

# On Teaching Logic and Algebraic Specification

Till Mossakowski

Courses on algebraic specification and logic have been important cornerstones of teaching theoretical computer science for many years. Moreover, algebraic specification and logic are applied in areas like software specification and verification, but also in ontologies and weak artificial intelligence<sup>1</sup>, and other areas. During my studies, I myself was greatly influenced by courses on algebraic specification and logic. The logic courses mainly provided a very abstract and dry introduction to the formalities of logic — the motivation for logic needed to have arisen independently of the course. By contrast, Hans-Jörg Kreowski always has carefully motivated his courses on algebraic specification (and other subjects), has brought spirit into concepts by using a graphic and descriptive style of presentation, and activated students by insisting on letting them answer questions, discuss points and solve exercises, with room for developing own ideas (especially within so-called student projects, a specialty of Bremen university). This teaching greatly influenced my choice of research subject.

Dear Hans-Jörg, I wish you all the best for your 60th birthday, and please continue your mixture of brilliant research and excellent teaching even though facing the fact that our university system by far does not encourage and support the latter to the degree actually needed,<sup>2</sup> and also students often are not used to an activating teaching style.

In this work, I will report on some research and some teaching I have done in the context of the Common Algebraic Specification Language (CASL [3,4]). CASL is a common language for algebraic specification that has been initiated by the IFIP working group 1.3 “Foundations of systems specification” (see also the report [1]), which was founded and initially lead by Hans-Jörg Kreowski.

## 1 First-Order Logic

Basically, I regularly teach a course about logic that is quite popular (attended by roughly 100 students) and a course on more specialised subjects usually attended only by smaller groups of students.

### 1.1 Language, Proof and Logic

For teaching first-order logic, I use the book “Language, proof and logic” [2], abbreviated LPL. The most striking feature of LPL is the use of software tools

---

<sup>1</sup>Here, *weak* AI is used for systems that solve tasks in specialised domains using heuristics or learning, as opposed to *strong* AI, which aims at passing the Turing test.

<sup>2</sup>As a curiosity: I tried to buy a book about university didactic in the university’s book shop — they had no such book directly available, only books about school didactic.

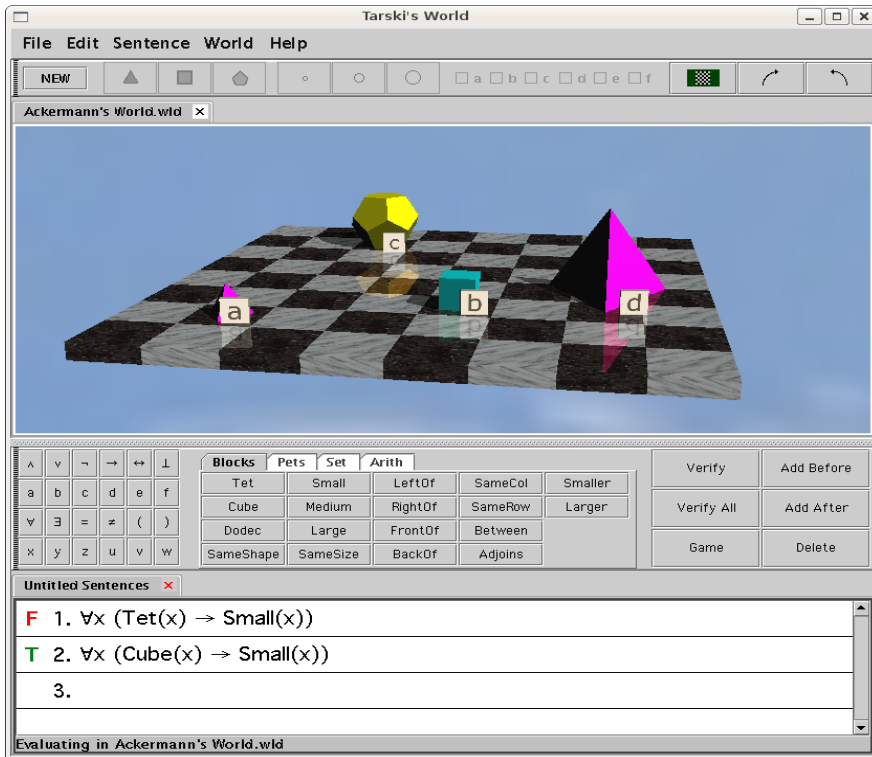


Fig. 1: Evaluating first-order sentences with the program *Tarski's world*.

