for  $f : x \rightarrow y, g : y \rightarrow z$ ,  $F(f \circ g) = Ff \circ Fg$  (preservation of compositions)

**Example 3.2.14.**

- $I : \mathbf{Sign}^{Prop} \rightarrow \mathbf{Sign}^{FOL}$   
 Objects:  $\Sigma \mapsto (\emptyset, \emptyset, P)$  with  $P_\lambda = \Sigma$   
 Morphisms:  $\Sigma \xrightarrow{\sigma'} \Sigma'$  with  $\sigma' \mapsto (\emptyset, \emptyset, P)$  with  $P_\lambda = \Sigma'$
- $\Pi_1 : \mathbf{C}_1 \times \mathbf{C}_2 \rightarrow \mathbf{C}_1$   
 $(A_1, A_2) \mapsto (A_1)$   
 $\downarrow (f_1, f_2) \mapsto \downarrow (f_1)$   
 $(B_1, B_2) \mapsto (B_1)$
- Analogously for  $\Pi_1 : \mathbf{C}_1 \times \mathbf{C}_2 \rightarrow \mathbf{C}_2$
- Given  $A_2 \in |\mathbf{C}_2|$ ,  $\_ \times A_2 : \mathbf{C}_1 \times \mathbf{C}_2$   
 $A_1 \mapsto A_1 \times A_2$   
 $\downarrow f_1 \mapsto \downarrow (f_1, id_{A_2})$   
 $B_1 \mapsto B_1 \times A_2$

**Exercise 32 (Functors)**

Given a category  $\mathbf{C}$ , what are the functors

(a) from  $\mathbf{1} = \begin{array}{c} \text{id}_A \\ \curvearrowright \\ A \end{array}$  to  $\mathbf{C}$ ?

(b) from  $\mathbf{2} = \begin{array}{ccc} \text{id}_A & & \text{id}_B \\ \curvearrowright & \xrightarrow{f} & \curvearrowright \\ A & & B \end{array}$  to  $\mathbf{C}$ ?

(c) from  $\mathbf{3} = \begin{array}{ccc} & \text{id}_B & \\ & \curvearrowright & \\ & B & \\ \text{id}_A \curvearrowright & \begin{array}{c} \swarrow f \\ \searrow g \end{array} & \curvearrowright \text{id}_C \\ A & \xrightarrow{h} & C \end{array}$  to  $\mathbf{C}$ ?

### 3.2.2 Institutions

**Definition 3.2.15.** A functor  $\models: \mathbf{C}_1 \rightarrow \mathbf{C}_2$  is faithful, if for  $f, g : A \rightarrow B \in \mathbf{C}_1$ ,  $Ff = Fg$  implies  $f = g$

**Definition 3.2.16.** A concrete category is a faithful functor  $U : \mathbf{C} \rightarrow \mathbf{Set}$ . Intuitively, this means that  $\mathbf{C}$  is a category of sets with structure and structure preserving maps.

**Definition 3.2.17.** A category  $\mathbf{C}$  is concretisable, if there is a faithful functor  $U : \mathbf{C} \rightarrow \mathbf{Set}$

**Example 3.2.18.**

- **PreOrd:** Pre-Orders and monotone maps
- **Mon:** Monoids and homomorphisms

*Recall:* **SAT** is the (quasi)<sup>1</sup>-category of all satisfaction systems and satisfaction system morphisms  $(S, M, \models)$

- $S$  a set
- $M$  a class
- $\models \subseteq M \times S$

**Definition 3.2.19.** An institution is a functor  $\mathbf{Sign} \xrightarrow{I} \mathbf{Sat}$  where  $\mathbf{Sign}$  is any category of signatures. Explicitly this means, that an institution is given by

- a category  $\mathbf{Sign}$  of signatures
- for each signature  $\Sigma \in |\mathbf{Sign}|$  a satisfaction system  $I(\Sigma) = (Sen(\Sigma), Mod(\Sigma), \models_\Sigma)$ 
  - $Sen(\Sigma)$  is the set of sentences over  $\Sigma$
  - $Mod(\Sigma)$  is the set of models over  $\Sigma$
  - $\models_\Sigma$  is the satisfaction relation for  $\Sigma$
- for each signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  a satisfaction system morphism  $I(\sigma) = (\sigma, \_ |_\sigma) : I(\Sigma_1) \rightarrow I(\Sigma_2)$ 
  - $\sigma : Sen(\Sigma_1) \rightarrow Sen(\Sigma_2)$  is called sentence translation
  - $\_ |_\sigma : Mod(\Sigma_2) \rightarrow Mod(\Sigma_1)$  is called model reduction.

---

<sup>1</sup>there is no class of all classes, hence we need the notion of conglomerates

such that for each  $\phi_1 \in \text{Sen}(\Sigma_1), M_2 \in \text{Mod}(\Sigma_2)$ ,  $M_2|_{\sigma} \models_{\Sigma_1} \phi_1$  iff  $M_2 \models_{\Sigma_2} \sigma(\phi_1)$  (satisfaction condition)<sup>2</sup> and such that

$$\begin{aligned} \text{id}(\phi) &= \phi \\ M|_{\text{id}} &= M \\ \sigma_1 \circ \sigma_2(\phi) &= \sigma_1(\sigma_2(\phi)) \\ M|_{\sigma_1 \circ \sigma_2} &= (M|_{\sigma_1})|_{\sigma_2} \end{aligned}$$

### Alternative definition of institutions

**Definition 3.2.20.** A category  $\mathbf{C}$  is small, if  $|\mathbf{C}|$  is a set. Let  $\mathbf{Cat}$  be the category with

- $|\mathbf{Cat}| =$  all small categories
- $\mathbf{Cat}(\mathbf{C}, \mathbf{D}) =$  functors from  $\mathbf{C}$  to  $\mathbf{D}$
- obvious identities and compositions.

**Definition 3.2.21.**  $\mathbf{CAT}$  is the quasi-category of all categories and functors.

**Definition 3.2.22.** An institution  $(\mathbf{Sign}, \text{Sen}, \text{Mod}, \models)$  consists of

- a category  $\mathbf{Sign}$  of signatures.
- a functor  $\text{Sen} : \mathbf{Sign} \rightarrow \mathbf{Set}$  mapping each  $\Sigma \in |\mathbf{Sign}|$  to the set  $\text{Sen}(\Sigma)$  of sentences over  $\Sigma$  and each  $\sigma : \Sigma_1 \rightarrow \Sigma_2 \in \mathbf{Sign}$  to the sentence translation  $\text{Sen}(\Sigma_1) \rightarrow \text{Sen}(\Sigma_2)$  usually also denoted by  $\sigma$ .
- for each functor  $\text{Mod} : \mathbf{Sign}^{\text{op}} \rightarrow \mathbf{CAT}$  mapping each  $\Sigma \in |\mathbf{Sign}|$  to the model category  $\text{Mod}(\Sigma)$  and each signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2 \in \mathbf{Sign}$  (i.e.  $\sigma : \Sigma_2 \rightarrow \Sigma_1 \in \mathbf{Sign}^{\text{op}}$ ) to the reduct functor  $\text{Mod}(\sigma) : \text{Mod}(\Sigma_2) \rightarrow \text{Mod}(\Sigma_1)$  usually written as  $_ |_{\sigma}$ .
- for each  $\Sigma \in |\mathbf{Sign}|$ , a satisfaction relation  $\models_{\Sigma} \subseteq |\text{Mod}(\Sigma)| \times \text{Sen}(\Sigma)$ , such that for  $\sigma : \Sigma_1 \rightarrow \Sigma_2 \in \mathbf{Sign}$ ,  $\phi_1 \in \text{Sen}(\Sigma_1)$ ,  $M_2 \in \text{Mod}(\Sigma_2)$ ,

$$M_2|_{\sigma} \models_{\Sigma_1} \phi_1 \text{ iff } M_2 \models_{\Sigma_2} \sigma(\phi_1)$$

**Definition 3.2.23.** A room consists of

- a set  $S$
- a category  $\mathbf{M}$
- a relation  $|\mathbf{M}| \times S$

---

<sup>2</sup>“truth is invariant under change of notation”

**Definition 3.2.24.** Given two rooms  $(S_1, M_1, \models_1)$  and  $(S_2, M_2, \models_2)$  a corridor  $(\alpha, \beta) : (S_1, M_1, \models_1) \rightarrow (S_2, M_2, \models_2)$ , where

- $\alpha : S_1 \rightarrow S_2$  is a function
- $\beta : M_2 \rightarrow M_1$  is a function,

such that for  $\phi_1 \in S_1, M_2 \in |M_2|$

$$\beta(M_2) \models_1 \phi_1 \text{ iff } M_2 \models_2 \alpha(\phi_1)$$

**Proposition 3.2.25.** An institution as defined above (alternative definition) is the same as a functor  $I : \mathbf{Sign} \rightarrow \mathbf{Room}$

*Proof.*

$$\mathbf{Sign} \xrightarrow{I} \mathbf{Sat}$$

- $I(\Sigma) = (\text{Sen}(\Sigma), \text{Mod}(\Sigma), \models_\Sigma)$
- $I(\sigma) = (\text{Sen}(\sigma), \text{Mod}(\sigma)) = (\sigma, \_ |_\sigma)$

□

Convention: from now on let  $I : \mathbf{Sign} \rightarrow \mathbf{Sat}$  be an arbitrary but fixed institution:

**Definition 3.2.26.** Given a signature  $\Sigma \in |\mathbf{Sign}|, \Gamma \subseteq \text{Sen}(\Sigma), \phi \in \text{Sen}(\Sigma)$   $\phi$  is a logical consequence of  $\Gamma$  (written as  $\Gamma \models_\Sigma \phi$ ), iff for all  $M \in \text{Mod}(\Sigma)$

$$M \models \Gamma \text{ implies } M \models_\Sigma \phi$$

**Definition 3.2.27.** A theory is a pair  $(\Sigma, \Gamma)$ , where  $\Sigma \in |\text{Sig}|, \Gamma \subseteq \text{Sen}(\Sigma)$ .

**Definition 3.2.28.** A theory morphism  $\sigma : (\Sigma_1, \Gamma_1) \rightarrow (\Sigma_2, \Gamma_2)$  is a signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  such that for all  $M_2 \in \text{Mod}(\Sigma_2)$   $M_2 \models \Gamma_2$  implies  $M_2 |_\sigma \models \Gamma_1$

**Proposition 3.2.29.**  $\sigma : (\Sigma_1, \Gamma_1) \rightarrow (\Sigma_2, \Gamma_2)$  iff  $\Gamma_2 \models \sigma(\Gamma_1)$

*Proof.*

$$\begin{aligned} & \Gamma_2 \models \sigma(\Gamma_1) \\ \text{iff} & \text{ for all } M_2 \in \text{Mod}(\Sigma_2), M_2 \models \Gamma_2 \Rightarrow M_2 \models \sigma(\Gamma_1) \\ \text{iff} & \text{ by satisfaction condition} \\ & \text{for all } M_2 \in \text{Mod}(\Sigma_2), M_2 \models \Gamma_2 \Rightarrow M_2 |_\sigma \models \Gamma_1 \\ \text{iff} & \sigma : (\Sigma_1, \Gamma_1) \rightarrow (\Sigma_2, \Gamma_2) \text{ is a theory morphism.} \end{aligned}$$

□

**Exercise 33** (Institution of theories)

Given an institution  $I$ , obtain  $I^{\text{th}}$  by replacing the category of signatures with the category of theories in  $I$ . Show that this yields an institution.

### 3.2.3 Structured specifications

- Manageability
- re-use
- clarity

**Definition 3.2.30.** We inductively define the set of structured specifications in  $I$ :

- $\langle \Sigma, \Gamma \rangle$  with  $\Sigma \in \mathbf{Sign}, \Gamma \subseteq \text{Sen}(\Sigma)$ 
  - $\text{Sig}(\langle \Sigma, \Gamma \rangle) = \Sigma$
  - $\text{Mod}(\langle \Sigma, \Gamma \rangle) = \{M \in \text{Mod}(\Sigma) \mid M \models \Gamma\}$
- Given two specifications  $SP_1$  and  $SP_2$  with  $\text{Sig}(SP_1) = \text{Sig}(SP_2) = \Sigma$  “ $SP_1$  and  $SP_2$ ” is a specification with
  - $\text{Sig}(SP_1 \text{ and } SP_2) = \Sigma$
  - $\text{Mod}(SP_1 \text{ and } SP_2) = \text{Mod}(SP_1) \cap \text{Mod}(SP_2)$
- Given a specification  $SP$  with  $\text{Sig}(SP) = \Sigma$  and a signature morphism  $\sigma : \Sigma \rightarrow \Sigma'$  “ $SP$  with  $\sigma$ ” is a specification with
  - $\text{Sig}(SP \text{ with } \sigma) = \Sigma'$
  - $\text{Mod}(SP \text{ with } \sigma) = \{M \in \text{Mod}(\Sigma') \mid M|_{\sigma} \in \text{Mod}(SP)\}$
- Given a specification  $SP$  with  $\text{Sig}(SP) = \Sigma'$  and a signature morphism  $\sigma : \Sigma \rightarrow \Sigma'$  “ $SP$  hide  $\sigma$ ” is a specification with
  - $\text{Sig}(SP \text{ hide } \sigma) = \Sigma$
  - $\text{Mod}(SP \text{ hide } \sigma) = \{M' \mid_{\sigma} \mid M' \in \text{Mod}(SP)\}$

**Definition 3.2.31.** A signature fragment  $\Delta$  is a signature morphism  $\Delta : \Sigma \rightarrow \Sigma'$ . Intuitively “ $\Sigma' \setminus \Sigma$ ”

**Definition 3.2.32** (structured specifications). If  $SP$  is a specification with  $\text{Sig}(SP) = \Sigma$ ,  $\Delta : \Sigma \rightarrow \Sigma'$  a signature fragment and  $\Gamma' \subseteq \text{Sen}(\Sigma')$ , “ $SP$  then  $(\Delta, \Gamma)$ ” is a specification with the same semantics as  $(SP \text{ with } \Delta)$  and  $\langle \Sigma', \Gamma' \rangle$

If we have a union operation on signatures  $\cup : |\mathbf{Sign}| \times |\mathbf{Sign}| \rightarrow |\mathbf{Sign}|$  such that there exists signature morphisms for  $\Sigma_1$  and  $\Sigma_2$  such that  $\Sigma_1 \xrightarrow{\iota_1} \Sigma_1 \cup \Sigma_2 \xleftarrow{\iota_2} \Sigma_2$  We can form unions of specifications with different signatures: Given specifications  $SP_1$  and  $SP_2$  with  $\text{Sig}(SP_i) = \Sigma_i$  “ $SP_1$  and  $SP_2$ ” is a specification with semantics as

$$(SP_1 \text{ with } \iota_1) \text{ and } (SP_2 \text{ with } \iota_2)$$



**Definition 3.2.33.** Let  $SP$  be a specification with  $Sig(SP) = \Sigma$  and  $\phi \in Sen(\Sigma)$ , then  $SP \models \phi$  iff for each  $M \in Mod(SP)$ ,  $M \models \phi$ .

