Electronic Communications of the EASST
Volume 26 (2010)

Manipulation of Graphs, Algebras and Pictures

Essays Dedicated to Hans-Jörg Kreowski on the Occasion of His 60th Birthday

On Teaching Logic and Algebraic Specification

Till Mossakowski

19 pages

Guest Editors: Frank Drewes, Annegret Habel, Berthold Hoffmann, Detlef Plump
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

On Teaching Logic and Algebraic Specification

Till Mossakowski

Till.Mossakowski@dfki.de, http://www.dfki.de/sks/till/
German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany

SFB/TR 8 Spatial Cognition, Universität Bremen, Germany

Abstract: We discuss teaching experiences with courses on first-order logic and on algebraic specification, with an emphasis on software tools that can be used by students and that illustrate the meaning of logical notions. In particular, we discuss *Language, Proof and Logic* and the *Heterogeneous Tool Set*. Moreover, we claim that structuring constructs like those of the Common Algebraic Specification Language can be better digested when starting with applying them to propositional logic.

Keywords: First-order logic, algebraic specification, structured specification, teaching

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 not only in areas like software specification and verification, but also in ontologies and weak artificial intelligence[1], 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 was expected 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.

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 [BM04, CoF04]). 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 [AKK99]), 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] 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.

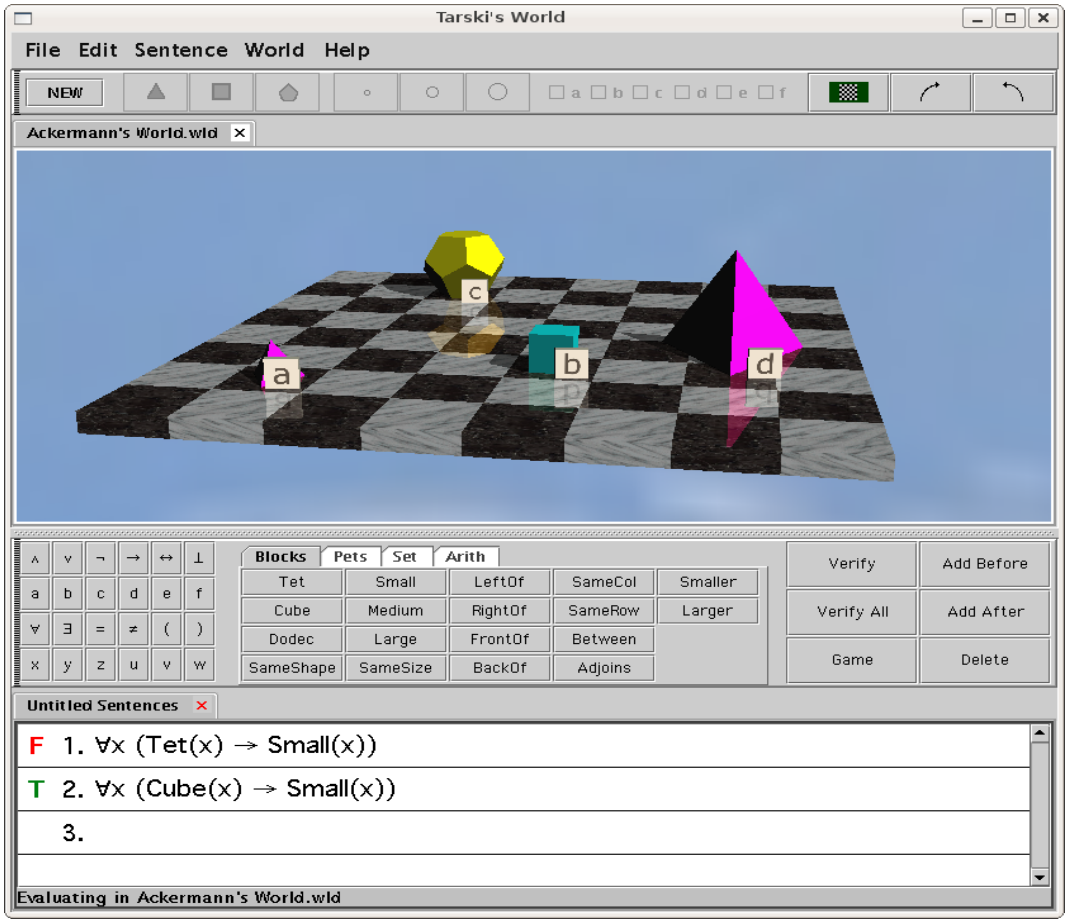

Figure 1: Evaluating first-order sentences with the program *Tarski's world*.