supporting the students with their own exercises and experiments in logic. This goes as far that a server in Stanford can automatically evaluate some of the students' exercises and give detailed feedback, such that students can revise their solutions. This allows a far better activation of students than with lectures alone — in an ex-cathedra lecture with 100 students, only a small portion of them can actually participate.

However, the usefulness of the software tools should also not be overestimated: it is still very important to have handwritten exercises that are corrected by the teacher, as well as explanations of the students and discussions within the lecture.

In my view, the most important insight of LPL is the following: the notion of first-order structure (or model) is an advanced topic!<sup>1</sup> (The same holds for notion of algebra used in algebraic specification.) Instead, LPL largely uses a

<sup>1</sup>It is only treated in part III of the book. Part I is about propositional logic, part II about first-order logic, and part III about advanced topics.

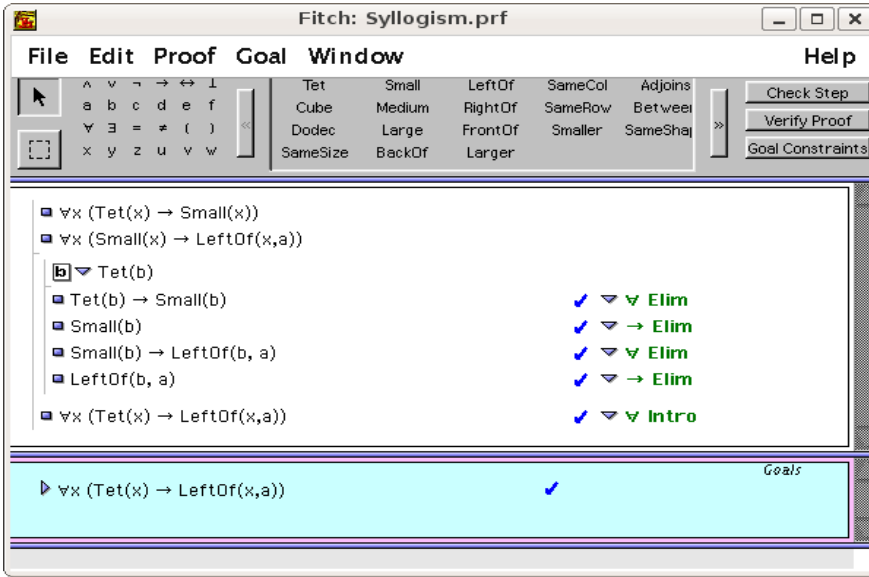


Fig. 2: Sample proof with the program *Fitch*.

fixed interpretation of first-order logic in a blocks world (see Fig. 1 showing a screenshot of the program *Tarski’s world*).

Of course, with using a fixed domain of interpretation (carrier set) and fixed interpretation of predicates, one loses much of the “loose specification” approach used in both algebraic specification and logic. However, the essential gain over the traditional approach is that a fixed interpretation is much easier to grasp. Indeed, a useful didactic will proceed from the concrete to the abstract (and not vice versa), and the abstractness of the concept of (carrier) set (and of function and relation) is often underestimated — even if illustrated with useful example carrier sets from computer science like lists, strings or trees.<sup>1</sup> Moreover, fixing the carrier set and interpretation of predicates is not as harmful as it looks: in a blocks world, it is still possible to obtain some degree of looseness by using different configurations of the blocks. Students can then inspect the effect of different configurations on the evaluation of sentences, and use a game, a so-called Henkin-Hintikka game, to understand the evaluation in more detail. Some looseness of course is also essential to understand the concept of logical

<sup>1</sup>Let me further illustrate this point with some anecdotes about the concept of function. Vladimiro Sassone told me that he taught a course on recursive functions. After several weeks, he spent one lecture on students’ questions. The first question was: “what is a function?”. Michael Kohlhase regularly poses this question in his oral exams, and in spite of him announcing this question, only about 60% of students know the answer.

consequence — another concept that is surprisingly difficult to grasp for many students. The most difficult part to understand is that logical consequence does not imply the truth of the premises — it also holds in cases where the premises are always false.