**Example 3.2.34** (in  $\mathcal{ALC}$ ):

$\Sigma = \text{Classes: Person, Female, Woman, Man}$ $\Gamma = \{\text{Woman} \equiv \text{Person} \sqcap \text{Female}, \text{Man} \equiv \text{Person} \sqcap \neg \text{Female}\}$
---

Consider  $\langle \Sigma, \Gamma \rangle_{\text{hide Person, Female}}$  (shorthand for  $\langle \Sigma, \Gamma \rangle_{\text{hide } \sigma}$  with the signature inclusion  $\sigma : (\text{Classes: Woman, Man}) \hookrightarrow \Sigma$ )

$\langle \Sigma, \Gamma \rangle_{\text{hide Person, Female}} \models \text{Man} \sqcap \text{Woman} \equiv \perp$

$Mod(SP \text{ hide } \sigma) = \{M|_{\sigma} \mid M \in Mod(SP)\}$

**Definition 3.2.35.** Let  $SP_1, SP_2$  be specifications. A signature morphism  $\sigma : Sig(SP_1) \rightarrow Sig(SP_2)$  is a specification morphism, if for all  $M_2 \in Mod(SP_2)$ :

$$M_2|_{\sigma} \in Mod(SP_1)$$

$\sigma$  is model theoretically conservative, if each  $M_1 \in Mod(SP_1)$  has a  $\sigma$ -expansion to a  $SP_2$ -model.  $\sigma$  is consequence theoretically conservative, if for all  $\phi_1 \in Sen(Sig(SP_1))$ ,  $SP_2 \models \sigma(\phi_1)$  implies  $SP_1 \models \phi_1$ .

**Proposition 3.2.36.** Any model theoretically conservative specification morphism is also consequence theoretically conservative.

*Proof.* Let  $\sigma : SP_1 \rightarrow SP_2$  be a mod.-th.-cons. specification morphism. Let  $SP_2 \models \sigma(\phi_1)$  and  $M_1 \in Mod(SP_1)$ . We have to show  $M_1 \models \phi_1$ .

Since  $\sigma$  is mod.-th.-cons. there is  $M_2 \in Mod(SP_2)$  with  $M_2|_{\sigma} = M_1$ . By  $SP_2 \models \sigma(\phi_1)$  we have  $M_2 \models \sigma(\phi_1)$ . By the satisfaction condition  $M_2|_{\sigma} \models \phi_1$  thus  $M_1 \models \phi_1$ .  $\square$

**Exercise 34** (Structured specifications)

- (a) Show that, given a structured specification  $SP$  and a signature morphism  $\sigma : \Sigma \rightarrow Sig(SP)$ ,
  - (i)  $\sigma$  is a specification morphism  $\sigma : SP \text{ hide } \sigma \rightarrow SP$
  - (ii) the specification morphism is model-theoretically conservative
- (b) Structure some of the specifications that you have written so far and/or that have been used in the lecture using the new structuring mechanisms.

**Normal forms of specifications**

**Definition 3.2.37.** The normal form of a specification is inductively defined as follows

- $NF(\langle \Sigma, \Gamma \rangle) := \langle \Sigma, \Gamma \rangle$

- Let  $SP = \text{“}SP_1 \text{ and } SP_2\text{”}$  with  $NF(SP_1) = \langle \Sigma, \Gamma_1 \rangle$  and  $NF(SP_2) = \langle \Sigma, \Gamma_2 \rangle$ . Then  $NF(SP) := \langle \Sigma, \Gamma_1 \cup \Gamma_2 \rangle$
- Let  $SP = \text{“}SP_1 \text{ with } \sigma\text{”}$  with  $\sigma : \Sigma_1 \rightarrow \Sigma$  and  $NF(SP_1) = \langle \Sigma_1, \Gamma_1 \rangle$ . Then  $NF(SP) := \langle \Sigma, \sigma(\Gamma_1) \rangle$
- $NF(SP \text{ hide } \sigma)$  is undefined.

**Example 3.2.38.** Consider the following specification  $SP$

```

1 sort s
2 op  f: s → s
3   ∀ x, y : s
4     . ∃ z : s. ¬∃ u : s. f(u) = z
5     . f(x) = f(y) => x = y
6 hide f

```

We have

$$\begin{aligned}
SP &\models \exists x, y : s. x \neq y \\
SP &\models \exists x, y, z : s. (x \neq y \wedge x \neq z \wedge y \neq z) \\
&\dots
\end{aligned}$$

$SP$  is not finitely axiomatisable.

**Proposition 3.2.39.** Let  $SP$  be a specification without `hide` and let  $NF(SP) = \langle \Sigma, \Gamma \rangle$ . Then

- $Sig(SP) = \Sigma$
- $Mod(SP) = Mod(\langle \Sigma, \Gamma \rangle)$

*Proof.* Induction over  $SP$

- $SP = \langle \Sigma, \Gamma \rangle$ : clear
- $SP = \text{“}SP_1 \text{ and } SP_2\text{”}$ :

$$\begin{aligned}
Mod(SP_1 \text{ and } SP_2) &=^{\text{def}} Mod(SP_1) \cap Mod(SP_2) \\
&=^{\text{I.H.}} Mod(\langle \Sigma, \Gamma_1 \rangle) \cap Mod(\langle \Sigma, \Gamma_2 \rangle) \\
&= Mod(\langle \Sigma, \Gamma_1 \cup \Gamma_2 \rangle) = Mod(NF(SP))
\end{aligned}$$

- $SP = \text{“}SP_1 \text{ with } \sigma\text{”}$ :

$$\begin{aligned}
Mod(SP_1 \text{ with } \sigma) &=^{\text{def}} \{M \in Mod(\Sigma) \mid M|_{\sigma} \in Mod(SP_1)\} \\
&=^{\text{I.H.}} \{M \in Mod(\Sigma) \mid M|_{\sigma} \in Mod(\langle \Sigma_1, \Gamma_1 \rangle)\} \\
&=^{\text{def}} \{M \in Mod(\Sigma) \mid M|_{\sigma} \models \Gamma_1\} \\
&=^{\text{sat.cond.}} \{M \in Mod(\Sigma) \mid M \models \sigma(\Gamma_1)\} \\
&= Mod(\langle \Sigma, \sigma(\Gamma_1) \rangle) = Mod(NF(SP))
\end{aligned}$$

□

**Exercise 35** (Specification morphisms)

Let  $SP_1, SP_2$  be two specifications. Show that for any signature morphism  $\sigma : \text{Sig}(SP_1) \rightarrow \text{Sig}(SP_2)$ , the following are equivalent:

- (a)  $\sigma : SP_1 \rightarrow SP_2$  is a specification morphism
- (b)  $\text{Mod}(SP_2 \text{ hide } \sigma) \subseteq \text{Mod}(SP_1)$
- (c)  $\text{Mod}(SP_2) \subseteq \text{Mod}(SP_1 \text{ with } \sigma)$

**Exercise 36** (Models of specifications)

Show that the following statements are not equivalent. Provide counterexamples for both implications.

- (a)  $\text{Mod}(SP_1) \subseteq \text{Mod}(SP_2 \text{ hide } \sigma)$
- (b)  $\text{Mod}(SP_1 \text{ with } \sigma) \subseteq \text{Mod}(SP_2)$

**Exercise 37** (Algebraic laws for specifications)

Check which of the following algebraic laws hold:

- (a)  $SP$  and  $SP \equiv SP$
- (b)  $SP_1$  and  $SP_2 \equiv SP_2$  and  $SP_1$
- (c)  $(SP \text{ with } \sigma_1) \text{ with } \sigma_2 \equiv SP \text{ with } \sigma_2 \circ \sigma_1$
- (d)  $(SP_1 \text{ and } SP_2) \text{ with } \sigma \equiv (SP_1 \text{ with } \sigma) \text{ and } (SP_2 \text{ with } \sigma)$
- (e)  $(SP \text{ hide } \sigma_2) \text{ hide } \sigma_1 \equiv SP \text{ hide } \sigma_2 \circ \sigma_1$
- (f)  $(SP_1 \text{ and } SP_2) \text{ hide } \sigma \equiv (SP_1 \text{ hide } \sigma) \text{ and } (SP_2 \text{ hide } \sigma)$
- (g)  $(SP \text{ with } \sigma) \text{ hide } \sigma \equiv SP$
- (h)  $(SP \text{ hide } \sigma) \text{ with } \sigma \equiv SP$

### 3.2.4 Institutions with proofs

**Definition 3.2.40.** Given an institution  $\mathbf{Sign} \xrightarrow{I} \mathbf{Room}$

(or  $(\mathbf{Sign}, \text{Sen}, \text{Mod}, \models)$ ) an entailment system for  $I$  is given by relations

$$\vdash_{\Sigma} \subseteq \mathcal{P}(\text{Sen}(\Sigma)) \times \text{Sen}(\Sigma)$$

for each  $\Sigma \in |\mathbf{Sign}|$ , such that

- $\{\phi\} \vdash_{\Sigma} \phi$  for  $\phi \in \text{Sen}(\Sigma)$  (reflexivity)
- for all  $i \in J$ ,  $\Gamma_i \vdash_{\Sigma} \phi_i$  and  $\{\phi_i | i \in J\} \vdash_{\Sigma} \psi$  implies  $\bigcup_{i \in J} \Gamma_i \vdash_{\Sigma} \psi$  (transitivity)
- $\Gamma \vdash_{\Sigma} \phi$  and  $\Gamma \subseteq \Gamma'$  implies  $\Gamma' \vdash_{\Sigma} \phi$  with  $\Gamma, \Gamma', \{\phi\} \subseteq \text{Sen}(\Sigma)$  (monotonicity)
- if  $\sigma : \Sigma_1 \rightarrow \Sigma_2 \in \mathbf{Sign}$  and  $\Gamma \vdash_{\Sigma_1} \phi$  with  $\Gamma, \{\phi\} \subseteq \text{Sen}(\Sigma_1)$  then  $\sigma(\Gamma) \vdash_{\Sigma_2} \sigma(\phi)$  (translation)

**Definition 3.2.41.** An entailment system for  $I$  is

- sound, if  $\Gamma \vdash_{\Sigma} \phi$  implies  $\Gamma \models_{\Sigma} \phi$
- strongly complete, if  $\Gamma \models_{\Sigma} \phi$  implies  $\Gamma \vdash_{\Sigma} \phi$
- weakly complete, if  $\emptyset \models_{\Sigma} \phi$  implies  $\emptyset \vdash_{\Sigma} \phi$

**Example 3.2.42.** Prop and FOL have sound and complete entailment systems

### Proof system for structured specifications

Let an institution with a sound entailment system be given.

$$\begin{array}{c}
\frac{(SP \vdash_{\Sigma} \phi_i)_{i \in J} \quad \{\phi_i | i \in J\} \vdash_{\Sigma} \psi}{SP \vdash_{\Sigma} \phi} \text{ (CR)} \quad \frac{\phi \in \Gamma}{\langle \Sigma, \Gamma \rangle \vdash_{\Sigma} \phi} \text{ (basic)} \\
\\
\frac{SP_1 \vdash_{\Sigma} \phi}{SP_1 \text{ and } SP_2 \vdash_{\Sigma} \phi} \text{ (and-left)} \quad \frac{SP_2 \vdash_{\Sigma} \phi}{SP_1 \text{ and } SP_2 \vdash_{\Sigma} \phi} \text{ (and-right)} \\
\\
\frac{SP \vdash_{\Sigma} \phi}{SP \text{ with } \sigma \vdash_{\Sigma'} \sigma(\phi)} \text{ (translation)} \quad \frac{SP \vdash_{\Sigma'} \sigma(\phi)}{SP \text{ hide } \sigma \vdash_{\Sigma} \phi} \text{ (hiding)}
\end{array}$$

Figure 3.1: Natural deduction rules for structured specifications

**Proposition 3.2.43.** Proving in structured specifications is sound, i.e.

$$SP \vdash_{\Sigma} \phi \text{ implies } SP \models_{\Sigma} \phi$$

*Proof.* Induction over the derivation of  $SP \vdash_{\Sigma} \phi$ .