1.1 Language, Proof and Logic

For teaching first-order logic, I use the book "Language, proof and logic" [BE02], abbreviated LPL. The most striking feature of LPL is the use of software tools 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 a lecture using ex-cathedra teaching in front of 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

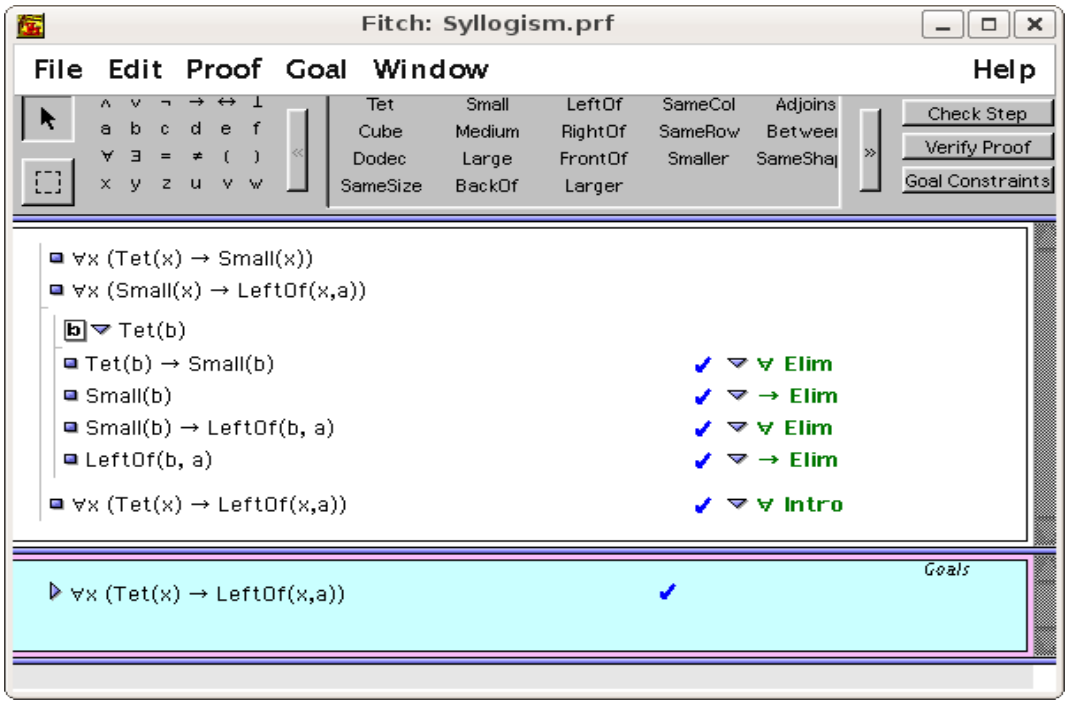

Figure 2: Sample proof with the program *Fitch*.

structure (or model) is an advanced topic![1] (The same holds for the notion of algebra used in algebraic specification.) Instead, LPL largely uses a fixed interpretation of first-order logic in a blocks world (see Figure 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.[2] 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 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.

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 — in the other cases, Tarski's world can be used to construct counter-models. A sample proof with Fitch is shown in Figure 2. It uses a subproof for proving a universally quantified implication. Fitch does not construct such proofs, it only checks them and sometimes provides the resulting formula for a step (if a proof rule and formulas supporting its application have been selected appropriately). Fitch also checks if restrictions set up by the instructor have met, e.g. that the built-in first-order prover must not be used, or that a certain conclusion has to be reached.

The feedback that Fitch gives allows students to try out and play around with proof construction. Moreover, LPL offers a great deal of motivation and explanation of the natural deduction calculus (and Fitch) in terms of common natural language arguments. LPL also discusses strategies to find proofs; e.g. a simple strategy is to look at the main connectives of the premises and try corresponding elimination rules, or try the introduction rule corresponding to the main connective of the conclusion. However, it must be noted that students much 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 experience, Fitch is about proofs that follow certain rules which are quite common in mathematical arguments, but not in everyday 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

[1] First-order structures are only treated in part III of the book (covering advanced topics).

[2] 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.