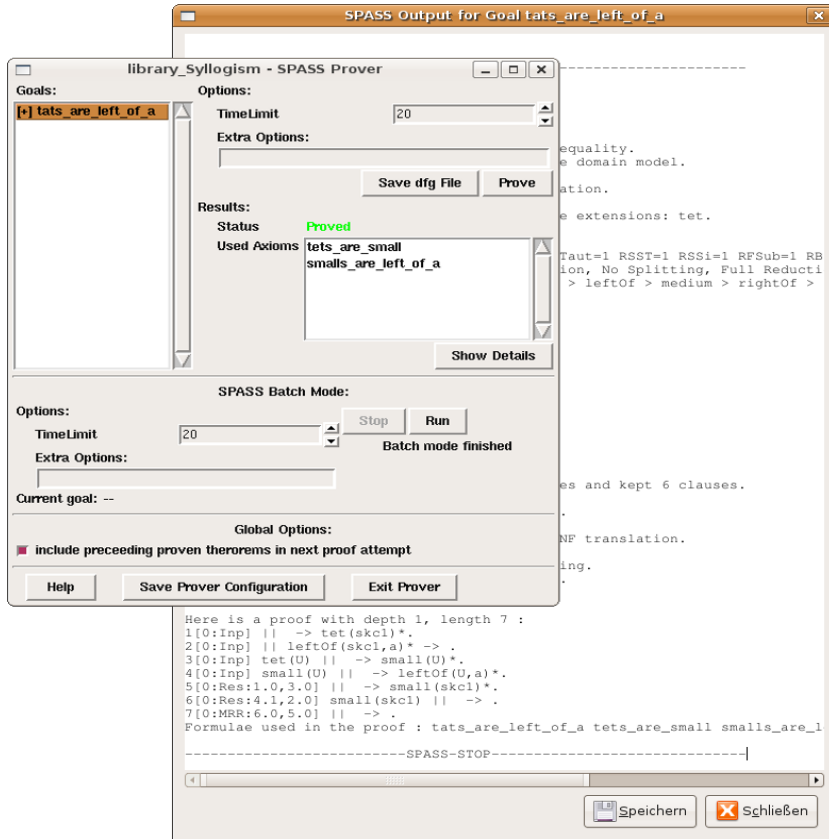


Fig. 3: Sample proof with HETS and SPASS.

Here, the interplay of Tarski's world with *Fitch* greatly helps: *Fitch* is a program that can be used for the construction of a natural deduction proof, in case that a logical consequence actually holds, see Fig. 2. In the other cases, Tarski's world can be used to construct countermodels.

LPL offers a great deal of motivation and explanation of the natural deduction calculus (and *Fitch*) in terms of common natural language arguments. It must be noted though that students more often have difficulties with *Fitch* than with Tarski's world. The reason seems to be again the level of abstraction: while Tarski's world is about a blocks world that is still close to everyday's experience,

Fitch is about proofs that follow certain rules which are quite common in mathematical arguments, but not in everyday's experience. Moreover, students often have difficulties with finding suitable rules to apply in a given situation, or with the development of a proof strategy. Therefore, the development of proof strategies is explicitly discussed in the lecture and supported with numerous exercises. However, I think that this still does not suffice. An interactive dialogue suggesting different strategies or heuristics might help to stimulate more experiments also for those students that do not grasp natural deduction so quickly.