- (CR): By I.H.  $SP \models_{\Sigma} \phi_i$ . By soundness of the entailment system  $\{\phi_i | i \in J\} \models_{\Sigma} \psi$ . Hence  $SP \models_{\Sigma} \psi$ .
- (basic): clear
- (and-left): By I.H.  $SP_1 \models_{\Sigma} \phi$ . Any  $(SP_1 \text{ and } SP_2)$  model is a  $SP_1$  model, hence  $(SP_1 \text{ and } SP_2) \models_{\Sigma} \phi$
- (and-right): analogous
- (translation): By I.H.  $SP \models_{\Sigma} \phi$ . For  $M \in \text{Mod}(SP \text{ with } \sigma)$  we have  $M|_{\sigma} \in \text{Mod}(SP)$ . Hence  $M|_{\sigma} \models_{\Sigma} \phi$ . By satisfaction condition  $M \models_{\Sigma'} \sigma(\phi)$
- (hiding): By I.H.  $SP \models_{\Sigma} \sigma(\phi)$ . Let  $M \in \text{Mod}(SP \text{ hide } \sigma)$ . Then there is  $M' \in \text{Mod}(SP)$  with  $M'|_{\sigma} = M$ .  $M' \models_{\Sigma'} \sigma(\phi)$  and by satisfaction condition  $M'|_{\sigma} = M \models_{\Sigma} \phi$

□

### 3.3 Colimits

“Given a species of structure, say widgets, the result of interconnecting a system of widgets to a super-widget corresponds to taking the *colimit* of the diagram of widgets in which the morphisms show how they are interconnected” (J. Goguen)

Colimits are also used in Ontology alignment

#### 3.3.1 Coproducts

Combination without interaction:

**Definition 3.3.1.** Given two objects  $A, B \in |\mathbf{C}|$  a coproduct of  $A$  and  $B$  is an object  $A \amalg B$  with two morphisms  $A \xrightarrow{\Pi_A} A \amalg B \xleftarrow{\Pi_B} B$  (called the coproduct injections) such that for any  $A \xrightarrow{f} C \xleftarrow{g} B$  there is a unique  $h : A \amalg B \rightarrow C$  such that

$$\begin{array}{ccc}
 & A \amalg B & \\
 \Pi_A \nearrow & \vdots & \nwarrow \Pi_B \\
 A & h & B \\
 f \searrow & & \nearrow g \\
 & C & 
 \end{array}$$

*commutes.*

**Example 3.3.2.** In **Set**, coproducts are disjoint unions

$$\begin{aligned}
 A \cup B &= A \times \{0\} \cup B \times \{1\} \\
 A &\xrightarrow{\Pi_A} A \cup B \\
 a &\mapsto (a, 0) \\
 B &\xrightarrow{\Pi_B} A \cup B \\
 b &\mapsto (b, 1)
 \end{aligned}$$

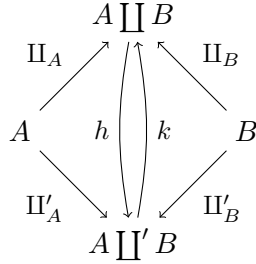
Given two functions  $A \xrightarrow{f} C \xleftarrow{g} B$  define  $h : A \cup B \rightarrow C$  as

$$\begin{aligned}
 h(a, 0) &= f(a) \\
 h(b, 1) &= g(b) \\
 h(\Pi_A(a)) &= f(a) \\
 h(\Pi_B(b)) &= g(b)
 \end{aligned}$$

Now let  $k : A \cup B \rightarrow C$  with  $k \circ \Pi_A = f$  and  $k \circ \Pi_B = g$  But then  $k(a, 0) = f(a)$  and  $k(b, 1) = g(b)$  Hence  $h = k$

**Proposition 3.3.3.** *Coproducts are unique up to isomorphism*

*Proof.* Given  $A, B \in |\mathbf{C}|$ , and let  $A \xrightarrow{\Pi_A} A \amalg B \xleftarrow{\Pi_B} B$  and  $A \xrightarrow{\Pi'_A} A \amalg' B \xleftarrow{\Pi'_B} B$  be coproducts. By the universal property of  $A \amalg B$  we have  $k : A \amalg' B \rightarrow C$  such that the following commutes



In order to show that  $h$  is in fact an isomorphism, we need to show:  $h \circ k = id$  and  $k \circ h = id$  □

**Exercise 38** (Coproducts I)

Show that

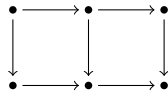
- (a)  $[h \circ f, h \circ g] = h \circ [f, g]$
- (b)  $(f \amalg g) \circ (k \amalg h) = (f \circ k) \amalg (g \circ h)$

**Exercise 39** (Coproducts II)

- (a) Let  $A$  and  $B$  be elements of a partially ordered set, considered as a category. What is the coproduct of  $A$  and  $B$ ?
- (b) Let  $P$  and  $Q$  be objects in  $\mathbf{PoSet}$ , the category of partially ordered sets. What is the coproduct of  $P$  and  $Q$ ?

**Exercise 40** (Pushouts)

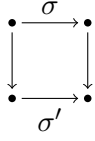
Consider the following diagram:



- (a) Prove that if both squares are pushouts, then the outside rectangle (with top and bottom edges the evident composites) is a pushout.
- (b) Prove that if the outside rectangle and the left-hand square are pushouts and the whole diagram commutes, then the right-hand square is a pushout.
- (c) Try to empirically verify this with some examples in HETS.

**Exercise 41** (Pushouts of theories and theory morphisms)

Consider a pushout of theories and theory morphisms in a weakly semi-exact institution:



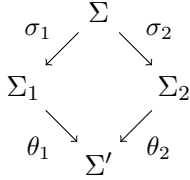
Show that if  $\sigma$  is model-theoretically conservative, then so is  $\sigma'$ .

**Exercise 42** (Pushouts of theories – example)

Formalize the example from the lecture as a pushout of theories.

### 3.3.2 Semi-exactness

**Definition 3.3.4.** An institution  $\mathbf{Sign} \xrightarrow{I} \mathbf{Room}$  is (weakly) semi-exact, if for any pushout of signatures in  $\mathbf{Sign}$



and each pair of models  $M_1 \in \text{Mod}(\Sigma_1), M_2 \in \text{Mod}(\Sigma_2)$  that is compatible, i.e.  $M_1|_{\sigma_1} = M_2|_{\sigma_2}$ , there is a unique (not necessarily unique) amalgamation  $M_1 \oplus M_2 \in \text{Mod}(\Sigma')$  with  $(M_1 \oplus M_2)|_{\theta_1} = M_1$  and  $(M_1 \oplus M_2)|_{\theta_2} = M_2$

2

EdNote(2)

$\diamond \heartsuit \spadesuit$   
 $\diamond \heartsuit @. > @ * [(4)][ur] \heartsuit \spadesuit @ * [(4)] @. > [ul]$   
 $\heartsuit$   
 $@ * [(7)][ul] @ * [(7)][ur]$

Given any models  $M_1$  and  $M_2$  with common reduct  $M_0 \dots$

$\diamond \heartsuit \spadesuit M_1 @ | - > @ * [(7)][ddrr] \in \diamond \heartsuit @. > @ * [(4)][ur] \heartsuit \spadesuit \ni @ * [(4)] @. > [ul] M_2 @ | - > @ * [(7)][dddl]$

... they must have a unique amalgamation  $M_3$

$M_3 @ | - > @ * [(7)][ddrr] @ | - > @ * [(7)][dddl] \diamond \heartsuit \spadesuit M_1 @ | - > @ * [(7)][ddrr] \in \diamond \heartsuit @. > @ * [(4)][ur] \heartsuit \spadesuit$

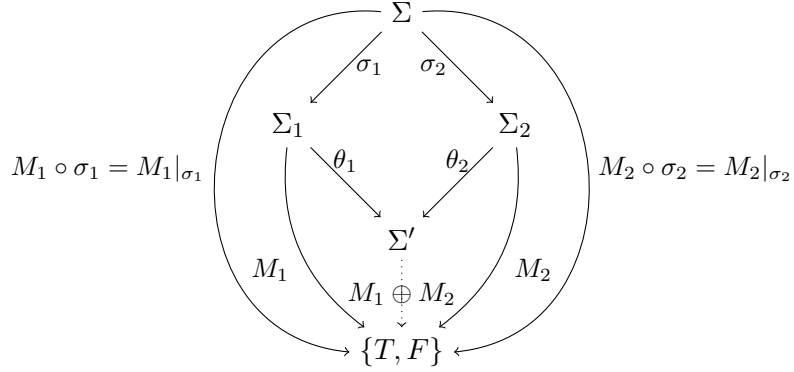
**Proposition 3.3.5.** Propositional logic is semi-exact.

<sup>2</sup>EDNOTE: TODO: update illustration

*Proof.* **Sign** = **Set**,  $\text{Mod}(\Sigma) = \{\Sigma \xrightarrow{M} \{T, F\}\}$

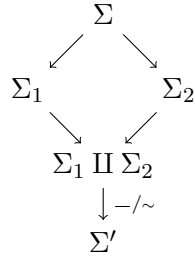
$$\Sigma_1 \xrightarrow{\sigma} \Sigma_2 \xrightarrow{M_2} \{T, F\}$$

$$\xrightarrow{M_2|_{\sigma} = M_2 \circ \sigma}$$



$M_1 \oplus M_2$  uniquely exists due to the universal property of the pushout.

How is  $M_1 \oplus M_2$  constructed?



$$M_1 \oplus M_2([\Pi_1(p_1)]_{\sim}) = M_1(p_1)$$

$$M_1 \oplus M_2([\Pi_2(p_2)]_{\sim}) = M_2(p_2)$$

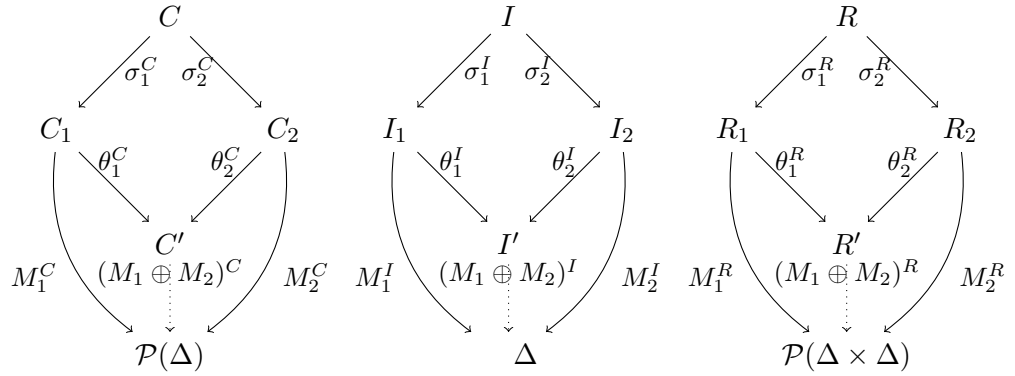
This is well defined, since for  $p \in \Sigma$  :

$$M_1(\sigma_1(p)) = M_2(\sigma_2(p))$$

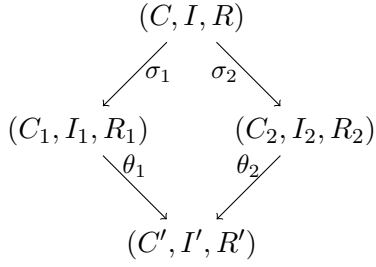
□

**Proposition 3.3.6.** *DL is semi-exact.*

*Proof.* Let  $M_1|_{\sigma_1} = M_2|_{\sigma_2}$  and  $\Delta := \Delta^{M_1} = \Delta^{M_2}$







$$M_1 \oplus M_2 = (\Delta, (M_1 \oplus M_2)^C, (M_1 \oplus M_2)^I, (M_1 \oplus M_2)^R) \quad \square$$

**Proposition 3.3.7.** *FOL is semi-exact.*

*Proof.* Similar to DL but with some technical difficulties due to sorts.  $\square$

**Definition 3.3.8** (Subsorted first-order logic (SubFOL)).

Signatures  $(\Sigma, \leq, F, P)$  where

- $(S, \leq)$  is a preorder (i.e.  $\leq$  is transitive and reflexive)
- $(S, F, P)$  is a FOL-Signature

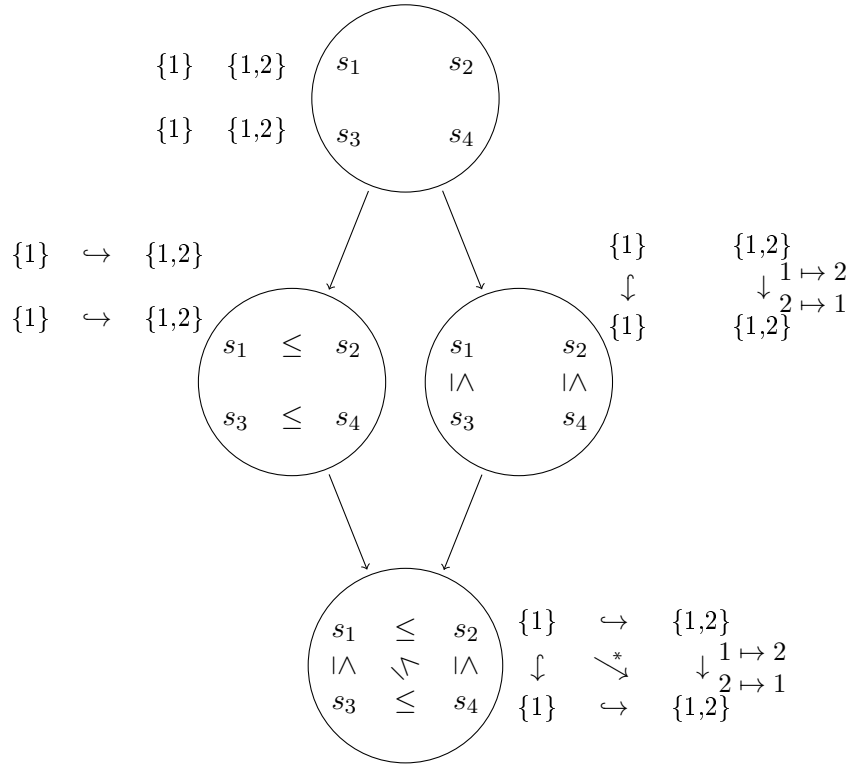
Models like FOL-Models plus: If  $s_1 \leq s_2$  then  $\text{inj}_{s_1 \rightarrow s_2}^M : M_{s_1} \rightarrow M_{s_2}$  injective, such that:

