Abstract logical constants

Tin Perkov*

June 3, 2018

Abstract

A possibility of defining logical constants within abstract logical frameworks is discussed, in relation to abstract definition of logical consequence. We propose using duals as a general method of applying the idea of invariance under replacement as a criterion for logicality.

1 Introduction

Which symbols (or expressions) of a (formal) language are logical? Fundamental questions like this one are rarely answered with consensus and often lead to controversies, but trying to answer such questions turns out to be very fruitful for developing useful theories, with sometimes unintended applications.

Without ambition to give a comprehensive overview on the topic, let us point out some approaches to answer the question of logicality:

- grammatical (atomic sentences are non–logical, while complex sentences are built using logical connectives)
- proof–theoretical (logical is what is applied in the same way to any reasoning, regardless of its subject, i.e. definable by inferential rules)
- semantical (logical is what has fixed meaning, not depending on properties of individuals, i.e. invariant under permutations, isomorphisms and so on).

We focus on a recent development [2] in the semantical approach, which explores a close relation between logical constants and logical consequence. We can have in mind the following goals:

- ambitious: find the proper notion of logical constants – probably no answer
- less ambitious: understand how a *choice* of constants generates a consequence relation, and vice versa.

*This work has been supported by Croatian Science Foundation (HRZZ) under the project UIP-05-2017-9219.

For the latter converse goal, we use the idea of consequence extraction as presented by Bonnay and Westerståhl in [2]. Given a language and a consequence relation, they consider a symbol to be a constant if replacing it with another symbol of the same category (categories being e.g. binary Boolean connectives, unary modal operators, quantifiers and so on) makes at least one valid inference of that consequence relation to fail.

Unfortunately, it is often the case that there is only one symbol of a given category in a language, so we have nothing to replace it with in order to test whether it is a constant. This can be solved by introducing a new symbol which does not essentially change the language. This is not very unnatural, in fact it is often done in usual expositions of logical theories: for the sake of simplicity we have a minimal number of primitives, while other symbols are defined as abbreviations. Introducing new symbols depends, however, on the nature of a particular language. This is probably easy to do from case to case, but it is attempted here to give a general method that would work in a broad family of languages in a uniform way.

The idea is very simple, but fairly general: we assume that any symbol s such as connective, quantifier, modal operator and so on, has the dual s', either already present in the language, or we introduce it in it. Given a consequence relation $\Rightarrow$, duality means that we have valid inferences of the form $s'(\varphi_1, \varphi_2, \dots) \Leftrightarrow \neg s(\neg\varphi_1, \neg\varphi_2, \dots)$ and $s(\varphi_1, \varphi_2, \dots) \Leftrightarrow \neg s'(\neg\varphi_1, \neg\varphi_2, \dots)$. If s is not self–dual, then at least one of these inferences fails if we replace s with s'. Therefore, s is a constant. As for self–duals, we use the following trick: replace with some other symbol of the same category which is not self–dual, thus making at least one of the inferences which express self–duality to fail.

It is rather obvious that any missing dual can always be included in a language by defining it as an abbreviation and then adding it to the list of symbols, and that this way the language remains essentially the same. So, for any symbol that is used inductively in building formulas in a way that the truth of these formulas depends on the truth of one or more (depending on arity) formulas in the scope of that symbol, we have the dual symbol. By the reasoning presented above, it is immediately verified that any such symbol is a constant for a given language and a consequence relation. Whether one considers this desirable or not, this turns out to coincide with the grammatical approach, as summarized above.

Another goal of the paper is to outline possible generalizations of techniques from [2], in particular to the framework of abstract logic (cf. e.g. [3]). For this purpose, in Section 2 three abstract logical frameworks are compared, to show that the abstract logic setting is the most general. In Section 3 we compare two abstract definitions of consequence relation and show that they are equivalent, and furthermore we review basic ideas from [2]. Section 4 contains details on the idea of using duals as candidates for replacement. We conclude with Section 5, which contains remarks on refining the definition of abstract logic to encompass the previously presented ideas.

2 Comparison of some abstract logical frameworks

There are various abstract frameworks for general reasoning about logic, depending on authors' focus or intends. Let us compare three such approaches, as presented in [2], [3] and [4].

Having the usual way of presenting logical systems in mind (the syntax of a language, then a class of structures on which the language is interpreted, and finally the semantics – a way in which syntax relates to structures), abstract model theory provides the following definition (cf. [3]).