## 1.2 Hets and State-of-the-Art Provers

This also brings me to another point: the relation of Fitch to state-of-the-art automated and interactive theorem provers. Some students are motivated to conduct larger proofs, but Fitch is not suited for this, since it is not possible to prove lemmas and theorems for later re-use. Here, I use CASL and the Heterogeneous Tool Set HETS [8,7], which offers the connection to a selection of resolution provers (SPASS, Vampire) and tableau provers (Isabelle), as well as to SAT solvers (zChaff, minisat) — all tools that are used in current research. However, these tools of course do not offer the special proof rule provided by Fitch that can be used to derive facts that are specific to the blocks world (this rule is called “AnaCon”). Actually, the rule AnaCon can be simulated with a suitable first-order axiomatisation of the blocks world in CASL. Then proofs can be conducted e.g. with the automated resolution prover SPASS [10]. A drawback is that the output format of resolution proofs is still rather cryptic, since the problem is first translated to clause form. A translation from resolution proofs to natural deduction (using tools like Tramp [5] or Metis [6]) could help here, but one should be careful not to provide an automatic tool that completely discourages students to build their own natural deduction proofs.

## 2 Structured Specification

While research in algebraic specification started with the application of methods from universal algebra and equational logic to the specification of abstract data types, later the algebraic nature was found more in the powerful constructs that are used to build larger specifications from smaller ones in a modular way. One such construct is the restriction to so-called initial and free models, a quite central but complex notion in the area of algebraic specification. While teaching this notion, I developed the idea to use propositional logic (instead of equational or first-order logic) to illustrate constructs for structuring specification. The advantage is that the logic is so simple that one can really concentrate on the structuring. Moreover, it is possible to display individual models: they are just rows in a truth table. Using this approach, the following subsections explain logical consequence, conservative extensions, and initial/free specifications. The development will be a bit more technical than above, and also will rely on mathematical notation. However, it will be intensively illustrated with results from HETS.

## 2.1 Logical Consequence

Logical consequence is the central notion of logic (and is also important for algebraic specification): what follows from what? As indicated above, logical consequence is a notion that is difficult to grasp for many students. Hence, with HETS, we provide an easy truth table approach for illustrating this notion.

**Definition 1 (Signature).** A propositional signature  $\Sigma$  is a set (of propositional symbols, or variables).

**Definition 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

**Definition 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.

A  $\Sigma$ -model  $M$  can be extended to

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

using truth tables.

**Definition 4.**  $\phi$  holds in  $M$  (or  $M$  satisfies  $\phi$ ), written  $M \models_\Sigma \phi$  iff

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

**Definition 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  $M \in \text{Mod}(\Sigma)$

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

*Example 6.* An argument in natural language is tested for validity by translating it into propositional logic.

John plays tennis, if it's a sunny weekend day. If John plays tennis, then Mary goes shopping. It is Saturday. It is sunny. Mary goes shopping	$\left. \begin{array}{l} \text{sunny} \wedge \text{weekend} \rightarrow \text{tennis} \\ \text{tennis} \rightarrow \text{shopping} \\ \text{saturday} \\ \text{sunny} \\ \text{saturday} \rightarrow \text{weekend} \\ \text{shopping} \end{array} \right\}$
--	--

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

```

1  logic Propositional
2
3  spec Weekend =
4    props tennis, shop, sunny, sat, we
5    . sunny /\ we => tennis    %(SWT)%
6    . tennis => shop          %(TSh)%
7    . sat                     %(sat)%
8    . sat => we                %(satW)%
9    . sunny                   %(sun)%
10   . shop                    %(shop)% %implied
11 end

```

Listing 1: A simple logical consequence

	sat	shop	sunny	tennis	we	SWT	TSh	sat	satW	sun	shop		
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	sat   shop   sunny   tennis   we    SWT   TSh   sat   satW   sun    shop												
8	=====												
9	o		F		F		F		F		F		F
10	o		F		F		F		F		T		F
11	o		F		F		F		T		F		F
12	o		F		F		F		T		T		F
13	o		F		F		T		F		F		F
14	o		F		F		T		F		T		F
15	o		F		F		T		F		T		F
16	o		F		F		T		T		T		F
17	o		F		T		F		F		F		T
18	o		F		T		F		T		T		T
19	o		F		T		F		T		F		T
20	o		F		T		F		T		F		T
21	o		F		T		T		F		T		T
22	o		F		T		T		F		T		T
23	o		F		T		T		F		T		T
24	o		F		T		T		F		T		T
25	o		T		F		F		F		F		F
26	o		T		F		F		T		T		F
27	o		T		F		F		T		F		F
28	o		T		F		F		T		T		F
29	o		T		F		T		F		F		F
30	o		T		F		T		F		T		F
31	o		T		F		T		F		T		F
32	o		T		F		T		T		T		F
33	o		T		T		F		F		F		T
34	o		T		T		F		T		T		T
35	o		T		T		F		T		F		T
36	o		T		T		F		T		T		T
37	o		T		T		T		T		F		T
38	o		T		T		T		F		T		T
39	o		T		T		T		T		F		T
40	M+		T		T		T		T		T		T