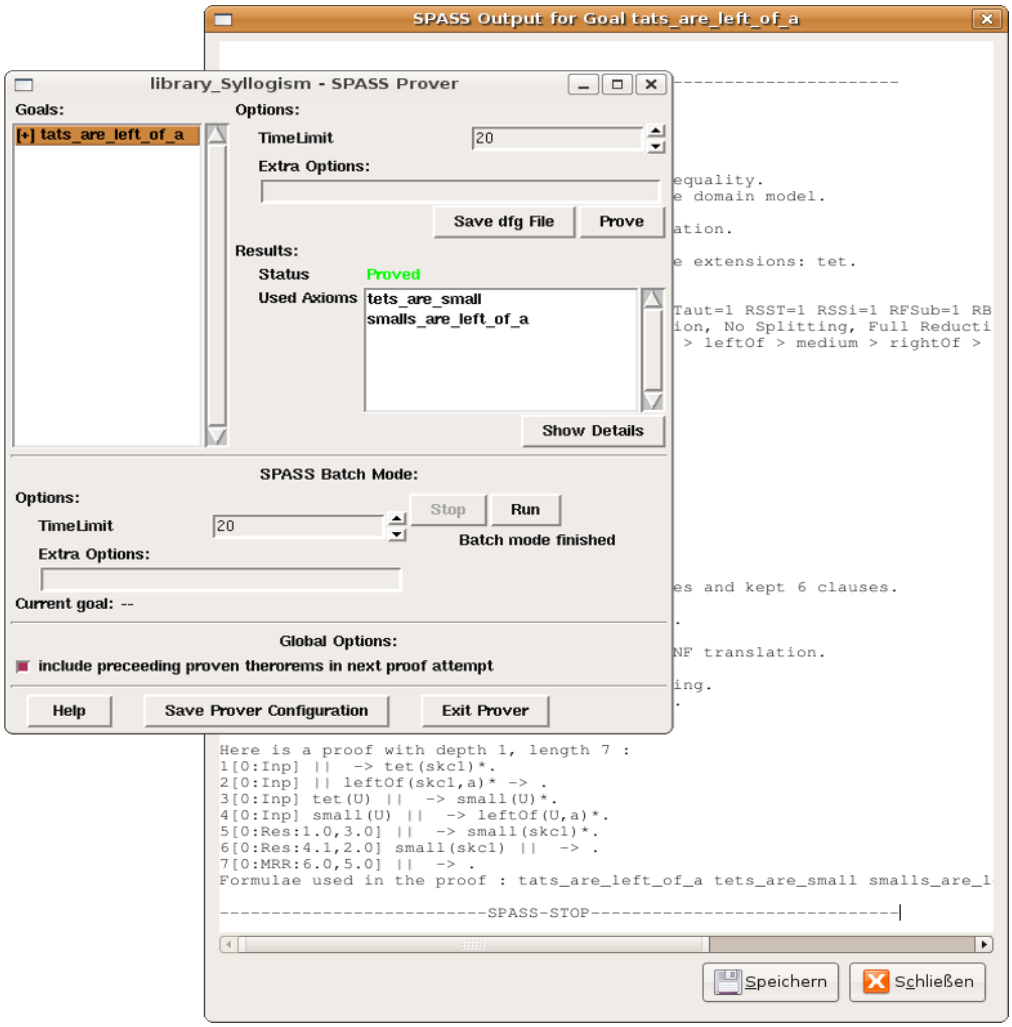

Figure 3: Sample proof with HETS and SPASS.

strategies or heuristics might help to stimulate more experiments also for those students that do not grasp natural deduction so quickly. Probably, also more exercises requiring the judgement of given Fitch proofs would be useful.

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 [MML07, Mos05], 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 is not explained in the book. However, it can be simulated with a suitable first-order axiomatisation of the blocks world. Such an axiomatisation is begun in the LPL book. We have (together with students) developed a much more complete first-order axiomatisation of the blocks world in CASL.[1] Then proofs can be conducted e.g. with the automated resolution prover SPASS [WBH+02], see Fig. 3. 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 [Mei00] or Metis [MQP06]) could help here in the future, 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.

[1] We have not published this axiomatisation, since it would enable students to cheat when solving the exercises in the book. However, the axiomatisation can be mailed to instructors on request.

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* Σ is a set (of propositional symbols, or variables).

Definition 2 (Sentence) Given a propositional signature Σ, a *propositional sentence* over Σ is one produced by the following grammar

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

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

Definition 3 (Model) Given a propositional signature Σ, a Σ-*model* (or Σ-valuation) is a function in $\Sigma \rightarrow \{T,F\}$. $\mathrm{Mod}(\Sigma)$ is the set of all Σ-models.

A Σ-model M can be extended to $M^{\#} : \mathrm{Sen}(\Sigma) \rightarrow \{T,F\}$ using truth tables.