Definition 1. An *abstract logic* is a triple $L = (S, F, \models)$, where S is a class of *structures* or *models*, F is a set of *sentences* or *formulas* and $\models$ is a binary relation between elements of S and elements of F. We call $\models$ the *satisfaction relation.*

Let $\mathfrak{M}$ be a structure and Γ a set of sentences. We write $\mathfrak{M} \models \Gamma$ if $\mathfrak{M} \models \varphi$ for all $\varphi \in \Gamma$.

In full generality this definition has no further conditions on S, F and $\models$, i.e. *structures* and *sentences* are just names for elements of S and F, respectively. A particular example in which S is the class of all first–order structures and F is closed under conjunction and negation (this closure is defined in a natural way, see below) provides the framework for famous Lindström theorems. These theorems themselves have some further assumptions on F and $\models$.

But even in full generality we can already define usual logical notions like model of a given formula, theory of a given structure, logical equivalence of formulas or elementary equivalence of structures, compactness of an abstract logic etc. In particular, since the other two frameworks we consider focus on logical consequence, let us emphasize that we can define the notion of logical consequence in an abstract logic in the following natural way.

Definition 2. Let $L = (S, F, \models)$ be an abstract logic. Let Γ be a set of sentences and φ a sentence. We say that φ is a *logical consequence* of Γ and we write $\Gamma \models \varphi$ if for each $\mathfrak{M} \in S$ the following holds: if $\mathfrak{M} \models \Gamma$, then $\mathfrak{M} \models \varphi$.

Instead of $\{\varphi\} \models \psi$ we write $\varphi \models \psi$ (similarly also for other notions of consequence considered below).

Another way to define the notion of logical consequence is the following (cf. [4] or [5]).

Definition 3. Let $\mathcal{L} = (F, \mathcal{M})$, where F is a set and $\mathcal{M}$ is a set of partitions (T, U), $T, U \subseteq F$, $T \cup U = F$, $T \cap U = \emptyset$. Elements of F are called *sentences* or *formulas* and elements of $\mathcal{M}$ are called *possible states of affairs.*

Let $\Gamma \subseteq F$ and $\varphi \in F$. We say that φ is a *consequence* of Γ w.r.t. $\mathcal{M}$ and we write $\Gamma \Vdash_{\mathcal{M}} \varphi$ if for all $(T, U) \in \mathcal{M}$ we have: if $\Gamma \subseteq T$, then $\varphi \in T$.

Let $\Vdash$ be a binary relation between subsets of F and elements of F. We say that $\Vdash$ is a *consequence relation* if there is $\mathcal{M}$ such that $\Vdash = \Vdash_{\mathcal{M}}$.[1]

Clearly, each, $(T, U) \in \mathcal{M}$ is fully determined by T, so we can simplify the definition and consider $\mathcal{M}$ to be simply a family of subsets of F. Intended interpretation is that $T \in \mathcal{M}$ is the set of all true sentences in a possible state of affairs.

The following proposition shows that there is a natural connection between these two frameworks. The proof is trivial, so we omit it.

Proposition 1. Let $L = (S, F, \models)$ be an abstract logic. Let $\mathfrak{M} \in S$. Let $T_{\mathfrak{M}}$ be the theory of $\mathfrak{M}$, i.e. $T_{\mathfrak{M}} = \{\varphi \in F : \mathfrak{M} \models \varphi\}$. Put $\mathcal{M} = \{T_{\mathfrak{M}} : \mathfrak{M} \in S\}$. Then the logical consequence of L equals $\Vdash_{\mathcal{M}}$.

Conversely, let $\mathcal{L} = (F, \mathcal{M})$. Let $(T, F \setminus T) \in \mathcal{M}$. Put $T \models \varphi$ if and only if $\varphi \in T$. Then $(\mathcal{M}, F, \models)$ is an abstract logic.

The notion of abstract logic is more general, since in the other framework structures are identified with true sentences, i.e. elementarily equivalent structures are identified, while abstract logic allows reasoning about distinct structures with equal theories.

At the first glance, the third setting (cf. [2]) just seems to fix a state of affairs from the previous framework, at the same time giving some more structure to the syntax, i.e. F is no longer just a set, it is actually a language over an alphabet.

Definition 4. An *interpreted language* is a triple $L = (A, F, T)$, where A is a set which we call an *alphabet*, while its elements are called *symbols*, F is a set of words, i.e. finite sequences of symbols (some of which are[2]) from A, and $T \subseteq F$. We call elements of F *sentences* or *formulas*, and elements of T *true sentences*.