- $\text{inj}_{s \rightarrow s}^M = \text{id}$
- $\text{inj}_{s_2 \rightarrow s_3}^M \circ \text{inj}_{s_1 \rightarrow s_2}^M = \text{inj}_{s_1 \rightarrow s_3}^M \quad (s_1 \leq s_2 \leq s_3)$

shortly:  $(S, \leq) \xrightarrow{M} \mathbf{Set}^{\text{inj}}$

**Proposition 3.3.9.** *SubFOL is not semi-exact.*

*Proof.* In the following pushout there cannot be an amalgamation for the models written next to the signatures



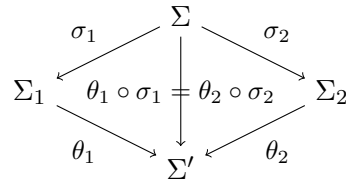
There is no injective function for  $\xrightarrow{*}$ , such that the diagram commutes.  $\square$

**Definition 3.3.10.** In an institution with signature pushouts we can define a normal form of specifications with  $Sig(SP) = \Sigma$  of format:

$$NF(SP) = \langle \Sigma', \Gamma \rangle \text{ hide } \sigma, \quad \text{with } \sigma : \Sigma \rightarrow \Sigma'$$

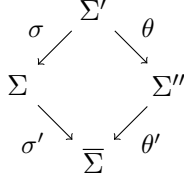
It is uniquely defined as follows:

- $NF(\langle \Sigma, \Gamma \rangle) := \langle \Sigma, \Gamma \rangle \text{ hide } id$
- Let  $NF(SP_i) = \langle \Sigma_i, \Gamma_i \rangle \text{ hide } \sigma_i, \quad i = 1, 2.$  Take a pushout



$$NF(SP_1 \text{ and } SP_2) := \langle \Sigma', \theta_1(\Gamma_1) \cup \theta_2(\Gamma_2) \rangle \text{ hide } (\theta_1 \circ \sigma_1)$$

- Let  $NF(SP) = \langle \Sigma, \Gamma \rangle \text{ hide } \sigma : \Sigma' \rightarrow \Sigma.$  In the pushout



$$NF(SP \text{ with } \theta : \Sigma' \rightarrow \Sigma'') := \langle \bar{\Sigma}, \sigma'(\Gamma) \rangle \text{ hide } \theta'$$

- Let  $NF(SP) = \langle \Sigma, \Gamma \rangle \text{ hide } \sigma : \Sigma' \rightarrow \Sigma$ . Then

$$NF(SP \text{ hide } \theta : \Sigma'' \rightarrow \Sigma') = \langle \Sigma, \Gamma \rangle \text{ hide } \sigma \circ \theta$$

**Proposition 3.3.11.** *IN a semi-exact institution  $Mod(SP) = Mod(NF(SP))$*

*Proof.* Induction over  $SP$ :

- $Mod(\langle \Sigma, \Gamma \rangle) = Mod(\langle \Sigma, \Gamma \rangle \text{ hide id}) \checkmark$
- By i.h. assume  $Mod(SP_i) = Mod(\langle \Sigma_i, \Gamma_i \rangle \text{ hide } \sigma_i) \quad i = 1, 2$ 
  - Prove  $Mod(SP_1 \text{ and } SP_2) \subseteq Mod(NF(SP_1 \text{ and } SP_2))$ :  
Let  $M \in Mod(SP_1) \cap Mod(SP_2)$ ,  
hence by i.h.  $M \in Mod(\langle \Sigma_i, \Gamma_i \rangle \text{ hide } \sigma_i)$  for  $i = 1, 2$ .  
That is, there is a  $M_i \in Mod(\Sigma_i, \Gamma_i)$  with  $M_i|_{\sigma_i} = M$ . Hence  $M_1 \oplus M_2$  exists with  $M_1 \oplus M_2|_{\theta_i} \models \Gamma_i$ . By the satisfaction condition,  $M_1 \oplus M_2 \models \theta_i(\Gamma_i)$ . Hence  $M_1 \oplus M_2 \in Mod(\langle \Sigma', \theta_1(\Gamma_1) \cup \theta_2(\Gamma_2) \rangle)$ , witnessing that  $M \in Mod(NF(SP_1 \text{ and } SP_2))$
  - Prove  $Mod(NF(SP_1 \text{ and } SP_2)) \subseteq Mod(SP_1 \text{ and } SP_2)$ :  
Let  $M \in Mod(\langle \Sigma', \theta_1(\Gamma_1) \cup \theta_2(\Gamma_2) \rangle \text{ hide } \theta_1 \circ \sigma_1)$ . Hence there is  $M'$  with  $M'|_{\theta_1 \circ \sigma_1} = M$  and  $M' \models \theta_1(\Gamma_1) \cup \theta_2(\Gamma_2)$ . By the satisfaction condition  $M'|_{\theta_i}$  is a  $\sigma_i$ -expansion of  $M$  satisfying  $\Gamma_i$ . Hence  $M \in Mod(\langle \Sigma_i, \Gamma_i \rangle \text{ hide } \sigma_i) = Mod(SP_i)$
  - Translation and hiding similarly.

□

**Exercise 43** (Normal forms of structured specifications)

- Complete the proof that in a semi-exact institution, building normal forms preserves the model class (the cases of translation and hiding are missing).
- Does the result hold for weakly semi-exact institutions as well?

**Exercise 44** (Subsorts)

Consider the following specification:

```

spec sp1 =
  sorts Man, Woman < Person
  sort Hybrid < Man
  sort Hybrid < Woman
end

spec sp2 =
  sorts Man, Woman < Person
  sorts Female < Person
  forall p: Person
  . p in Woman => p in Female
  . p in Man => not p in Female
end

spec sp = {sp1 hide Hybrid} and {sp2 hide Female} end

```

- (a) Use the semantics of structured specifications to argue that the model class of  $sp$  is empty (i.e.,  $sp$  is inconsistent). Use the proof calculus for structured specifications to derive  $sp \vdash \perp$ .
- (b) Compute the normal form and prove that  $sp$  is inconsistent.
- (c) Try out HETS. Extend the above specification by:

```

spec spimplies = sp
then %implies
  . false
end

```

 and then use “Edit -> Proofs -> TheoremHideShift” to compute the normal form.

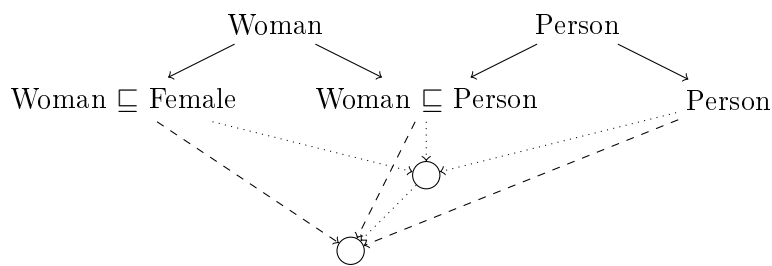
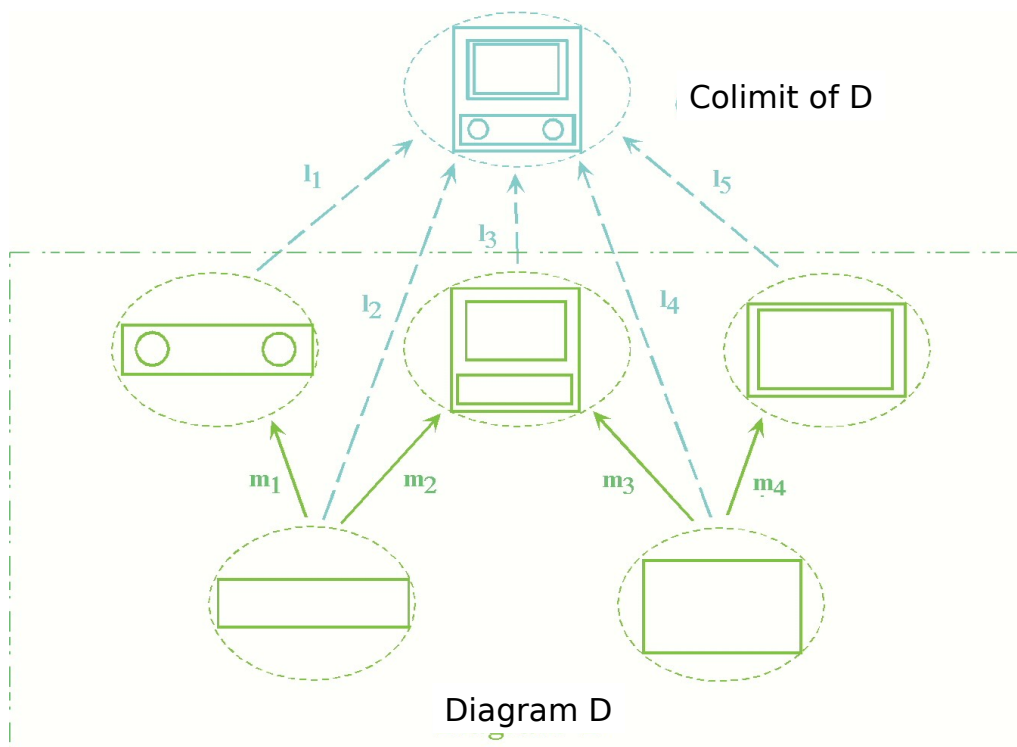


Figure 3.2: Ontology alignment

### 3.3.3 Colimits in general

## A colimit combines everything



- Motivation: Goguen quote (see above)
- Ontology alignment (cf. Figure 3.2)

**Definition 3.3.12.** A diagram is a functor  $\mathbf{I} \rightarrow \mathbf{C}$  where  $\mathbf{I}$  is a small (index) category (i.e. with a set of objects)

**Definition 3.3.13.** Given a diagram  $D : \mathbf{I} \rightarrow \mathbf{C}$ , a sink is a family of morphisms  $\mu = (\mu_i : D_i \rightarrow C)_{i \in |\mathbf{I}|}$  into a common object  $C$ . ( $D_i = D(i)$ )




**Definition 3.3.14.** A cocone is a sink  $\mu$  such that for each  $d : i \rightarrow j \in \mathbf{I}$  the following commutes:

$$\begin{array}{ccc} D_i & \xrightarrow{Dd} & D_j \\ & \searrow \mu_i & \swarrow \mu_j \\ & C & \end{array}$$

**Definition 3.3.15.** A colimiting cocone is a cocone  $\mu : D \rightarrow C$  such that for any other cocone  $\mu' : D \rightarrow C'$  there exists a unique  $h : C \rightarrow C'$ , such that for all  $i \in |\mathbf{I}|$

$$\begin{array}{ccc} & \mu_i & \rightarrow C \\ D_i & \searrow & \downarrow h \\ & \mu'_i & \rightarrow C' \end{array} \text{ commutes.}$$

**Example 3.3.16.**

- Pushouts are colimits for diagrams of the shape 
- Binary coproducts are colimits for diagrams of the shape 
- Initial objects are colimits for diagrams of the empty shape ( $\mathbf{I} = \emptyset$ )
- Arbitrary coproducts:  $\mathbf{I}$  is discrete (all morphisms are identities).  
Notation:  $\coprod_{i \in \mathbf{I}} A_i$
- Coequalizers: 

**Definition 3.3.17.** Given  $A \begin{array}{c} \xrightarrow{f} \\ \xrightarrow{g} \end{array} B$  a coequalizer is an arrow  $B \xrightarrow{c} C$

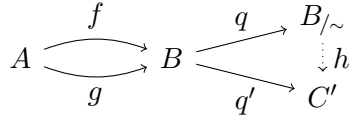
with  $c \circ f = c \circ g$  such that for any  $B \xrightarrow{c'} C'$  with  $c' \circ f = c' \circ g$  there is a unique  $h : C \rightarrow C'$  with

$$\begin{array}{ccccc} A & \begin{array}{c} \xrightarrow{f} \\ \xrightarrow{g} \end{array} & B & \begin{array}{c} \xrightarrow{c} \\ \xrightarrow{c'} \end{array} & \begin{array}{c} C \\ C' \end{array} \\ & & & & \downarrow h \end{array}$$

**Example 3.3.18.** Coequalizers in Set

$$A \begin{array}{c} \xrightarrow{f} \\ \xrightarrow{g} \end{array} B \xrightarrow{q} B/\sim$$

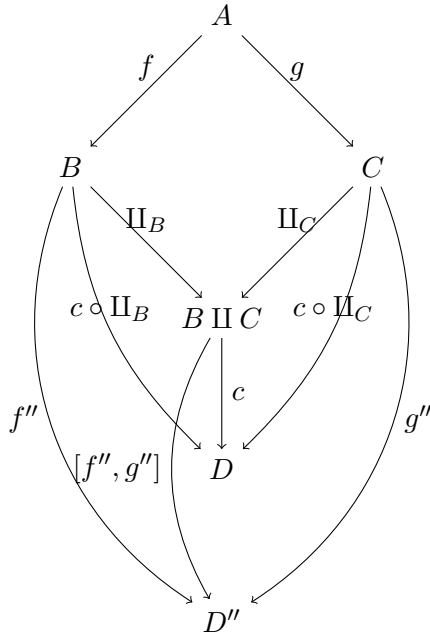
The equivalence relation  $\sim$  is generated by  $f(a) \sim g(a)$  for  $a \in A$ . Define  $q(b) := [b]_{\sim}$ . Then clearly:  $q \circ f = q \circ g$ .



Define  $h([b]_{\sim}) = q'(b)$ . This is well defined since  $b \sim b' \Rightarrow q'(b) = q'(b')$ .  
 $q'(f(a)) = q'(g(a))$ .