Listing 2: Truth table for the logical consequence from Listing 1

Note that the formalisation contains an axiom  $\text{saturday} \rightarrow \text{weekend}$  not present in the informal version. This axiom represents implicit background knowledge. The HETS input syntax for this example is shown in Listing 1.

With HETS, we can now construct the following truth table as shown in Listing 2. The truth table is divided into three parts, using `||`. The first part consists of the signature: all propositional letters are listed. Below the signature, you find all possible models, one per row. The second part consists of the theory (the axioms, also playing the role of *premises* of the argument): for each axiom, its truth value is listed. Only rows containing `T` for every axiom are models of the theory (indicated by an `M`). Finally, the third part contains the proof goal, or *conclusion* of the argument. The conclusion needs to be true for each row that is a model.

A simple non-example of a logical consequence (actually, we omitted the fact that `saturday` is a weekend day) is shown in Listing 3.

```

1 spec Weekend2 =
2   props tennis, shop, sunny, sat, we
3   . sunny /\ we => tennis    %(SWT)%
4   . tennis => shop          %(TSh)%
5   . sat                     %(sat)%
6   . sunny                   %(sun)%
7   . shop                    %(shop)% %implied
8 end

```

Listing 3: Example of a non-consequence

## 2.2 Conservative Extensions

A theory is *satisfiable*, if it has a model.<sup>1</sup> Satisfiability of theories is quite important for an axiomatic or loose approach to specification: it is easy to introduce unintentional inconsistencies, and an inconsistent (unsatisfiable) specification cannot be realised, hence it does not successfully model an aspect of reality.<sup>2</sup>

Satisfiability of large theories is hard to show. Actually, there are large first-order theories like the SUMO ontology for which satisfiability is an open question — indeed there is a prize set up for proving consistency of SUMO [9]. A modular way to satisfiability is opened up by conservative extensions: in a sense, these transport satisfiability.

<sup>1</sup>In some logics like equational logic, each theory is trivially satisfiable. In these cases, satisfiability should be replaced with satisfiability by a non-trivial model, where the latter is a model that falsifies at least one sentence.

<sup>2</sup>This is different for paraconsistent logics, which however will not be considered here.



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	sat   shop   sunny   tennis   we    SWT   TSh   sat   sun    shop
8	-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
9	o    F   F   F   F   F    T   T   F   F    F
10	o    F   F   F   F   F    T   T   F   F    F
11	o    F   F   F   T   F    T   F   F   F    F
12	o    F   F   F   F   F    T   T   F   F    F
13	o    F   F   T   F   F    T   T   F   T    F
14	o    F   F   T   F   T    F   T   F   T    F
15	o    F   F   T   F   F    T   F   F   F    F
16	o    F   F   T   T   T    T   F   F   T    F
17	o    F   T   F   F   F    T   T   F   F    T
18	o    F   T   F   F   F    F   T   F   F    T
19	o    F   T   F   T   F    T   F   T   F    T
20	o    F   T   F   T   T    T   T   F   F    T
21	o    F   T   T   T   F    F   F   T   F    T
22	o    F   T   T   F   F    F   T   F   T    T
23	o    F   T   T   T   F    T   T   F   T    T
24	o    F   T   T   T   T    T   T   F   T    T
25	o    T   F   F   F   F    T   T   T   F    F
26	o    T   F   F   F   T    F   T   T   F    F
27	o    T   F   F   F   F    T   F   F   T    F
28	o    T   F   F   T   F    T   T   F   T    F
29	M-    T   F   T   F   F    F   F   T   T    F
30	o    T   F   T   F   F    F   T   T   T    F
31	o    T   F   T   T   F    T   F   T   T    F
32	o    T   F   T   T   T    T   F   T   T    F
33	o    T   T   F   F   F    F   F   T   T    T
34	o    T   T   F   F   T    F   T   T   F    T
35	o    T   T   F   T   F    T   F   T   T    T
36	o    T   T   F   T   T    T   T   T   F    T
37	M+    T   T   T   F   F    F   F   T   T    T
38	o    T   T   T   F   F    F   T   T   T    T
39	M+    T   T   T   T   F    T   F   T   T    T
40	M+    T   T   T   T   T    T   T   T   T    T