But then the definition of consequence corresponding to an interpreted language is somewhat different. It is given with respect to a choice of an interpreted language L and a subset $X \subseteq A$, intended to be a choice of constants. Symbols of all interpreted languages are supposed to be partitioned in classes called *categories* (e.g. binary Boolean connectives, propositional variables, ternary relational symbols and so on). Although the interpretation is fixed, in this setting replacement of symbols is used instead, with purpose to simulate reinterpretation.

Definition 5. A *replacement* is a function $\rho : A \to A$ which respects categories, i.e. $\rho(u)$ is in the same category as u for all $u \in A$.

Denote by $\varphi[\rho]$ the result of replacing each occurrence of any $u \in A$ in $\varphi \in F$ by $\rho(u)$. Analogously, we use notation $\Gamma[\rho]$ for $\Gamma \subseteq F$.

[1]This is the notion of single–conclusion consequence relation in the sense of [4]. In this paper we do not consider multiple–conclusion consequence relations.

[2]In [2] symbols not from A are allowed in sentences, having in mind auxiliary symbols like parentheses.

Fix $X \subseteq A$. Put $\Gamma \Rightarrow_X \varphi$ if and only if for each replacement ρ such that $\rho|_X = id_X$ we have: if $\Gamma[\rho] \subseteq T$, then $\varphi[\rho] \in T$.

Clearly, quantifying over replacements amounts to quantifying over states of affairs, making this framework embeddable in the previous one, as the following proposition states. Again, the proof is very easy and therefore omitted.

Proposition 2. Let $L = (A, F, T)$ be an interpreted language, let $X \subseteq A$ and let $\Rightarrow_X$ be defined as above. For each replacement $\rho : A \to A$ such that $\rho|_X = id_X$, define $T[\rho] = \{\varphi : \varphi[\rho] \in T\}$.

Furthermore, let $\mathcal{M} = \{T[\rho] : \rho \text{ is a replacement such that } \rho|_X = id_X\}$ and let $\mathcal{L} = (F, \mathcal{M})$. Then $\Vdash_{\mathcal{M}}$ equals $\Rightarrow_X$.

Remark 1. Clearly, distinct sets of true sentences T_1 and T_2 may generate the same $\mathcal{M}$, but this is a desirable identification between interpreted languages. Choice of T which serves as a starting point of quantifying over any replacement, and consequently over possible states of affairs other then T, is arbitrary. If we are interested in general considerations about consequence or other notions, and not in a particular possible state of affairs, we can still say that the previous approach is more general then this one. On the other hand, all the presented frameworks allow reasoning about particular possible states of affairs, if needed.

We leave for a future consideration the following converse question: given $\mathcal{L} = (F, \mathcal{M})$, can we (or under which conditions we can) find A, T and $X \subseteq A$ such that $\Rightarrow_X$ equals $\Vdash_{\mathcal{M}}$.

3 Logical constants and logical consequence

To reason about fundamental question of logical constants, we need an abstract definition of logical consequence.

The following characterization of consequence relation (in the sense of Definition 3) is given in [4].

Theorem 1. Given F, a relation $\Vdash \subseteq \mathcal{P}(F) \times F$ is a consequence relation if and only if all of the following always hold:

1. if $\varphi \in \Gamma$, then $\Gamma \Vdash \varphi$
2. if $\Gamma' \Vdash \varphi$ and $\Gamma' \subseteq \Gamma$, then $\Gamma \Vdash \varphi$
3. if $\Gamma \cup \Delta \Vdash \varphi$ and $\Gamma \Vdash \psi$ for all $\psi \in \Delta$, then $\Gamma \Vdash \varphi$.

We can consider conditions of the theorem to be an abstract definition of consequence relation. Another abstract definition of this notion is given in the setting of interpreted languages (cf. [2]).

Definition 6. Given F, a *consequence relation* is $\Rightarrow \subseteq \mathcal{P}(F) \times F$ such that the following always hold:

1. if $\varphi \in \Gamma$, then $\Gamma \Rightarrow \varphi$

2. if $\Delta \Rightarrow \varphi$ and $\Gamma \Rightarrow \psi$ for all $\psi \in \Delta$, then $\Gamma \Rightarrow \varphi$.

If $\varphi \Rightarrow \psi$ and $\psi \Rightarrow \varphi$, we write $\varphi \Leftrightarrow \psi$.

Let us point out that these definitions are in fact equivalent.

Proposition 3. Given F, let $\Vdash \subseteq \mathcal{P}(F) \times F$. Then $\Vdash$ satisfies the conditions of Theorem 1 if and only if $\Vdash$ is a consequence relation in the sense of Definition 6.