Definition 4 ϕ holds in a Σ-model M (or M *satisfies* ϕ), written $M \models_{\Sigma} \phi$ iff

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

Definition 5 (Logical consequence) Given $\Gamma \subseteq \mathrm{Sen}(\Sigma)$ and $\phi \in \mathrm{Sen}(\Sigma)$, ϕ is a *logical consequence* of Γ (written as $\Gamma \models \phi$), if for all $M \in \mathrm{Mod}(\Sigma)$

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

Example 1 *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*	*sunny* ∧ *weekend* → *tennis* *tennis* → *shopping* *saturday* *sunny* *saturday* → *weekend* ⊢ *shopping*

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

Note that the formalisation contains an axiom `saturday → 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. Note that logical implication → is input as =>, conjunction ∧ as /\ etc. Moreover, each formula is preceded by a dot, and formulas can be labelled with

`%(label)%`. The notation `%implied` marks a formula to be a theorem (that has to be proved to follow from the axioms); all other formulas are axioms.

```
logic Propositional

spec Weekend =
  props tennis, shop, sunny, sat, we
  . sunny /\ we => tennis   %(SWT)%
  . tennis => shop          %(TSh)%
  . sat                     %(sat)%
  . sat => we               %(satW)%
  . sunny                   %(sun)%
  . shop                    %(shop)% %implied
end
```

Listing 1: A simple logical consequence

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.

2.2 Conservative Extensions

A theory is *satisfiable* if it has a model.[1] 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.[2]

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 [PSST08]. A modular way to satisfiability is opened up by conservative extensions: in a sense, these transport satisfiability.

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 6 Given two signatures Σ_1, Σ_2 a *signature morphism* is a function $\sigma : \Sigma_1 \to \Sigma_2$ (note that signatures here are sets).

Sentences can be translated along signature morphisms:

[1] 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.

[2] This is different for paraconsistent logics, which however will not be considered here.

```
Legend:
M = model of the premises
+ = OK, model fulfils conclusion
- = not OK, counterexample for logical consequence
o = OK, premises are not fulfilled, hence conclusion is irrelevant

     ------------ signature -----------   ---------- premises ---------  conclusion
   || sat  | shop  | sunny | tennis  | we || SWT  | TSh  | sat  | satW  | sun  || shop
===++======+======+=======+=========+====++======+======+======+=======+======++======
 o ||   F  |    F |     F |       F |  F ||   T  |   T  |   F  |     T |   F  ||    F
 o ||   F  |    F |     F |       F |  T ||   T  |   T  |   F  |     T |   F  ||    F
 o ||   F  |    F |     F |       T |  F ||   T  |   F  |   F  |     T |   F  ||    F
 o ||   F  |    F |     F |       T |  T ||   T  |   F  |   F  |     T |   F  ||    F
 o ||   F  |    F |     T |       F |  F ||   T  |   T  |   F  |     T |   T  ||    F
 o ||   F  |    F |     T |       F |  T ||   F  |   T  |   F  |     T |   T  ||    F
 o ||   F  |    F |     T |       T |  F ||   T  |   F  |   F  |     T |   T  ||    F
 o ||   F  |    F |     T |       T |  T ||   T  |   F  |   F  |     T |   T  ||    F
 o ||   F  |    T |     F |       F |  F ||   T  |   T  |   F  |     T |   F  ||    T
 o ||   F  |    T |     F |       F |  T ||   T  |   T  |   F  |     T |   F  ||    T
 o ||   F  |    T |     F |       T |  F ||   T  |   T  |   F  |     T |   F  ||    T
 o ||   F  |    T |     F |       T |  T ||   T  |   T  |   F  |     T |   F  ||    T
 o ||   F  |    T |     T |       F |  F ||   T  |   T  |   F  |     T |   T  ||    T
 o ||   F  |    T |     T |       F |  T ||   F  |   T  |   F  |     T |   T  ||    T
 o ||   F  |    T |     T |       T |  F ||   T  |   T  |   F  |     T |   T  ||    T
 o ||   F  |    T |     T |       T |  T ||   T  |   T  |   F  |     T |   T  ||    T
 o ||   T  |    F |     F |       F |  F ||   T  |   T  |   T  |     F |   F  ||    F
 o ||   T  |    F |     F |       F |  T ||   T  |   T  |   T  |     T |   F  ||    F
 o ||   T  |    F |     F |       T |  F ||   T  |   F  |   T  |     F |   F  ||    F
 o ||   T  |    F |     F |       T |  T ||   T  |   F  |   T  |     T |   F  ||    F
 o ||   T  |    F |     T |       F |  F ||   T  |   T  |   T  |     F |   T  ||    F
 o ||   T  |    F |     T |       F |  T ||   F  |   T  |   T  |     T |   T  ||    F
 o ||   T  |    F |     T |       T |  F ||   T  |   F  |   T  |     F |   T  ||    F
 o ||   T  |    F |     T |       T |  T ||   T  |   F  |   T  |     T |   T  ||    F
 o ||   T  |    T |     F |       F |  F ||   T  |   T  |   T  |     F |   F  ||    T
 o ||   T  |    T |     F |       F |  T ||   T  |   T  |   T  |     T |   F  ||    T
 o ||   T  |    T |     F |       T |  F ||   T  |   T  |   T  |     F |   F  ||    T
 o ||   T  |    T |     F |       T |  T ||   T  |   T  |   T  |     T |   F  ||    T
 o ||   T  |    T |     T |       F |  F ||   T  |   T  |   T  |     F |   T  ||    T
 o ||   T  |    T |     T |       F |  T ||   F  |   T  |   T  |     T |   T  ||    T
 o ||   T  |    T |     T |       T |  F ||   T  |   T  |   T  |     F |   T  ||    T
M+ ||   T  |    T |     T |       T |  T ||   T  |   T  |   T  |     T |   T  ||    T
```