**Theorem 3.3.19.** *Pushouts are coequalizers of coproducts.*

*Proof.* Consider the following diagram



$$\begin{aligned}
& [f'', g''] \circ \Pi_B \circ f \\
& = f'' \circ f \\
& = g'' \circ g \\
& = [f'', g''] \circ \Pi_C \circ g
\end{aligned}$$

Since  $C$  is universal with □

$$c \circ \Pi_B \circ f = c \circ \Pi_C \circ g$$

we have a unique  $h : D \rightarrow D''$   
with  $h \circ c = [f'', g'']$  iff  
 $h \circ c \circ \Pi_B = f''$  and  $h \circ c \circ \Pi_C = g''$   
(universal property of coproduct)

### 3.4 Natural transformations

*Recall:* An Institution is a functor  $\mathbf{Sign} \xrightarrow{I} \mathbf{Room}$ , or by the alternative definition

- a category  $\mathbf{Sign}$  of signatures
- a functor  $\text{Sen} : \mathbf{Sign} \rightarrow \text{Set}$
- a functor  $\text{Mod} : \mathbf{Sat}^{\text{op}} \rightarrow \text{Set}$
- for each  $\Sigma \in |\mathbf{Sign}|$ , a satisfaction relation  $\models_{\Sigma} \subseteq |\text{Mod}(\Sigma)| \times \text{Sen}(\Sigma)$

such that for any signature morphism  $\sigma : \Sigma' \rightarrow \Sigma' \in \mathbf{Sign}$  and  $M' \in \text{Mod}(\Sigma'), \phi \in \text{Sen}(\Sigma)$

$$M'|_{\sigma} \models \phi \text{ iff } M' \models \sigma(\phi)$$

Natural transformations are morphisms between functors.

**Definition 3.4.1.** Given two functors  $F, G : \mathbf{C} \rightarrow \mathbf{D}$  a natural transformation  $\tau : F \rightarrow G$  is a family of morphism  $(\tau_C : F_C \rightarrow G_C)_{C \in |\mathbf{C}|}$ <sup>1</sup> such that for any  $f : C_1 \rightarrow C_2 \in \mathbf{C}$

$$\begin{array}{ccc} F_{C_1} & \xrightarrow{\tau_{C_1}} & G_{C_1} \\ Ff \downarrow & & \downarrow Gf \\ F_{C_2} & \xrightarrow{\tau_{C_2}} & G_{C_2} \end{array} \quad \text{commutes}$$

**Example 3.4.2.**  $i : \text{Mod}^{\text{SubFOL}(\leq)} \rightarrow \text{Mod}^{\text{SubFOL}}$   
for any  $\text{SubFOL}(\leq)$  signature  $\Sigma$  we have

$$\begin{array}{ccc} \text{Mod}^{\text{SubFOL}(\leq)}(\Sigma_1) & \xrightarrow{\iota_{\Sigma_1}} & \text{Mod}^{\text{SubFOL}}(\Sigma_1) \\ \uparrow \text{Mod}^{\text{SubFOL}(\leq)}(\sigma) & & \uparrow \text{Mod}^{\text{SubFOL}}(\sigma) \\ \text{Mod}^{\text{SubFOL}(\leq)}(\Sigma_2) & \xrightarrow{\iota_{\Sigma_2}} & \text{Mod}^{\text{SubFOL}}(\Sigma_2) \end{array}$$

**Example 3.4.3.** Given a diagram  $D : \mathbf{I} \rightarrow \mathbf{C}$  a cocone  $\mu : D \rightarrow C = (C, (\mu_i : D_i \rightarrow C)_{i \in \mathbf{I}})$  is the same as the natural transformation

$$\mu : D \rightarrow \text{const}(c) \text{ where } \text{const}(c) : \mathbf{I} \rightarrow \mathbf{C}, \text{ with}$$

$$\begin{array}{ccc} \iota_1 & \longrightarrow & C \\ \lambda \downarrow & \longrightarrow & \downarrow \text{id}_C \\ \iota_2 & \longrightarrow & C \end{array}$$

Naturality means that for any  $d : i \rightarrow j \in \mathbf{I}$

$$\begin{array}{ccc} D(i) & \xrightarrow{\mu_i} & C \\ D(d) \downarrow & & \downarrow \text{id}_C \\ D(j) & \xrightarrow{\mu_j} & C \end{array}$$

**Example 3.4.4.** Given categories  $\mathbf{C}, \mathbf{D}$  the functor category has as its object the functors  $F : \mathbf{C} \rightarrow \mathbf{D}$  and the morphisms:  $\tau : F \rightarrow G$  are natural transformations.

**Exercise 45** (Functor categories)

Given categories  $\mathbf{C}$  and  $\mathbf{D}$ , prove that  $[\mathbf{C}, \mathbf{D}]$  (functors from  $\mathbf{C}$  to  $\mathbf{D}$  as objects and natural transformations as morphisms) indeed forms a category.

---

<sup>1</sup>  $F_C$  and  $G_C$  live in  $\mathbf{D}$ .



### 3.4.1 Institution comorphisms

Motivation:

- Embeddings or encodings between institutions
- Re-use (“borrowing”) of proof tools
- heterogeneous specification

**Definition 3.4.5.** *Given Institutions  $I = (\mathbf{Sign}, Sen, Mod, \models)$  and  $I' = (\mathbf{Sign}', Sen', Mod', \models')$  an institution comorphism  $\rho(\phi, \alpha, \beta) : I \rightarrow I'$  consists of*

- a functor  $\phi : \mathbf{Sign} \rightarrow \mathbf{Sign}'$
- a natural transformation  $\alpha : Sen \rightarrow Sen' \circ \phi$
- a natural transformation  $\beta : Mod' \circ \phi^{OP} \rightarrow Mod$ , such that the following diagrams commute.

$$\begin{array}{ccc}
 Sen(\Sigma_1) & \xrightarrow{\alpha_{\Sigma_1}} & Sen'(\phi\Sigma_1) \\
 \downarrow Sen(\sigma) & & \downarrow Sen'(\phi\sigma) \\
 Sen(\Sigma_2) & \xrightarrow{\alpha_{\Sigma_2}} & Sen'(\phi\Sigma_2) \\
 \\ 
 Mod(\Sigma_1) & \xleftarrow{\alpha_{\Sigma_1}} & Mod'(\phi\Sigma_1) \\
 \uparrow \_|\sigma Mod(\sigma) & & \uparrow \_|\phi\sigma Mod'(\phi\sigma) \\
 Mod(\Sigma_2) & \xleftarrow{\alpha_{\Sigma_2}} & Mod'(\phi\Sigma_2)
 \end{array}$$

**Example 3.4.6.** *Propositional  $\rightarrow$  FOL*

- $\alpha_{\Sigma}(\phi) = \phi$
- Given  $M' \in Mod^{FOL}(\phi\Sigma)$ ,  $\beta_{\Sigma}(M')(p) = \begin{cases} T & , \text{ if } () \in M'_p \\ F & , \text{ if } () \notin M'_p \end{cases}$
- $\beta_{\Sigma}(M') \models \varphi$  iff  $M' \models \varphi$

*Proof.* Induction over  $\varphi$ , because propositional logic is the same in Propositional and FOL.  $\square$

**Example 3.4.7.** *ALC  $\rightarrow$  SROIQ (“Sublogic embedding”)*

- $\phi = id$
- $\alpha_\Sigma(\varphi) = \varphi$
- $\beta_\Sigma = id$
- *satisfaction condition trivially holds*

$$\begin{array}{ccc}
Sen^{ALC}(\Sigma) & \longrightarrow & Sen^{SROIQ}(\Sigma) \\
Sen^{ALC}(\sigma) \downarrow & & \downarrow Sen^{SROIQ}(\sigma) \\
Sen^{ALC}(\Sigma') & \longrightarrow & Sen^{SROIQ}(\Sigma')
\end{array}$$

**Example 3.4.8.**  $\mathcal{ALC} \rightarrow \text{FOL}$

- $\phi((C, \mathbf{R}, \mathbf{I})) = (S, F, P)$  with
  - $S = \{Thing\}$
  - $F = \{a : Thing \mid a \in \mathbf{I}\}$
  - $P = \{A : Thing \mid A \in \mathbf{C}\} \cup \{R : Thing \times Thing \mid R \in \mathbf{R}\}$
- $\phi(\sigma) = \theta$  with

$$\begin{aligned}
\theta(Thing) &= Thing \\
\theta(I^a : Thing) &= \sigma^I(a) : Thing \\
\theta(R : Thing \times Thing) &= \sigma^R(R)
\end{aligned}$$

- *Concept translation*
  - $\alpha_x(A) = A(x : Thing)$
  - $\alpha_x(\neg C) = \neg \alpha_x(C)$
  - $\alpha_x(C \sqcap D) = \alpha_x(C) \wedge \alpha_x(D)$
  - $\alpha_x(C \sqcup D) = \alpha_x(C) \vee \alpha_x(D)$
  - $\alpha_x(\exists R.C) = \exists y : Thing. (R(x, y) \wedge \alpha_y(C))$
  - $\alpha_x(\forall R.C) = \forall y : Thing. (R(x, y) \rightarrow \alpha_y(C))$
- *Sentence translation*
  - $\alpha_\Sigma(C \sqsubseteq D) = \forall x : Thing. (\alpha_x(C) \rightarrow \alpha_x(D))$
  - $\alpha_\Sigma(a : C) = \alpha_x(C)[a/x]^1$
  - $\alpha_\Sigma(R(a, b)) = R(a, b)$
- *Model reduction*

---

<sup>1</sup>Replace  $x$  by  $a$ .

- For  $M' \in \text{Mod}^{FOL}(\phi\Sigma)$  define  $\beta_\Sigma(M') := (\Delta, \cdot^I)$  with  $\Delta = M'_{Thing}$  and  $A^I = M'_A, a^I = M'_a, R^I = M'_R$ .

**Proposition 3.4.9.**  $C^{\mathcal{I}} = \{m \in M'_{Thing} \mid M' + \{x \mapsto m\} \models \alpha_x(C)\}$

*Proof.* By Induction over the structure of  $C$ .

- $A^{\mathcal{I}} = M'_A = \{m \in M'_{Thing} \mid M' + \{x \mapsto m\} \models A(x)\}$
- $(\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}} \stackrel{I.H.}{=} \Delta \setminus \{m \in M'_{Thing} \mid M' + \{x \mapsto m\} \models \alpha_x(C)\} = \{m \in M'_{Thing} \mid M' + \{x \mapsto m\} \models \neg\alpha_x(C)\}$

□

The satisfaction condition holds as well.

**Example 3.4.10.**  $\rho : \mathcal{ALC} \rightarrow FOL$  can be extended to  $\rho : \mathcal{SROIQ} \rightarrow FOL$ .  
For example qualified number restrictions

$$\begin{aligned} \alpha_x(\geq nR.C) &= \exists^{\geq n} y : \text{Thing}. R(x, y) \wedge \alpha_y(C) \\ &= \exists y_1, \dots, y_n : \text{Thing}. \bigwedge_{i=1}^n R(x, y_i) \wedge \alpha_{y_i}(C) \wedge \bigwedge_{i \neq j} y_i \neq y_j \\ \alpha_x(\leq nR.C) &= \exists^{\leq n} y : \text{Thing}. R(x, y) \wedge \alpha_y(C) \\ &= \neg \exists^{\geq n+1} y : \text{Thing}. R(x, y) \wedge \alpha_y(C) \end{aligned}$$

**Exercise 46** (Institution comorphisms I)

Spell out the definition of institution comorphisms in terms of the “institutions as functors” definition  $I : \mathbf{Sign} \rightarrow \mathbf{Room}$ .

**Exercise 47** (Institution comorphisms II)

Define a model-expansive institution comorphism  $\text{Propositional} \rightarrow \mathcal{ALC}$ .

**Exercise 48** (Satisfaction relations as natural transformations)

Show that the satisfaction relation in an institution is a natural transformation  $\models : \text{Gr} \circ \text{Sen} \rightarrow \overline{\text{Gr}} \circ U^{\text{op}} \circ \text{Mod}^{\text{op}}$ , where  $\text{Gr} : \mathbf{Set} \rightarrow \mathbf{Rel}$  and  $\overline{\text{Gr}} : \mathbf{Set}^{\text{op}} \rightarrow \mathbf{Rel}$  map a function to its graph.

**Exercise 49** (Comorphisms in HETS)

Experiment with HETS: Translate theories that you have used along comorphisms.

### 3.5 Borrowing

Motivation: re-use of proof tools.

**Definition 3.5.1.** *An institution comorphism  $\rho = (\phi, \alpha, \beta) : \mathcal{I} \rightarrow \mathcal{I}'$  is model expansive, if for each  $\Sigma \in |\mathbf{Sign}|$ ,  $\beta_\Sigma : \text{Mod}'(\phi\Sigma) \rightarrow \text{Mod}(\Sigma)$  is surjective on objects.*

**Theorem 3.5.2** (Borrowing). *Let  $\rho = (\phi, \alpha, \beta) : \mathcal{I} \rightarrow \mathcal{I}'$  be a comorphism,  $\Sigma \in |\mathbf{Sign}|$ ,  $\Gamma \subseteq \text{Sen}(\Sigma)$  and  $\phi \in \text{Sen}(\Sigma)$ . Then*

- (a)  $\Gamma \models_\Sigma \phi \Rightarrow \alpha_\Sigma(\Gamma) \models_{\phi\Sigma} \alpha_\Sigma(\phi)$  (“completeness”)
- (b) *If  $\rho$  is model expansive, then*  
 $\Gamma \models_\Sigma \phi \Leftarrow \alpha_\Sigma(\Gamma) \models_{\phi\Sigma} \alpha_\Sigma(\phi)$  (“soundness”)

*Proof.*

- 1a. Assume  $\Gamma \models_\Sigma \phi$ . If  $M' \models_{\phi\Sigma} \alpha_\Sigma(\Gamma)$ , then  $\beta_\Sigma(M') \models_\Sigma \Gamma$ . By assumption  $\beta_\Sigma(M') \models_\Sigma \phi$ . By the satisfaction condition  $M' \models_{\phi\Sigma} \alpha_\Sigma(\phi)$ .
- 1b. Assume  $\alpha_\Sigma(\Gamma) \models_{\phi\Sigma} \alpha_\Sigma(\phi)$ . Let  $M \models \Gamma$ . Since  $\beta_\Sigma$  is surjective, we get  $M' \in \text{Mod}(\phi\Sigma)$  with  $\beta_\Sigma(M') = M$ . Hence  $\beta_\Sigma(M') \models_\Sigma \Gamma$ . By the satisfaction condition  $M' \models_{\phi\Sigma} \alpha_\Sigma(\Gamma)$ . By assumption  $M' \models_{\phi\Sigma} \alpha_\Sigma(\phi)$ . By the satisfaction condition  $M = \beta_\Sigma(M') \models_\Sigma \phi$ .