Proof. Let $\Vdash$ satisfy the conditions of Theorem 1 and let $\Delta \Vdash \varphi$ and $\Gamma \Vdash \psi$ for all $\psi \in \Delta$. Now, the second condition of Theorem 1 implies $\Gamma \cup \Delta \Vdash \varphi$ and thus the third condition implies $\Gamma \Vdash \varphi$, as desired.

Conversely, let $\Vdash$ be a consequence relation in the sense of Definition 6. Let us prove the second condition of Theorem 1. Let $\Gamma' \Vdash \varphi$ and $\Gamma' \subseteq \Gamma$. The first condition implies in particular that $\Gamma \Vdash \psi$ for all $\psi \in \Gamma'$. Hence, the second condition of Definition 6 implies the claim. It remains to prove the third condition of Theorem 1. Let $\Gamma \cup \Delta \Vdash \varphi$ and $\Gamma \Vdash \psi$ for all $\psi \in \Delta$. But from the first condition we also have $\Gamma \Vdash \psi$ for all $\psi \in \Gamma$. So, we actually have $\Gamma \Vdash \psi$ for all $\psi \in \Gamma \cup \Delta$. Now, we apply the second condition of Definition 6 to conclude $\Gamma \Vdash \varphi$, as desired. $\square$

Due to Proposition 1, we immediately obtain the following corollary.

Corollary 1. Given F, let $\Vdash \subseteq \mathcal{P}(F) \times F$. Then $\Vdash$ is a consequence relation in the sense of Definition 6 if and only if there is an abstract logic $L = (S, F, \models)$ such that $\Vdash$ is the relation of logical consequence of L in the sense of Definition 2.

Keeping in mind these possibilities of generalization, let us go back to Bonnay and Westerståhl [2], who define constants with respect to a given consequence relation as follows.

Definition 7. Let $L = (A, F, T)$ be an interpreted language and let $\Rightarrow$ be a consequence relation which is truth preserving, i.e. for all $\Gamma \subseteq T$ we have: if $\Gamma \Rightarrow \varphi$, then $\varphi \in T$.

We define the set of *logical constants* $C_{\Rightarrow} \subseteq A$ by putting $u \in C_{\Rightarrow}$ if and only if there are $\Gamma \subseteq F$, $\varphi \in F$ and a replacement ρ which is identity on $A \setminus \{u\}$ such that $\Gamma \Rightarrow \varphi$ and $\Gamma[\rho] \not\Rightarrow \varphi[\rho]$.

In other words, given a language and a consequence relation, a symbol is a constant if replacing it with another symbol of the same category makes at least one valid inference of that consequence relation to fail.

Remark 2. Let L be an interpreted language. It is easy to see (cf. [2]) that the smallest and the largest truth preserving consequence relation with respect to L are defined as follows:

- $\Gamma \Rightarrow_{\text{min}} \varphi$ if and only if $\varphi \in \Gamma$
- $\Gamma \Rightarrow_{\text{max}} \varphi$ if and only if it is not the case $\Gamma \subseteq T$ and $\varphi \notin T$.

Note that we can generalize the definition by omitting the requirement that $\Rightarrow$ is truth preserving and that thus generalized definition does not depend on T.

In [2] the definition is tested on familiar examples like propositional and first–order logic, providing the results one should expect: constants extracted from their logical consequence relations are their standard sets of logical symbols.

We will not go further in reviewing results of [2], since basic definitions suffice for out purposes. Instead, let us focus on a way in which a single given symbol is proved to be a constant. Consider some examples.

Example 1. Let $\models_{PL}$ be the standard logical consequence relation of classical propositional logic. To see that $\vee$ is in $C_{\models_{PL}}$, note that $p \models_{PL} p \vee q$, but $p \not\models_{PL} p \wedge q$.

Example 2. Let $\models_{FO}$ be the standard logical consequence relation of first–order logic. Then $\forall$ is in $C_{\models_{FO}}$, since $\forall x A \models_{FO} \neg\exists x \neg A$, but $\exists x A \not\models_{FO} \neg\exists x \neg A$.

One technical problem with this approach is that there is often only one symbol of a given category in A, so we have nothing to replace it with in order to test whether it is a constant. In fact, it is usually convenient not to have too many primitive symbols. Notably, regarding our examples, we often have only one binary Boolean connective and one quantifier in the alphabet, while the others are defined as abbreviations.

In [2] a simple solution of this difficulty is proposed: introduce a new symbol which does not essentially change the language. In this case it is more convenient to have more symbols (at least two of each category), so we let them be in A. This provides what we need to prove that a symbol is a constant. Another solution is to assign categories to some expressions, not only symbols, thus allowing expressions to be considered as possible constants. Both approaches imply that a language may have many hidden logical constants, i.e. constants definable by primitive symbols of a language. It may be debatable whether this is desirable.