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

```
spec Weekend2 =
  props tennis, shop, sunny, sat, we
  . sunny /\ we => tennis    %(SWT)%
  . tennis => shop           %(TSh)%
  . sat                      %(sat)%
  . sunny                    %(sun)%
  . shop                     %(shop)% %implied
end
```

Listing 3: Example of a non-consequence

```
Legend:
M = model of the premises
+ = OK, model fulfils conclusion
- = not OK, counterexample for logical consequence
o = OK, premises are not fulfilled, hence conclusion is irrelevant

     ------------ signature -----------   ------ premises ------   conclusion
   || sat | shop | sunny | tennis | we || SWT | TSh | sat | sun || shop
===++=====+======+=======+========+====++=====+=====+=====+=====++=====
 o ||   F |    F |     F |      F |  F ||   T |   T |   F |   F ||    F
 o ||   F |    F |     F |      F |  T ||   T |   T |   F |   F ||    F
 o ||   F |    F |     F |      T |  F ||   T |   F |   F |   F ||    F
 o ||   F |    F |     F |      T |  T ||   T |   F |   F |   F ||    F
 o ||   F |    F |     T |      F |  F ||   T |   T |   F |   T ||    F
 o ||   F |    F |     T |      F |  T ||   F |   T |   F |   T ||    F
 o ||   F |    F |     T |      T |  F ||   T |   F |   F |   T ||    F
 o ||   F |    F |     T |      T |  T ||   T |   F |   F |   T ||    F
 o ||   F |    T |     F |      F |  F ||   T |   T |   F |   F ||    T
 o ||   F |    T |     F |      F |  T ||   T |   T |   F |   F ||    T
 o ||   F |    T |     F |      T |  F ||   T |   T |   F |   F ||    T
 o ||   F |    T |     F |      T |  T ||   T |   T |   F |   F ||    T
 o ||   F |    T |     T |      F |  F ||   T |   T |   F |   T ||    T
 o ||   F |    T |     T |      F |  T ||   F |   T |   F |   T ||    T
 o ||   F |    T |     T |      T |  F ||   T |   T |   F |   T ||    T
 o ||   F |    T |     T |      T |  T ||   T |   T |   F |   T ||    T
 o ||   T |    F |     F |      F |  F ||   T |   T |   T |   F ||    F
 o ||   T |    F |     F |      F |  T ||   T |   T |   T |   F ||    F
 o ||   T |    F |     F |      T |  F ||   T |   F |   T |   F ||    F
 o ||   T |    F |     F |      T |  T ||   T |   F |   T |   F ||    F
M- ||   T |    F |     T |      F |  F ||   T |   T |   T |   T ||    F
 o ||   T |    F |     T |      F |  T ||   F |   T |   T |   T ||    F
 o ||   T |    F |     T |      T |  F ||   T |   F |   T |   T ||    F
 o ||   T |    F |     T |      T |  T ||   T |   F |   T |   T ||    F
 o ||   T |    T |     F |      F |  F ||   T |   T |   T |   F ||    T
 o ||   T |    T |     F |      F |  T ||   T |   T |   T |   F ||    T
 o ||   T |    T |     F |      T |  F ||   T |   T |   T |   F ||    T
 o ||   T |    T |     F |      T |  T ||   T |   T |   T |   F ||    T
M+ ||   T |    T |     T |      F |  F ||   T |   T |   T |   T ||    T
 o ||   T |    T |     T |      F |  T ||   F |   T |   T |   T ||    T
M+ ||   T |    T |     T |      T |  F ||   T |   T |   T |   T ||    T
M+ ||   T |    T |     T |      T |  T ||   T |   T |   T |   T ||    T
```

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

```
spec Sp =
    Σ_I
    Γ_I
then
    Σ_Δ
    Γ_Δ
end
```

```
spec Animals =
    props bird, penguin
    . penguin => bird
then
    prop can_fly
    . penguin => not can_fly
end
```