Listing 4: Truth table for the non-consequence from Listing 3

1	<b>spec</b> Sp =	1	<b>spec</b> Animals =
2	$\Sigma_1$	2	<b>props</b> bird, penguin
3	$\Gamma_1$	3	. penguin => bird
4	<b>then</b>	4	<b>then</b>
5	$\Sigma_\Delta$	5	<b>prop</b> can_fly
6	$\Gamma_\Delta$	6	. penguin => not can_fly
7	<b>end</b>	7	<b>end</b>

Listing 5: Theory extensions in CASL.

```

1 logic Propositional
2
3 spec Animal =
4   props bird, penguin, living
5   . penguin => bird    %(pb)%
6   . bird => living    %(bl)%
7 then %cons
8   prop animal
9   . bird => animal    %(ba)%
10  . animal => living %(al)%
11 end

```

```

1
2 spec Penguin =
3   props bird, penguin
4   . penguin => bird    %(pb)%
5 then
6   prop can_fly
7   . bird => can_fly    %(bc)%
8   . penguin => not can_fly %(pnc)%
9 end

```

Listing 6: Example of a conservative and a non-conservative extension in CASL

To illustrate the concept, consider the specification in Listing 6. Indeed, to formally underpin this, we introduce some notions that will be central for structured specification:

**Definition 7.** *Given two signatures  $\Sigma_1, \Sigma_2$  a signature morphism is a function  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  (note that signatures are sets).*

Sentences can be translated along signature morphisms:

**Definition 8.** *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.*

Models are translated *against* signature morphisms. The intuition is that the translated model  $M|_\sigma$  works as follows: interpret a symbol by first translating it along the signature morphism  $\sigma$  and then look up the interpretation in the original model  $M$ .

**Definition 9.** *A signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  induces a model reduction  $\_|\sigma : \text{Mod}(\Sigma_2) \rightarrow \text{Mod}(\Sigma_1)$ . Given  $M \in \text{Mod}(\Sigma_2)$  i.e.  $M : \Sigma \rightarrow \{T, F\}$ , then  $M|_\sigma \in \text{Mod}(\Sigma_1)$  is defined as  $M|_\sigma(\phi) := M(\sigma(\phi))$  i.e.  $M|_\sigma = M \circ \sigma$*

Sentence and model translation interact well with each other:

**Theorem 10 (Satisfaction condition).** *Given a signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$ ,  $M_2 \in \text{Mod}(\Sigma_2)$  and  $\phi_1 \in \text{Sen}(\Sigma_1)$ , 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.”)

**Definition 11.** *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  $M_2|_{\sigma} \in \text{Mod}(\Sigma_1, \Gamma_1)$*

Extensions (written in CASL with the keyword `then`; cf. Listing 5) always lead to a theory morphism (by definition). The semantics of the CASL specification is the 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.

We are now ready to define conservative extensions:

**Definition 12.** *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  $T_2$ -model, that is, a model*

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

We can now evaluate which of the extensions shown in Listing 6 are conservative. Actually, the first extension is conservative. In the truth table output by HETS (see Listing 7), we can see that each model (marked with an M in the leftmost column) has an expansion (marked with an M in the column right to the middle).

By contrast, the second extension is *not* conservative: the last model fails to have an expansion, see Listing 8.

The central theorem that allows us to transport satisfiability is the following:

**Theorem 13.** *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.*

### 2.3 Initial and Free Specifications

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 somewhat circular statements:

Harry: John tells the truth.