□

**Proposition 3.5.3.** *All comorphisms introduced so far are model expansive.*

**Example 3.5.4.**  $\rho = (\phi, \alpha, \beta) : \text{FOL} \rightarrow \text{Prop}$  with

- $\phi(S, F, P) = \biguplus_{w \in s^*} P_w$
- $\alpha_\Sigma(t_1 = t_2) = \top$   
 $\alpha_\Sigma(p_w(t_1, \dots, t_n)) = p_w$   
 $\alpha_\Sigma(\top) = \top$   
 $\alpha_\Sigma(\perp) = \perp$   
 $\alpha_\Sigma(\neg\varphi) = \neg\alpha_\Sigma(\varphi)$  *similarly for  $\wedge, \vee, \dots$*   
 $\alpha_\Sigma(\forall x : s. \varphi) = \alpha_\Sigma(\varphi)$   
 $\alpha_\Sigma(\exists x : s. \varphi) = \alpha_\Sigma(\varphi)$
- $\beta_\Sigma(M' : (\biguplus_{w \in s^*} P_w) \rightarrow \{T, F\}) = M$   
*with  $M_s = \{*\}$  and*
  - $(f_{w,s})_M(*, \dots, *) = *$  *for  $f \in F_{w,s}$*
  - $(*, \dots, *) \in p_w$  *iff  $M'(p_w) = T$*

*Satisfaction condition follows by induction.*

*This is not model expansive. Still we have  $\Gamma \models_{\Sigma} \varphi \Rightarrow \alpha_{\Sigma}(\Gamma) \models_{\phi\Sigma} \alpha_{\Sigma}(\varphi)$ , i.e.  $\alpha_{\Sigma}(\Gamma) \not\models_{\phi\Sigma} \alpha_{\Sigma}(\varphi) \Rightarrow \Gamma \not\models_{\Sigma} \varphi$ . That is, we can use a SAT-solver for disproving FOL theorems.*

**Exercise 50** (Satisfaction condition of institution comorphisms)

For the institution comorphism  $\text{FOL} \rightarrow \text{Propositional}$  from the lecture, show the satisfaction condition.

**Exercise 51** (Semi-exactness of institution comorphisms)

Show that if the model translation of an institution comorphism consists of isomorphisms, then the comorphism is semi-exact.

**Exercise 52** (Borrowing for structured specifications)

Complete the proof of the following fact:

For a comorphism  $\rho = (\Phi, \alpha, \beta) : I \rightarrow I'$ , a signature  $\Sigma \in |\mathbf{Sign}|$ , a  $\Sigma$  specification SP, and a model  $M' \in \text{Mod}'(\Sigma)$ ,

$$M' \in \text{Mod}'(\hat{\rho}(\text{SP})) \text{ implies } \beta_{\Sigma}(M') \in \text{Mod}(\text{SP}).$$

The case of hiding was missing.

**Exercise 53** (Preservation of pushouts and normal forms)

Consider the following theorem from the lecture:

If the signature translation of an institution comorphism  $\rho$  preserves pushouts, then  $\hat{\rho}$  preserves normal forms.

Experiment with HETS (computation of normal forms and translation of development graphs along comorphisms).

Do all the comorphisms in HETS preserve pushouts?

Show that those coding out subsorting do not.

### 3.5.1 Borrowing for structured specifications

**Definition 3.5.5.** *An institution comorphism  $\rho = (\phi, \alpha, \beta) : \mathcal{I} \rightarrow \mathcal{I}'$  is (weakly) semi-exact, if for each  $\sigma : \Sigma_1 \rightarrow \Sigma_2 \in \mathbf{Sign}$*

$$\begin{array}{ccc} \text{Mod}(\Sigma_1) & \xleftarrow{\beta_{\Sigma_1}} & \text{Mod}'(\phi\Sigma_1) \\ \uparrow \text{---} | \sigma & & \uparrow \text{---} | \phi\sigma \\ \text{Mod}(\Sigma_2) & \xleftarrow{\beta_{\Sigma_2}} & \text{Mod}'(\phi\Sigma_2) \end{array}$$

and any  $M_2 \in \text{Mod}(\Sigma_2)$  and  $M'_1 \in \text{Mod}'(\phi\Sigma_1)$  with  $M_2|_{\sigma} = \beta_{\Sigma_1}(M'_1)$  there exists a unique (not necessarily unique)  $M'_2 \in \text{Mod}'(\phi\Sigma_2)$  with  $\beta_{\Sigma_2}(M'_2) = M_2$  and  $M'_2|_{\phi\sigma} = M'_1$

**Definition 3.5.6.** Given a comorphism  $\rho = (\phi, \alpha, \beta) : \mathcal{I} \rightarrow \mathcal{I}'$  the translation  $\hat{\rho}$  of structured specifications in  $\mathcal{I}$  is defined as follows:

$$\begin{aligned}\hat{\rho}(\langle \Sigma, \Gamma \rangle) &:= \langle \phi\Sigma, \alpha_{\Sigma}(\Gamma) \rangle \\ \hat{\rho}(SP_1 \text{ and } SP_2) &:= \hat{\rho}(SP_1) \text{ and } \hat{\rho}(SP_2) \\ \hat{\rho}(SP \text{ with } \sigma) &:= \hat{\rho}(SP) \text{ with } \phi\sigma \\ \hat{\rho}(SP \text{ hide } \sigma) &:= \hat{\rho}(SP) \text{ hide } \phi\sigma\end{aligned}$$

**Theorem 3.5.7.** If  $\text{Sig}(SP) = \Sigma$ , then  $\text{Sig}(\hat{\rho}(SP)) = \phi\Sigma$

*Proof.* Easy induction. □

**Theorem 3.5.8.**  $M' \in \text{Mod}'(\hat{\rho}(SP)) \Rightarrow \beta_{\text{Sig}(SP)}(M') \in \text{Mod}(SP)$

*Proof.* Induction over  $SP$

- $SP = \langle \Sigma, \Gamma \rangle$   
Let  $M' \in \text{Mod}'(\hat{\rho}(SP))$ , i.e.  $M' \models \alpha_{\Sigma}(\Gamma)$   
and thus  $\beta_{\Sigma}(M') \models \Gamma$ ,  
so  $\beta_{\Sigma}(M') \in \text{Mod}(\langle \Sigma, \Gamma \rangle)$
- $SP = SP_1 \text{ and } SP_2$   
Let  $M' \in \text{Mod}'(\hat{\rho}(SP_1 \text{ and } SP_2))$ ,  
i.e.  $M' \in \text{Mod}'(\hat{\rho}(SP_1))$  and  $M' \in \text{Mod}'(\hat{\rho}(SP_2))$ .  
By i.h. we have  $\beta_{\Sigma}(M') \in \text{Mod}(SP_1)$  and  $\beta_{\Sigma}(M') \in \text{Mod}(SP_2)$ ,  
so  $\beta_{\Sigma}(M') \in \text{Mod}(SP_1 \text{ and } SP_2)$ .
- $SP = SP' \text{ with } \sigma$   
Let  $M' \in \text{Mod}(\hat{\rho}(SP' \text{ with } \sigma))$ ,  
i.e.  $M' \in \text{Mod}(\hat{\rho}(SP') \text{ with } \phi\sigma)$ .  
By definition  $M|_{\phi\sigma} \in \text{Mod}'(\hat{\rho}(SP'))$ .  
By i.h.  $\beta_{\Sigma_1}(M|_{\phi\sigma}) \in \text{Mod}(SP')$ .  
Hence  $\beta_{\Sigma_2}(M)|_{\sigma} \in \text{Mod}(SP')$   
and thus  $\beta_{\Sigma_2}(M) \in \text{Mod}(SP' \text{ with } \sigma) = \text{Mod}(SP)$
- $SP = SP' \text{ hide } \sigma$   
exercise

□

**Theorem 3.5.9.** Let  $\rho = (\phi, \alpha, \beta) : \mathcal{I} \rightarrow \mathcal{I}'$  be weakly semi-exact. Then  $\beta_{\text{Sig}(SP)}(M') \in \text{Mod}(SP) \Rightarrow M' \in \text{Mod}(\hat{\rho}(SP))$

*Proof.* Induction over  $SP$

- $SP = \langle \Sigma, \Gamma \rangle$   
Use satisfaction condition.
- $SP = SP_1$  and  $SP_2$   
Use induction hypothesis.
- $SP = SP'$  with  $\sigma : \Sigma_1 \rightarrow \Sigma_2$   
Let  $\beta_{\Sigma_2}(M') \in \text{Mod}(SP'$  with  $\sigma)$   
 $\Rightarrow \beta_{\Sigma_2}(M')|_{\sigma} \in \text{Mod}(SP')$   
 $\Rightarrow \beta_{\Sigma_1}(M'|_{\phi\sigma}) \in \text{Mod}(SP')$   
 $\Rightarrow$  *i.h.*  $M'|_{\phi\sigma} \in \text{Mod}(\hat{\rho}(SP'))$   
 $\Rightarrow M' \in \text{Mod}(\hat{\rho}(SP')$  with  $\phi\sigma) = \text{Mod}(\hat{\rho}(SP))$
- $SP = SP'$  hide  $\sigma : \Sigma_1 \rightarrow \Sigma_2$   
Let  $\beta_{\Sigma_1}(M') \in \text{Mod}(SP'$  hide  $\sigma)$   
 $\Rightarrow$  There is  $M'' \in \text{Mod}(SP')$  with  $M''|_{\sigma} = \beta_{\Sigma_1}(M')$   
By weak semi-exactness, there is  $\bar{M} \in \text{Mod}'(\phi\Sigma_2)$  with  
 $\bar{M}|_{\phi\sigma} = M'$  and  $\beta_{\Sigma_2}(\bar{M}) = M''$   
 $\Rightarrow \beta_{\Sigma_2}(\bar{M}) \in \text{Mod}(SP')$  and by i.h.  $\bar{M} \in \text{Mod}(\hat{\rho}(SP'))$   
witnessing that  $\bar{M}|_{\phi\sigma} = M' \in \text{Mod}(\hat{\rho}(SP')$  hide  $\phi\sigma) = \text{Mod}(\hat{\rho}(SP))$

□

**Theorem 3.5.10.** *Let  $\rho = (\phi, \alpha, \beta) : I \rightarrow I'$  be a model-expansive and weakly semi-exact comorphism. Let  $SP$  be a  $\Sigma$ -specification and  $\varphi \in \text{Sen}(\Sigma)$ . Then*

$$SP \models_{\Sigma} \varphi \text{ iff } \hat{\rho}(SP) \models'_{\psi\Sigma} \alpha_{\Sigma}(\varphi)$$

*Proof.*

- $\Rightarrow$ : Let  $SP \models_{\Sigma} \varphi$  and  $M' \in \text{Mod}'(\hat{\rho}(SP))$ . Then  $\beta_{\Sigma}(M') \in \text{Mod}(SP)$ , hence  $\beta_{\Sigma}(M') \models_{\Sigma} \varphi$ . By sat. cond.  $M' \models'_{\psi\Sigma} \alpha_{\Sigma}(\varphi)$ .
- $\Leftarrow$ : Assume  $\hat{\rho}(SP) \models \alpha_{\Sigma}(\varphi)$ . Let  $M \in \text{Mod}(SP)$ . By surjectivity of  $\beta_{\Sigma}$ , there is  $M' \in \text{Mod}(\hat{\rho}(SP))$  with  $\beta_{\Sigma}(M') = M$ . By assumption  $M' \models \alpha_{\Sigma}(\varphi)$ . By sat. cond.  $\beta_{\Sigma}(M') = M \models \varphi$ .

□

**Theorem 3.5.11.** *Let  $\rho = (\phi, \alpha, \beta) : I \rightarrow I'$  be a comorphism such that  $\phi$  preserves pushouts. Then*

$$NF(\hat{\rho}(SP)) = \hat{\rho}(NF(SP))$$

## 3.6 Free specifications

### 3.6.1 Model homomorphisms

*Recall:* Institutions are functors  $\mathbf{Sign} \xrightarrow{I} \mathbf{Room}$ , where a room contains a functor  $\mathbf{Sign}^{\text{op}} \xrightarrow{\text{Mod}} \mathbf{CAT}$ , mapping a signature to its *model category*.

#### Propositional logics

**Definition 3.6.1.**  $M \leq M'$  iff for all  $p \in \Sigma : M(p) = T \Rightarrow M'(p) = T$ .

This defines a partial order on  $\Sigma$ -models and hence a category, i.e. there is a (unique) morphism  $M \rightarrow M'$  iff  $M \leq M'$

#### Description logics

**Definition 3.6.2.** A homomorphism  $h : (\Delta, \cdot^{\mathcal{I}}) \rightarrow (\Delta', \cdot^{\mathcal{I}'})$  is a function  $h : \Delta \rightarrow \Delta'$  such that

- $x \in C^{\mathcal{I}} \Rightarrow h(x) \in C^{\mathcal{I}'}$
- $(x, y) \in R^{\mathcal{I}} \Rightarrow (h(x), h(y)) \in R^{\mathcal{I}'}$
- $h(a^{\mathcal{I}}) = a^{\mathcal{I}'}$

#### First-order logic

**Definition 3.6.3.** Let  $\Sigma = (S, F, P)$  and  $M, M' \in \text{Mod}(\Sigma)$ . A homomorphism  $h : M \rightarrow M'$  is a family  $(h_s : M_s \rightarrow M'_s)_{s \in S}$ , such that for any  $f \in F_{w,s}, w = s_1, \dots, s_n$

$$h_s((f_{w,s})_M(x_1, \dots, x_n)) = (f_{w,s})_{M'}(h_{s_1}(x_1), \dots, h_{s_n}(x_n)), \text{ and}$$

$$(x_1, \dots, x_n) \in (P_w)_M = (h_{s_1}(x_1), \dots, h_{s_n}(x_n)) \in (P_w)_{M'}$$

In category notation this means that the following two diagrams have to commute



$$\begin{array}{ccc}
M_{s_1} \times \cdots \times M_{s_n} & \xrightarrow{(f_{w,s})_M} & M_s \\
\downarrow h_{s_1} \times \cdots \times h_{s_n} & & \downarrow h_s \\
M'_{s_1} \times \cdots \times M'_{s_n} & \xrightarrow{(f_{w,s})_{M'}} & M_{s'}
\end{array}$$

$$\begin{array}{ccc}
(P_w)_M & \hookrightarrow & M_{s_1} \times \cdots \times M_{s_n} \\
\downarrow (h_{s_1} \times \cdots \times h_{s_n})|(P_w)_M & & \downarrow h_{s_1} \times \cdots \times h_{s_n} \\
(P_w)_{M'} & \hookrightarrow & M'_{s_1} \times \cdots \times M'_{s_n}
\end{array}$$

### Extending structured specifications

**Definition 3.6.4.** A sentence  $\varphi$  is basic iff for all homomorphisms  $h : M \rightarrow M'$ :

$$M \models \varphi \Rightarrow M' \models \varphi$$

We can now add a new building block for structured specifications.