Listing 5: Theory extensions in CASL.

```
logic Propositional

spec Animal =
  props bird, penguin, living
  . penguin => bird  %(pb)%
  . bird => living   %(bl)%
then %cons
  prop animal
  . bird => animal   %(ba)%
  . animal => living %(al)%
end
```

```

spec Penguin =
  props bird, penguin
  . penguin => bird         %(pb)%
then
  prop can_fly
  . bird => can_fly         %(bc)%
  . penguin => not can_fly  %(pnc)%
end
```

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

Definition 7 A signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ induces a *sentence translation* $\sigma : \mathrm{Sen}(\Sigma_1) \to \mathrm{Sen}(\Sigma_2)$, defined inductively by

- $\sigma(\bot) = \bot$
- $\sigma(\top) = \top$
- $\sigma(\phi_1 \wedge \phi_2) = \sigma(\phi_1) \wedge \sigma(\phi_2)$
- 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 σ and then look up the interpretation in the original model M.

Definition 8 A signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ induces a *model reduction* $\_|_\sigma : \mathrm{Mod}(\Sigma_2) \to \mathrm{Mod}(\Sigma_1)$. Given $M \in \mathrm{Mod}(\Sigma_2)$ i.e. $M : \Sigma \to \{T, F\}$, then $M|_\sigma \in \mathrm{Mod}(\Sigma_1)$ is defined as $M|_\sigma(\phi) := M(\sigma(\phi))$ i.e. $M|_\sigma = M \circ \sigma$. Dually, M is called a σ-*expansion* of $M|_\sigma$.

Sentence and model translation interact well with each other:

Theorem 1 (Satisfaction condition) *Given a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, $M_2 \in Mod(\Sigma_2)$ 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.")

This condition is the central ingredient of the notion of institution [GB92]. In a special course, I first introduce several examples of the satisfaction condition in order to motivate this abstract notion.

Definition 9 A *theory morphism* $(\Sigma_1,\Gamma_1) \to (\Sigma_2,\Gamma_2)$ is a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ such that for $M_2 \in \mathrm{Mod}(\Sigma_2,\Gamma_2)$ we have $M_2|_\sigma \in \mathrm{Mod}(\Sigma_1,\Gamma_1)$

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 $\sigma : (\Sigma_1,\Gamma_1) \to (\Sigma_2,\Gamma_2)$, where $\Sigma_2 = \Sigma_1 \cup \Sigma_\Delta$ and $\Gamma_2 = \Gamma_1 \cup \Gamma_\Delta$, such that $\sigma : \Sigma_1 \to \Sigma_2$ is the inclusion.

We are now ready to define conservative extensions:

Definition 10 A theory morphism $\sigma : T_1 \to T_2$ is *model-theoretically-conservative*, if any $M_1 \in \mathrm{Mod}(T_1)$ has a σ-expansion to a T_2-model.

We can now evaluate which of the extensions shown in Listing 6 are conservative. Actually, the first extension is conservative. The truth table output by HETS is shown in Listing 7. On the left hand side (columns `bird`, `living` and `penguin`), we see valuations of the smaller signature, and the corresponding evaluations of the axioms `pb` and `bl`. Models of the smaller theory are marked with an `M` in the leftmost column. On the right hand side in column `animal`, possible expansions to the larger signature are listed, together with the evaluation of the axioms `ba` and `al`. Models of the larger theory are marked with an `M` in the column right to the middle. Altogether, we can see that each model of the smaller theory has at least one expansion to the larger theory.

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 2 *If* $T_1 \xrightarrow{\sigma_1} T_2 \xrightarrow{\sigma_2} \ldots \xrightarrow{\sigma_{n-1}} T_n$ *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.

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.

```
Legend:
M = model of the axioms
+ = OK, has expansion
- = not OK, has no expansion, hence conservativity fails
o = OK, not a model of the axioms, hence no expansion needed

      ---- small signature ---  -- axioms ---        large    ---- axioms ----
                                                    signature
   || bird | living | penguin ||  pb  |  bl  ||   || animal ||   ba   |   al
===++======+========+=========++======+======++===++========++========+=======
M+ ||    F |       F |        F ||    T |    T || M ||      F ||       T |       T
   ||      |         |          ||      |      ||   ||      T ||       T |       F
---++------+--------+---------++------+------++---++--------++--------+-------
 o ||    F |       F |        T ||    F |    T ||   ||        ||        |
---++------+--------+---------++------+------++---++--------++--------+-------
M+ ||    F |       T |        F ||    T |    T || M ||      F ||       T |       T
   ||      |         |          ||      |      || M ||      T ||       T |       T
---++------+--------+---------++------+------++---++--------++--------+-------
 o ||    F |       T |        T ||    F |    T ||   ||        ||        |
---++------+--------+---------++------+------++---++--------++--------+-------
 o ||    T |       F |        F ||    T |    F ||   ||        ||        |
---++------+--------+---------++------+------++---++--------++--------+-------
 o ||    T |       F |        T ||    T |    F ||   ||        ||        |
---++------+--------+---------++------+------++---++--------++--------+-------
M+ ||    T |       T |        F ||    T |    T ||   ||      F ||       F |       T
   ||      |         |          ||      |      || M ||      T ||       T |       T
---++------+--------+---------++------+------++---++--------++--------+-------
M+ ||    T |       T |        T ||    T |    T ||   ||      F ||       F |       T
   ||      |         |          ||      |      || M ||      T ||       T |       T
```