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

```

1 Legend:
2 M = model of the axioms
3 + = OK, has expansion
4 - = not OK, has no expansion, hence conservativity fails
5 o = OK, not a model of the axioms, hence no expansion needed
6
7   || bird | living | penguin || pb | bl ||   || animal ||   ba |   al
8 =====
9 M+ ||   F |   F |   F || T | T || M ||   F ||   T |   T
10  ||   |   |   ||   |   ||   ||   T ||   T |   F
11 -----
12 o ||   F |   F |   T || F | T ||   ||   ||   |   |
13 -----
14 M+ ||   F |   T |   F || T | T || M ||   F ||   T |   T
15  ||   |   |   ||   |   ||   || M ||   T ||   T |   T
16 -----
17 o ||   F |   T |   T || F | T ||   ||   ||   |   |
18 -----
19 o ||   T |   F |   F || T | F ||   ||   ||   |   |
20 -----
21 o ||   T |   F |   T || T | F ||   ||   ||   |   |
22 -----
23 M+ ||   T |   T |   F || T | T ||   ||   F ||   F |   T
24  ||   |   |   ||   |   || M ||   T ||   T |   T
25 -----
26 M+ ||   T |   T |   T || T | T ||   ||   F ||   F |   T
27  ||   |   |   ||   |   || M ||   T ||   T |   T

```

Listing 7: Truth table for a conservative extension from Listing 6

```

1 Legend:
2 M = model of the axioms
3 + = OK, has expansion
4 - = not OK, has no expansion, hence conservativity fails
5 o = OK, not a model of the axioms, hence no expansion needed
6
7   || bird | penguin || pb ||   || can_fly ||   bc |   pnc
8 =====
9 M+ ||   F |   F || T || M ||   F ||   T |   T
10  ||   |   ||   || M ||   T ||   T |   T
11 -----
12 o ||   F |   T || F ||   ||   ||   |   |
13 -----
14 M+ ||   T |   F || T ||   || F ||   F |   T
15  ||   |   ||   || M ||   T ||   T |   T
16 -----
17 M- ||   T |   T || T ||   || F ||   F |   T
18  ||   |   ||   ||   || T ||   T |   F

```

Listing 8: Truth table for a non-conservative extension from Listing 6

```

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

```

Listing 9: A circular set of statements

```

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

```

Listing 10: Truth table for the circular statements from Listing 9

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.

Actually, when calling HETS with the truth table prover, the first goal cannot be proved, see Listing 10.

The other goals 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 in Listing 10). 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 14.** Given a propositional signature  $\Sigma$  and two  $\Sigma$ -models  $M_1$  and  $M_2$ , then  $M_1 \leq M_2$  if  $M_1(p) = T$  implies  $M_2(p) = T$  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, have a look at Listing 11. With the HETS truth table prover, we now get the truth table in Listing 12. That is, Harry, John and Mary all are lying! Actually, 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.

```

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

```

Listing 11: Closed world assumption, specified as a free extension

	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

Listing 12: Truth table for the specification of Listing 11

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, see Listing 13. However, no

greatest model exists in this case, hence the cofree specification is inconsistent, as shown in Listing 14.

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

```

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

```

Listing 13: Closed world assumption, specified as a cofree extension

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

Listing 14: Truth table for the specification of Listing 13

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, see Listing 15.

The semantics is as follows:

$$\text{Mod}(SP_1 \text{ then } \text{free}\{SP_2\}) = \{M \in \text{Mod}(SP_1 \text{ then } SP_2) \mid M \text{ is the least model in } \{M' \in \text{Mod}(SP_1 \text{ then } SP_2) \mid M|_{\sigma} = M'|_{\sigma}\} \}$$

and as a result, we obtain that both Harry and John lie (independently of what Mary concerns!), see Listing 16.