**Definition 3.6.5.** The structured specification  $\text{free}\{SP\}$  has the following semantic

- $\text{Sig}(\text{free}\{SP\}) = \text{Sig}(SP)$
- $\text{Mod}(\text{free}\{SP\}) = \{M \in \text{Mod}(SP) \mid M \text{ initial in } \text{Mod}(SP)\}$

**Example 3.6.6.** The natural numbers are the only model (up to isomorphism) of the specification

$$\text{free}\{\text{sort Nat; ops } 0:\text{Nat}, \text{ suc}:\text{Nat} \rightarrow \text{Nat}\}$$

Similarly: lists, trees, etc.

**Definition 3.6.7.** A definit Horn-clause is a FOL-sentence of the Form

$$\forall x_1 : s_1, \dots, x_n : s : n. (\varphi_1 \wedge \cdots \wedge \varphi_n) \rightarrow \varphi_0$$

with  $\varphi_i$  atomic formulae.

**Theorem 3.6.8.** If  $\Gamma$  consists of definit Horn-clauses, then  $\langle \Sigma, \Gamma \rangle$  has an initial model, i.e.  $\text{Mod}(\text{free}\{\langle \Sigma, \Gamma \rangle\}) \neq \emptyset$ .

*Proof.* Let  $(T_\Sigma)_s$  be the set of terms of sort  $s$ . Define an equivalence relation on  $(T_\Sigma)_s$  by

$$t_1 \sim t_2 \text{ iff } \Gamma \models (t_1 = t_2).$$

Let  $M_s = (T_\Sigma)_{s/\sim}$ , then

$$(f_{w,s})_M([t_1]_\sim, \dots, [t_n]_\sim) := f_{w,s}(t_1, \dots, t_n)$$

This is well defined, because  $\Gamma \models (t_i = t'_i)$  implies  $\Gamma \models f_{w,s}(t_1, \dots, t_n) = f_{w,s}(t'_1, \dots, t'_n)$

$$(p_{w,s})_M := \{([t_1]_\sim, \dots, [t_n]_\sim) \mid \Gamma \models p_{w,s}(t_1, \dots, t_n)\}$$

This is well defined, because  $\Gamma \models (t_i = t'_i)$  implies  $\Gamma \models p_{w,s}(t_1, \dots, t_n) \leftrightarrow p_{w,s}(t'_1, \dots, t'_n)$ . This completes the definition of  $M$ . We have yet to show that  $M$  is initial.

Let  $M' \models \Gamma$ . Define  $h : M \rightarrow M'$  by

$$h([t]_\sim) := M'(t)$$

This is well defined, since  $[t]_\sim = [t']_\sim \Rightarrow t \sim t' \Rightarrow M'(t) = M'(t')$  □

To proof the homomorphic properties we first need some ne definitions.

**Definition 3.6.9.** A morphism  $f : A \rightarrow B$  is an epimorphism, if for any two  $r, s : B \rightarrow C$ ,  $r \circ f = s \circ f \Rightarrow r = s$

**Theorem 3.6.10.** In **SET** epimorphisms are exactly the surjective functions.

*Proof.* Let  $f : A \rightarrow B$  be surjective and  $r \circ f = s \circ f$ , then for any  $b \in B$  by surjectivity there is an  $a \in A$  with  $f(a) = b$ . Then  $r(f(a)) = s(f(a))$ , hence  $r(b) = s(b)$ . Since  $b \in B$  is arbitrary  $r = s$ . Hence  $f$  is an epimorphism. Conversely; assume that  $f : A \rightarrow B$  is *not* surjective. That is, there is  $b_0 \in B$ , such that  $b_0 \notin f[A]$ <sup>1</sup>. Now define  $r, s : B \rightarrow \{T, F\}$  as:

$$r(b) = T$$

$$s(b) = \begin{cases} F & , \text{if } b = b_0 \\ T & , \text{otherwise} \end{cases}$$

Then  $r(f(a)) = s(f(a))$  for all  $a \in A$ . Hence  $r \circ f = s \circ f$ , but  $r \neq s$ . Thus  $f$  is not an epimorphism. □

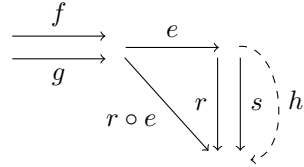
Dual notion: monomorphism. In **Set**: injective functions.

---

<sup>1</sup>Image of  $f$  over  $A$

**Definition 3.6.11.** If  $\begin{matrix} \xrightarrow{f} \\ \xrightarrow{g} \end{matrix} \xrightarrow{e}$  is a coequalizer, then  $e$  is called a regular epimorphism

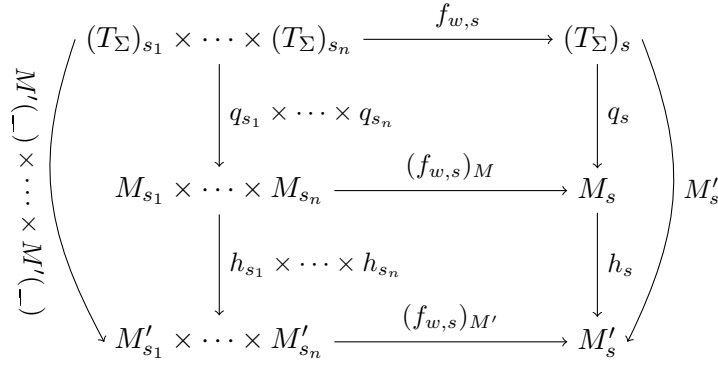
**Proposition 3.6.12.** Any regular epimorphism is an epimorphism.



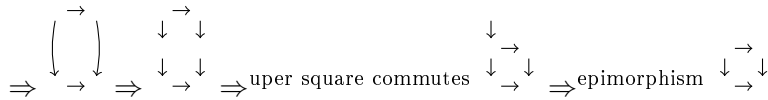
*Proof.*

since  $r \circ e$  equalizes  $f$  and  $g$ , there is a unique  $h$  with  $r \circ e = h \circ e$ . Hence  $h = r = s$   $\square$

*Proof of Theorem 3.6.8 (ctd.)* Homomorphic properties:



$$M'(f_{w,s}(t_1, \dots, t_n)) = (f_{w,s})_{M'}(M'(t_1), \dots, M'(t_n))$$



Predicates analogous.  $\square$

**Proposition 3.6.13** (satisfaction lemma).

- (a)  $M(t) = [t]$
- (b)  $M \models t_1 = t_2$  iff  $\Gamma \models t_1 = t_2$
- (c)  $M \models p_w(t_1, \dots, t_n)$  iff  $\Gamma \models p_w(t_1, \dots, t_n)$

*Proof.*

- (a) easy induction
- (b)  $M \models t_1 = t_2$  iff  $M(t_1) = M(t_2)$  iff (by 1.)  $[t_1] = [t_2]$  iff  $t_1 \sim t_2$  iff  $\Gamma \models t_1 = t_2$

(c) similar

□

**Proposition 3.6.14.**  $M \models \Gamma$

*Proof.*

Let  $\forall x_1 : s_1, \dots, x_n : s_n. \phi_1 \wedge \dots \wedge \phi_n \Rightarrow \phi_0 \in T$

Let  $M'$  be an expansion of  $M$  interpreting  $x_1 \dots x_n$ , i.e.  $M' \in \text{Mod}(\Sigma')$  with  $\Sigma' = \Sigma + \{x_1 : s_1, \dots, x_m : s_m\}$

Now, for any  $\Sigma'$ -term  $t$ ,  $M'(t) = M(\bar{t})$  with  $\bar{t} = t[t_1/x_1, \dots, t_n/x_n]$  with  $t_i \in M'(x_i)$

Assume that:  $M' \models \phi_1 \wedge \dots \wedge \phi_n$

hence  $M \models \bar{\phi}_1 \wedge \dots \wedge \bar{\phi}_n$

and by the satisfaction lemma  $\Gamma \models \bar{\phi}_1 \wedge \dots \wedge \bar{\phi}_n$

Since  $\models$  is closed under substitution and modus ponens  $\Gamma \models \bar{\phi}_0$

Hence  $M' \models \phi_0$

Hence  $M \models \forall x_1 : s_1, \dots, x_n : s_n. \phi_1 \wedge \dots \wedge \phi_n = \phi_0$

□

Altogether we have shown:

(a)  $M \in \text{Mod}(\langle \Sigma, \Gamma \rangle)$

(b) for any  $M' \in \text{Mod}(\langle \Sigma, \Gamma \rangle)$  there exists a unique homomorphism  $h : M \rightarrow M'$

Hence  $M$  is initial in  $\text{Mod}(\langle \Sigma, \Gamma \rangle)$  and thus  $M \in \text{Mod}(\text{free}\{\langle \Sigma, \Gamma \rangle\})$

### 3.7 Adjoint functors

“Adjoint functors arise everywhere” (Sunder MacLane)

**Example 3.7.1.** *Given a set  $A$ , the set of words  $A^*$  forms a monoid with unit  $\lambda^1$  and with string concatenation as monoid multiplication*

```

1 Spec monoid =
2     sort      : s
3     op        : e:s
4     op        : _ * _ : s x s -> s
5     forall    x,y,z :s
6     . x * e = x
7     . e * x = x
8     . (x * y) * z = x*(y*z)
9 end

```

<sup>1</sup>the empty word

**Proposition 3.7.2.** *Let  $i : A \rightarrow A^*$  be the injection taking any letter  $a \in A$  to the single letter word  $a \in A^*$ . Given a monoid  $M = (M_s, e_m, *_m)$  and a function  $f : A \rightarrow M_s$ , there is a unique monoid homomorphism  $f^\# : A^* \rightarrow M$  with  $f^\# \circ i = f$*

$$\begin{array}{ccc} A & \xrightarrow{i} & A^* \\ f \searrow & & \swarrow f^\# \\ & & M \end{array}$$

*Proof.* Define  $f^\#(\lambda) = e_M$  and  $f^\#(a_1, \dots, a_n) = f(a_1) * \dots * f(a_n)$ . These two conditions are equivalent to  $f^\#$  being a homomorphism with  $f^\# \circ i = f$ , hence  $f^\#$  is the unique such homomorphism.  $\square$

More formally: consider the functor  $U : \mathbf{Monoid} \rightarrow \mathbf{Set}$  taking any monoid  $M$  to its carrier set

$$\begin{array}{ccc} A & \xrightarrow{i} & UA^* \\ f \searrow & & \swarrow U(f^\#) \\ & & UM \end{array} \quad \begin{array}{ccc} & & A^* \\ & & \swarrow f^\# \\ & & M \end{array}$$

This property holds, because  $A^*$  satisfies

- no junk
- no confusion  $A^* \models t_1 \times t_1$  iff  $\mathbf{Monoid} \models t_1 = t_1$

**Definition 3.7.3.** *Given a functor  $U : \mathbf{C} \rightarrow \mathbf{D}$  and an object  $x \in |\mathbf{D}|$  a morphism  $\eta : X \rightarrow UC$  (for some  $C \in |\mathbf{C}|$ ) is called U-universal and  $C$  is called U-free over  $X$ , if for any morphism  $f : x \rightarrow UC'$  there is a unique  $f^\# : C \rightarrow C'$  with  $U(f^\#) \circ \eta = f$*

$$\begin{array}{ccc} X & \xrightarrow{\eta} & UC \\ f \searrow & & \swarrow U(f^\#) \\ & & UC' \end{array} \quad \begin{array}{ccc} & & C \\ & & \swarrow f^\# \\ & & C' \end{array}$$

**Proposition 3.7.4.** *Given a functor  $U : \mathbf{C} \rightarrow \mathbf{D}$ , if for any object  $x \in |\mathbf{D}|$  there is a universal arrow  $\eta_X : x \rightarrow U(C_x)$  these lead to a functor  $F : \mathbf{D} \rightarrow \mathbf{C}$  defined by*

$$\begin{array}{ccc}
X & \longrightarrow & C_X \\
c \downarrow & & \downarrow (\eta_y \circ f)^\# \\
Y & \longrightarrow & C_Y \\
\eta_y \downarrow & & \\
UC_Y & & 
\end{array}$$

In this case  $F$  is called left adjoint to  $U$

**Example 3.7.5.**

- **Monoid**  $\xrightarrow{U}$  **Set**  $FA = A^*$
- **TransRel**  $\xrightarrow{U}$  **BinRel**  $FR = \text{transitive closure of } R$
- **AcyclicGraph**  $\xrightarrow{U}$  **Graph**  $FG = \text{the graph of strongly connected components of } G$
- **Sign<sup>FOL</sup>**  $\xrightarrow{U}$  **Sign<sup>PROP</sup>**  $U$  forgets everything except for the 0-ary predicates.  $F$  is the inclusion.
- **PoSet**  $\xrightarrow{U}$  **Prost**  $F(x, \leq) = (x/\sim, \leq)$   
 $x \sim y$  iff  $x \leq y$  and  $y \leq x$
- **Beh**  $\xrightarrow{U}$  **Aut**  $FA = \text{external behaviour.}$
- **Vect**  $\xrightarrow{U}$  **Set**  $FX = \text{Vector space with base } X.$
- $\mathbb{Z} \xrightarrow{U} \mathbb{R}$   $Fr = [r]$

Typically left adjoints are creation processes.