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

```
Legend:
M = model of the axioms
+ = OK, has expansion
- = not OK, has no expansion, hence conservativity fails
o = OK, not a model of the axioms, hence no expansion needed

      small signature  axioms         large    --- axioms ---
                                     signature
   || bird | penguin ||  pb  ||   || can_fly ||   bc   | pnc
===++======+=========++======++===++=========++========+=====
M+ ||    F |       F ||    T || M ||       F ||      T |    T
   ||      |         ||      || M ||       T ||      T |    T
---++------+---------++------++---++---------++--------+-----
 o ||    F |       T ||    F ||   ||         ||        |
---++------+---------++------++---++---------++--------+-----
M+ ||    T |       F ||    T ||   ||       F ||      F |    T
   ||      |         ||      || M ||       T ||      T |    T
---++------+---------++------++---++---------++--------+-----
M- ||    T |       T ||    T ||   ||       F ||      F |    T
   ||      |         ||      ||   ||       T ||      T |    F
```

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

```
spec Liar0 =
    prop mary
    props harry, john
    . harry => john                %(whenjohn)%
    . john => (mary => not harry)  %(whenharry)%
then %implies
  . harry %(harry)%
  . john  %(john)%
  . mary  %(mary)%
  . not harry %(notharry)%
  . not john  %(notjohn)%
  . not mary  %(notmary)%
end
```

Listing 9: A circular set of statements

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

```
Legend:
M = model of the premises
+ = OK, model fulfils conclusion
- = not OK, counterexample for logical consequence
o = OK, premises are not fulfilled, hence conclusion is irrelevant

     ----- signature -----   ----- premises ------   conclusion
   || harry | john | mary || whenjohn | whenharry || harry
===++=======+======+======++==========+===========++======
M- ||     F |    F |    F ||        T |         T ||     F
M- ||     F |    F |    T ||        T |         T ||     F
M- ||     F |    T |    F ||        T |         T ||     F
M- ||     F |    T |    T ||        T |         T ||     F
 o ||     T |    F |    F ||        F |         T ||     T
 o ||     T |    F |    T ||        F |         T ||     T
M+ ||     T |    T |    F ||        T |         T ||     T
 o ||     T |    T |    T ||        T |         F ||     T
```

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

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 11 Given a propositional signature Σ and two Σ-models M_1 and M_2, then $M_1 \leq M_2$ if $M_1(p) = \mathrm{T}$ implies $M_2(p) = \mathrm{T}$ for all $p \in \Sigma$.

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

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

Note that a least model need not exist; in this case, the model class of **free**$\{SP\}$ 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.

```
spec Liar1 =
  free {
    prop mary
    props harry, john
    . harry => john              %(whenjohn)%
    . john => (mary => not harry)  %(whenharry)%
   }
then %implies
  . not harry %(notharry)%
  . not john  %(notjohn)%
  . not mary  %(notmary)%
end
```

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

```
      ----- signature -----   -------- premises --------   conclusion
   || 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.

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

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:

```
spec Liar2 =
  cofree {
    prop mary
    props harry, john
    . harry => john              %(whenjohn)%
    . john => (mary => not harry)  %(whenharry)%
   }
then %implies
  . false %(false)%
end
```

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

```
      ----- signature -----    ---------- premises ---------     conclusion
   || harry | john | mary || whenjohn | whenharry | cofree || false
===++=======+======+======++==========+===========+========++======
 o ||     F |    F |    F ||        T |         T |      F ||     F
 o ||     F |    F |    T ||        T |         T |      F ||     F
 o ||     F |    T |    F ||        T |         T |      F ||     F
 o ||     F |    T |    T ||        T |         T |      F ||     F
 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 14: Truth table for the specification of Listing 13

```
spec Liar3 =
  prop mary
then
  free {
    props harry, john
    . harry => john              %(whenjohn)%
    . john => (mary => not harry)  %(whenharry)%
   }
then %implies
  . not harry %(notharry)%
  . not john  %(notjohn)%
end
```