The dual concept is cofreeness with mixed open and closed world semantics, see Listing 17. Also the semantics is obtained by dualising:

```

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

```

Listing 15: Mixed open world and closed world semantics using free

	harry	john	mary	whenjohn	whenharry	free	harry
M+	F	F	F	T	T	T	T
M+	F	F	T	T	T	T	T
o	F	T	F	T	T	F	T
o	F	T	T	T	T	F	T
o	T	F	F	F	T	F	F
o	T	F	T	F	T	F	F
o	T	T	F	T	T	F	F
o	T	T	T	T	F	F	F

Listing 16: Truth table for the specification of Listing 15

```

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

```

Listing 17: Mixed open world and closed world semantics using cofree



$$\text{Mod}(SP_1 \text{ then } \text{cofree}\{SP_2\}) = \{M \in \text{Mod}(SP_1 \text{ then } SP_2) \mid M \text{ is the greatest model in } \{M' \in \text{Mod}(SP_1 \text{ then } SP_2) \mid M|_\sigma = M'|_\sigma\} \}$$

The result in the example is that John tells the truth, and at least either of Harry and Mary as well, see Listing 18.

	harry	john	mary	whenjohn	whenharry	cofree	harrymary
1							
2							
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	M+	F	T	T	T	T	T
7	o	T	F	F	F	T	F
8	o	T	F	T	F	T	F
9	M+	T	T	F	T	T	T
10	o	T	T	T	T	F	F

Listing 18: Truth table for the specification of Listing 17

### 3 Conclusion

The overall picture is as follows: typically, I start with a course on first-order logic as described in Sect. 1, followed by a more special course on structuring and institutions, following Sect. 2. The second course starts with propositional logic, which keeps the examples simple, and then proceeds to description logics (used for ontologies and semantic web) and finally again to first-order logic.

Teaching algebraic specification and logic can really be fun, and there is much room for developing better ideas and tools supporting this. Feedback and improvements are welcome!

### References

1. E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner. *Algebraic Foundations of Systems Specification*. Springer, 1999.
2. J. Barwise and J. Etchemendy. *Language, proof and logic*. CSLI publications, 2002.
3. Michel Bidoit and Peter D. Mosses. *CASL User Manual*, volume 2900 of *LNCS (IFIP Series)*. Springer, 2004. With chapters by T. Mossakowski, D. Sannella, and A. Tarlecki.
4. CoFI (The Common Framework Initiative). *CASL Reference Manual*, volume 2960 of *LNCS (IFIP Series)*. Springer, 2004.
5. Andreas Meier. System description: TRAMP: Transformation of machine-found proofs into ND-proofs at the assertion level. In David A. McAllester, editor, *Automated Deduction - CADE-17, 17th International Conference on Automated Deduction, Pittsburgh, PA, USA, June 17-20, 2000, Proceedings*, volume 1831 of *Lecture Notes in Computer Science*, pages 460–464. Springer, 2000.

6. Jia Meng, Claire Quigley, and Lawrence C. Paulson. Automation for interactive proof: First prototype. *Information and Computation*, 204(10):1575–1596, 2006.
7. T. Mossakowski. Heterogeneous specification and the heterogeneous tool set. Habilitation thesis, University of Bremen, 2005.
8. 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.
9. Adam Pease, Geoff Sutcliffe, Nick Siegel, and Steven Trac. The annual SUMO reasoning prizes at CASC. In Boris Konev, Renate A. Schmidt, and Stephan Schulz, editors, *Proceedings of the First International Workshop on Practical Aspects of Automated Reasoning, Sydney, Australia, August 10-11, 2008*, volume 373 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
10. C. Weidenbach, U. Brahm, T. Hillenbrand, E. Keen, C. Theobalt, and D. Topic. SPASS version 2.0. In Andrei Voronkov, editor, *Automated Deduction – CADE-18*, LNCS 2392, pages 275–279. Springer-Verlag, 2002.



**Dr. Till Mossakowski**

DFKI GmbH Bremen  
Safe & Secure Cognitive Systems  
Enrique-Schmidt-Str. 5  
D-28359 Bremen (Germany)  
Till.Mossakowski@dfki.de  
<http://www.dfki.de/sks/till>

During his studies, Till Mossakowski took several courses in Theoretical Computer Science held by Hans-Jörg Kreowski. Moreover, he was a member of his group from 1993 to 1998. Hans-Jörg also supervised Till's diploma and doctoral theses, and was an examiner of his habilitation thesis.

---