**Exercise 54** (Monomorphisms in **Set**)

Show that a morphism in **Set** is a monomorphism iff it is injective.

**Exercise 55** (Uniqueness of left adjoints)

Show that left adjoints are unique up to isomorphism.

That is, given  $U : \mathbf{C} \rightarrow \mathbf{D}$ , if  $F_1 : \mathbf{D} \rightarrow \mathbf{C}$  and  $F_2 : \mathbf{D} \rightarrow \mathbf{C}$  are left adjoints of  $U$ , then there is a natural transformation  $\tau : F_1 \rightarrow F_2$  that is an isomorphism in the category  $[\mathbf{D}, \mathbf{C}]$  of functors from  $\mathbf{D}$  to  $\mathbf{C}$ .

**Exercise 56** (Free specifications)

Write a free specification that, given a graph, specifies the graph of all paths.

**Definition 3.7.6.** A category with products  $\mathbf{A}$  is called cartesian closed, if for any  $A \in |\mathbf{A}|$  the functor  $\_ \times A : \mathbf{A} \rightarrow \mathbf{A}$  has a right adjoint (written  $\_{}^A : \mathbf{A} \rightarrow \mathbf{A}$ )

Given a  $\Sigma_2$  specification  $SP$  and a signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  then  $\text{free}\{\{\}SP, \sigma\}$  is a specification with

$\text{Sig}(\text{free}\{\{\Sigma_2, \sigma\} = \Sigma_2$   
 $\text{Mod}(\text{free}\{\{\Sigma_2, \sigma\} = \{M \in \text{Mod}(SP) \mid M \text{ is Mod}(\sigma)\text{-free over } M|_\sigma \text{ with } \eta = \text{id} \}$

**Example 3.7.7.**

```

1  spec Prost
2    sort s
3    pred __ <= __ : s x s
4      forall x,y,z :s
5        . x <= x
6        . x <=y /\ y <= z => x <= z
7  end
8
9  spec POSET =
10  Prost then free{
11    sort t
12    op i : s -> t
13    pred __ <= __ t x t

```





# Chapter 4

## Outlook

### 4.1 Modal logic

$\mathcal{ALC} \cong$  multimodal logic  $K$

$\mathcal{ALC}$	$K$	
concept $C$	formula $\varphi$	
atomic concept $A$	propositional variable $p$	
$C \sqcap D$	$\varphi \wedge \psi$	
$C \sqcup D$	$\varphi \vee \psi$	
$\neg C$	$\neg \varphi$	
$\exists R.C$	$\diamond_R \varphi$	
$\forall R.C$	$\square_R \varphi$	
$(\Delta, (C^I)_{C \in \mathbf{C}}, (R^I)_{R \in \mathbf{R}})$	Kripke model with set of worlds $(\Delta)$ , accessability relations $(R^I)$ and inter- pretation of propositional variables $C^I$	
$\square_{(R)} \varphi$	$\square \varphi \rightarrow \varphi$	...
necessarily $\varphi$	✓	
always $\varphi$	✓	
$\varphi$ ought to be	–	
Agent $R$ knows $\varphi$	✓	
Agent $R$ believes $\varphi$	–	
After program $R$ terminates $\varphi$ holds	– ( $\square_R \perp \cong R$ doesn't terminate)	

#### 4.1.1 Correspondence theory

**Proposition 4.1.1.** *A Kripke frame satisfies  $\square p \rightarrow p$  (actually meaning:  $\forall p. \square p \rightarrow p$ ) iff the accessability relation is reflexive.*

**Definition 4.1.2** (global satisfaction).  $\Gamma \models^g$  iff for all Kripke models  $M$ :  
(for all worlds  $w$  in  $M$ ,  $M, w \models \Gamma$ )  $\Rightarrow$  (for all worlds  $w$  in  $M$ ,  $M, w \models \varphi$ )

**Definition 4.1.3** (local satisfaction).  $\Gamma \models^l$  iff for all worlds  $w$  in  $M$ :  
If  $M, w \models \Gamma$  then  $M, w \models \varphi$

modal logic	formula schema	property
T	$\Box p \rightarrow p$	reflexive
B	$p \rightarrow \Box \Diamond p$	symmetric
D	$\Box p \rightarrow \Diamond p$	serial
4	$\Box p \rightarrow \Box \Box p$	transitive (positive introspection)
5	$\neg \Box p \rightarrow \Box \neg \Box p$	euclidian (negative introspection)

Table 4.1: Properties of different modal logics

## 4.2 Coalgebraic logic

$T : \mathbf{Set} \rightarrow \mathbf{Set}$  coalgebra

$$\begin{array}{ccc}
 X & \xrightarrow{f} & TX \\
 \downarrow h & & \downarrow Th \\
 Y & \longrightarrow & TY
 \end{array}$$

## 4.3 Higher-order logic

- Real numbers
- Inductive definitions
- Theorem provers: Isabelle, PVS, HOL
- Many codings (comorphisms) of logics in HOL

**Definition 4.3.1** (Syntax).

*Types:*

$$\tau ::= s \mid 1 \mid \tau_1 \times \tau_2 \mid \tau_1 \rightarrow \tau_2 \mid \mathit{Bool}$$

*Terms/formulae:*

$$t ::= c \mid () \mid (t_1, t_2) \mid \lambda x : \tau. t \mid t_1 t_2 \mid t_1 = t_2$$

**Definition 4.3.2** (Semantics).

Standard model: *Interpretation of basic types.  $\tau_1 \rightarrow \tau_2$  is interpreted as the function space.*

Henkin model:  *$\tau_1 \rightarrow \tau_2$  is interpreted as a subset of the function space.*

Henkin models admit a complete calculus (standard semantics does not - see Gödel)

**Example 4.3.3** (Tarski’s quantifier elimination for real numbers).

$$\begin{aligned} & \exists r, s. r + 1 \leq s \wedge s \leq 3 \\ & \exists s. s \leq 3 \\ & \top \end{aligned}$$

Natural numbers can be used for coding. In particular formulae and provability. This can be used to form a sentence: “I am not provable”.

## 4.4 Substructural logics

In entailment systems, we had

$$\Gamma \vdash \varphi \text{ and } \Gamma \subseteq \Gamma' \Rightarrow \Gamma' \vdash \varphi.$$

In non-monotone logics this is not the case.

### 4.4.1 Linear logic

Multiset of premises, every premise can only be used once.

### 4.4.2 Paraconsistent logic

Weak negation: Not necessarily  $\{\varphi, \neg\varphi\} \vdash \perp$   
Deductive databases.

- subsorted FOL, subsorted HOL
- partial functions
- polymorphism, type constructors ( $\Rightarrow$  HasCASL)

## 4.5 Institutional model theory

Gödel completeness can be extended to institutions.

### 4.5.1 Logic programming

In arbitrary institutions. Fix an institution  $I = (\mathbf{Sign}, \text{Sen}, \text{Mod}, \models)$

**Definition 4.5.1.** A  $\Sigma$ -sentence  $\varphi$  is basic, if there is a  $\Sigma$ -model  $M_\varphi$ , s.t. for any  $\Sigma$ -model  $M$ :

$$M \models \varphi \text{ iff } \exists h : M_\varphi \rightarrow M$$

**Example 4.5.2.** In FOL the atomic formulae are basic.

**Definition 4.5.3.**  $M_{p(t_1, \dots, t_n)}$  is the term algebra with  $p$  interpreted as  $\{(t_1, \dots, t_n)\}$

**Definition 4.5.4** (Existential quantification). *Given a signature morphism  $\sigma : \Sigma \rightarrow \Sigma'$  and a  $\Sigma'$ -sentence  $\varphi$ , define  $\exists\sigma.\varphi$  to be a  $\Sigma$ -sentence with*

$$M \models \exists\sigma.\varphi \text{ iff there is a } \Sigma'\text{-model } M' \text{ with } M'|_{\sigma} = M \text{ and } M' \models \varphi$$

**Definition 4.5.5.** *A signature morphism  $\sigma : \Sigma \rightarrow \Sigma'$  is quasi-representable, if for any model morphism  $h : M'|_{\sigma} \rightarrow N$  there is a unique  $\sigma$ -expansion  $h' : M' \rightarrow N'$ , i.e. with  $h'|_{\sigma} = h$  (and hence  $N'|_{\sigma} = N$ ).*

**Example 4.5.6.** *In FOL,  $\sigma$  is quasi-representable, if it adds new constants.*

*Proof.* Given  $h : M'|_{\sigma} \rightarrow N$  define  $N'$  like  $N$  and  $c_{N'} := h(c_{M'})$ .  $\square$

**Definition 4.5.7.** *A query is a sentence  $\exists\sigma.\varphi$  with  $\sigma$  quasi-representable and  $\varphi$  basic.*

**Theorem 4.5.8** (First Herbrand theorem). *Consider a theory  $T = \langle \Sigma, \Gamma \rangle$  with an initial model  $0_T$  (called Herbrand model). Then for each query  $\exists\sigma.\varphi$ :*

$$T \models \exists\sigma.\varphi \text{ iff } 0_T \models \exists\sigma.\varphi$$

## 4.6 Heterogenous specifications

Let **CoIns** be the category of institutions and institution comorphisms.

**Definition 4.6.1.** *A heterogenous logical environment is a diagram  $D : I \rightarrow \mathbf{CoIns}$*

**Definition 4.6.2.** *Given a heterogenous logical environment  $D : I \rightarrow \mathbf{CoIns}$ , define the Grothendieck institution  $D^\#$  as follows:*

- *Signatures are pairs  $(i, \Sigma)$  with  $i \in I$  and  $\Sigma \in |\mathbf{Sign}|^{D(i)}$*

$$\begin{array}{c} (i, \Sigma) \\ \downarrow (d, \sigma) \text{ with } \sigma : \phi^{D(d)}(\Sigma) \rightarrow \Sigma' \\ (j, \Sigma') \end{array}$$

- *$(i, \Sigma)$ -sentences are  $\Sigma$ -sentences in  $D(i)$*
- *$(i, \Sigma)$ -models are  $\Sigma$ -models in  $D(i)$*
- *$(i, \Sigma)$ -satisfaction is  $\Sigma$ -satisfaction in  $D(i)$*

$$\begin{array}{ccc} \text{Sen}^{D(i)}(\Sigma) & \xrightarrow{\alpha_\Sigma^{D(d)}} & \text{Sen}^{D(j)}(\phi^{D(d)}(\Sigma)) \xrightarrow{\text{Sen}(\sigma)} \text{Sen}^{D(j)}(\Sigma') \\ \text{Mod}^{D(i)}(\Sigma) & \xleftarrow{\beta_\Sigma^{D(d)}} & \text{Mod}^{D(j)}(\phi^{D(d)}(\Sigma)) \xleftarrow{|\sigma} \text{Mod}^{D(j)}(\Sigma') \end{array}$$

## Chapter 5

# Have fun

**Exercise 57** (Christmas bonus problem: existence of Santa Clause I)

Explain what is wrong with the following proof of the existence of Santa Clause.

Recall the  $\exists$ -introduction rule:

$$\frac{\varphi(t)}{\exists x : s.\varphi(x)}$$

*Theorem.* Santa Clause exists.

*Proof.* Assume to the contrary that Santa Clause does not exist. By  $\exists$ -introduction, there exists something that does not exist. This is a contradiction. Hence, the assumption that Santa Clause does not exist must be wrong. Thus, Santa Clause exists.  $\square$

**Exercise 58** (Christmas bonus problem: existence of Santa Clause II)

So, the existence proof in the last exercise was flawed. But I have another proof. What about this one?

Let  $c$  be the set  $\{x \mid \text{if } x \in x, \text{ then Santa Clause exists}\}$ .

Now, assume that  $c \in c$ . We know that  $c \in c$  iff “if  $c \in c$ , then Santa Clause exists”. Therefore, we may assert that “if  $c \in c$ , then Santa Clause exists”. Thus by modus ponens, Santa Clause exists.

However, I have just proven that “if  $c \in c$ , then Santa Clause exists”. Thus,  $c \in c$ , and therefore Santa Clause exists.  $\square$

**Exercise 59** (Christmas bonus problem: all reindeers have the same color)

So I could not convince you that Santa Clause exists. Can I convince you that all reindeers have the same color? The following proof should assure you of that claim, shouldn't it?

*Theorem.* Any number of reindeers have the same color.

*Proof.* By induction.

Basis: one reindeer has the same color (obviously!).

Inductive step: suppose that any collection of  $n$  reindeers has the same color.

We need to show that  $n + 1$  reindeers have the same color, too. By induction hypothesis, the first  $n$  reindeers have the same color. Take out the last reindeer of these and replace it with the  $n + 1$ st. Again by induction hypothesis, these have the same color. Hence, all  $n + 1$  reindeers have the same color.  $\square$

# Bibliography

- [1] J. Barwise and J. Etchemendy. *Language, proof and logic*. CSLI publications, 2002.
- [2] R. Diaconescu. *Institution-Independent Model Theory*. Birkhäuser, 2008.
- [3] Till Mossakowski. Heterogeneous specification and the heterogeneous tool set. Technical report, Universitaet Bremen, 2005. Habilitation thesis.
- [4] Till Mossakowski, Christian Maeder, and Klaus Lüttich. Hets user guide. Technical report, Department of Computer Science; Universität Bremen; Bibliothekstr. 1, 28359 Bremen; <http://www.informatik.uni-bremen.de/>, 2006.
- [5] Till Mossakowski, Christian Maeder, and Klaus Lüttich. The Heterogeneous Tool Set. In Orna Grumberg and Michael Huth, editors, *TACAS 2007*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522. Springer-Verlag Heidelberg, 2007.