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

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

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

```
      ----- signature -----    ---------- premises --------    conclusion
    || harry | john | mary || whenjohn | whenharry | free || notharry
====++=======+======+======++==========+===========+======++=========
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

```
spec Liar4 =
  prop mary
then
  cofree {
    props harry, john
    . harry => john              %(whenjohn)%
    . john => (mary => not harry) %(whenharry)%
   }
then %implies
  . harry \/ mary %(harrymary)%
  . john  %(john)%
end
```

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

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

$\mathrm{Mod}(SP_1$ **then cofree**$\{SP_2\}) =$
$\{M \in \mathrm{Mod}(SP_1$ **then** $SP_2) \mid$
M is the greatest model in $\{M' \in \mathrm{Mod}(SP_1$ **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.

3 Conclusion

The overall picture is as follows: typically, I start with a course on first-order logic as described in Section 1, followed by a more special course on structuring and institutions, following Section 2.

```
1       ----- signature -----    --------- premises -----------    conclusion
2     || harry | john | mary || whenjohn | whenharry | cofree || harrymary
3  ===++=======+======+======++==========+===========+========++==========
4   o ||     F |    F |    F ||        T |         T |      F ||         F
5   o ||     F |    F |    T ||        T |         T |      F ||         T
6   o ||     F |    T |    F ||        T |         T |      F ||         F
7  M+ ||     F |    T |    T ||        T |         T |      T ||         T
8   o ||     T |    F |    F ||        F |         T |      F ||         T
9   o ||     T |    F |    T ||        F |         T |      F ||         T
10 M+ ||     T |    T |    F ||        T |         T |      T ||         T
11  o ||     T |    T |    T ||        T |         F |      F ||         T
```

Listing 18: Truth table for the specification of Listing 17

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!

Acknowledgments This work has been supported by the German Federal Ministry of Education and Research (Project 01 IW 07002 FormalSafe).

Bibliography

[AKK99] E. Astesiano, H.-J. Kreowski, B. Krieg-Brückner. *Algebraic Foundations of Systems Specification*. Springer, 1999.

[BE02] J. Barwise, J. Etchemendy. *Language, proof and logic*. CSLI publications, 2002.

[BM04] M. Bidoit, P. D. Mosses. CASL *User Manual*. Lecture Notes in Computer Science (IFIP Series) 2900. Springer, 2004. With chapters by T. Mossakowski, D. Sannella, and A. Tarlecki.
http://www.cofi.info/index.php/CASL_user_manual

[CoF04] CoFI (The Common Framework Initiative). CASL *Reference Manual*. Lecture Notes in Computer Science (IFIP Series) 2960. Springer, 2004.
http://www.cofi.info/index.php/CASL_reference_manual

[GB92] J. Goguen, R. Burstall. Institutions: Abstract Model Theory for Specification and Programming. *Journal of the ACM* 39(1):95–146, 1992.

[Mei00] A. Meier. System Description: TRAMP: Transformation of Machine-Found Proofs into ND-Proofs at the Assertion Level. In McAllester (ed.), *Automated Deduction - CADE-17, 17th International Conference on Automated Deduction, Pittsburgh, PA, USA, June 17-20, 2000, Proceedings*. Lecture Notes in Computer Science 1831, pp. 460–464. Springer, 2000.

[MML07] T. Mossakowski, C. Maeder, K. Lüttich. The Heterogeneous Tool Set. In Grumberg and Huth (eds.), *TACAS 2007*. Lecture Notes in Computer Science 4424, pp. 519–522. Springer, 2007.

[Mos05] T. Mossakowski. Heterogeneous specification and the heterogeneous tool set. 2005. Habilitation thesis, University of Bremen.

[MQP06] J. Meng, C. Quigley, L. C. Paulson. Automation for Interactive Proof: First Prototype. *Information and Computation* 204(10):1575–1596, 2006. http://www.sciencedirect.com/science/article/B6WGK-4K9C6J3-1/2/e4e5661335eef1bb58ed32a67a7f2427

[PSST08] A. Pease, G. Sutcliffe, N. Siegel, S. Trac. The Annual SUMO Reasoning Prizes at CASC. In Konev et al. (eds.), *Proceedings of the First International Workshop on Practical Aspects of Automated Reasoning, Sydney, Australia, August 10-11, 2008*. CEUR Workshop Proceedings 373. CEUR-WS.org, 2008. http://ceur-ws.org/Vol-373/paper-05.pdf

[WBH+02] C. Weidenbach, U. Brahm, T. Hillenbrand, E. Keen, C. Theobalt, D. Topic. SPASS Version 2.0. In Voronkov (ed.), *Automated Deduction – CADE-18*. Lecture Notes in Computer Science 2392, pp. 275–279. Springer, 2002.