4 Dual symbols

Recall that we proved that $\vee$ is a constant by replacing it with $\wedge$, which is its dual, i.e. $\varphi \wedge \psi$ is equivalent to $\neg(\neg\varphi \vee \neg\psi)$. Also, we proved that $\forall$ is a constant by replacing it with its dual $\exists$. Consider a similar example from modal logic.

Example 3. Consider the basic modal logic with the standard (local) consequence relation $\Vdash_{ML}$, as defined, e.g., in [1]. To show that $\Box \in C_{\Vdash_{ML}}$, include its dual $\Diamond$ in the language. From duality itself we have $\Box p \Vdash_{ML} \neg\Diamond\neg p$, but $\Diamond p \nVdash_{ML} \neg\Diamond\neg p$.

Let us pursue the following idea: if a symbol is unique of its category, introduce its dual to the language and use it to show it is a constant. Or more generally, why not always use duals, whether a symbol is unique of its category or not?

To apply this, we need more structure in the abstract definition of language. We try to give minimal requirements, which are trivially fulfilled in usual recursively defined formal languages.

Definition 8. Let $L = (A, F, T)$ be an interpreted language and let $u \in A$. For any $\varphi \in F$ in which u occurs, and for any occurrence of u in φ, let ψ be the subsentence which contains this occurrence of u, but no proper subsentence of ψ contains this occurrence of u.

Arity of $u \in A$ is $k \in \mathbb{N}$ such that each such ψ has exactly k maximal proper subsentences $\psi_1, \ldots, \psi_k$. We denote ψ by $u(\psi_1, \ldots, \psi_k)$.

Example 4. In the sense of the above definition:

- $\neg$, $\Diamond$, $\Box$, $\forall$, $\exists$ are unary
- $\vee$, $\wedge$, $\rightarrow$ are binary.

Definition 9. We say that an interpreted language $L = (A, F, T)$ is *closed under (classical) negation* if for all $\varphi \in F$ there is $\psi \in F$ such that $\psi \in T$ if and only if $\varphi \notin T$. We denote ψ by $\neg\varphi$.

Let $L = (A, F, T)$ be closed under negation and let $\Rightarrow$ be a consequence relation. We say that L is *closed under duals* (with respect to $\Rightarrow$) if for all $k > 0$ and for each k–ary $u \in A$, there is a k–ary $u' \in A$ of the same category such that $u'(\psi_1, \ldots, \psi_k) \Leftrightarrow \neg u(\neg\psi_1, \ldots, \neg\psi_k)$ and $u(\psi_1, \ldots, \psi_k) \Leftrightarrow \neg u'(\neg\psi_1, \ldots, \neg\psi_k)$.

Example 5. Our previous examples of duals comply with this definition: $\vee$ and $\wedge$, $\forall$ and $\exists$, $\Box$ and $\Diamond$. As we have already noted, usually only one symbol from each of these pairs is considered primitive, while the other is defined as an abbreviation. In some cases dual is not often used even as an abbreviation, e.g. there is no standard notation for dual of $\rightarrow$. We can denote it $\not\leftarrow$ (where $A \not\leftarrow B$ is an abbreviation for $\neg(B \rightarrow A)$). But for our current purposes, we suppose all duals are present in the language.

Proposition 4. *Let L be an interpreted language and $\Rightarrow$ a consequence relation. Let L be closed under duals with respect to $\Rightarrow$, let $k > 0$, and let $u \in A$ be any k–ary symbol. If u is not self–dual, then it is a constant.*

Proof. Since L is closed under duals, there is a k–ary symbol $u' \in A$ such that $u'(\psi_1, \ldots, \psi_k) \Leftrightarrow \neg u(\neg\psi_1, \ldots, \neg\psi_k)$ and $u(\psi_1, \ldots, \psi_k) \Leftrightarrow \neg u'(\neg\psi_1, \ldots, \neg\psi_k)$. Consider the replacement ρ which is identity on $A \setminus \{u\}$ and $\rho(u) = u'$. Since u is not self–dual, at least one of the inferences which express duality does not hold under the replacement ρ. $\Box$

This result is rather trivial, but includes seemingly vast majority of symbols we have in mind when trying to generalize the notion of logical constants, like logical connectives $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$ etc., quantifiers $\forall$, $\exists$ (also as second–order quantifiers, even polyadic) and many more, modal operators $\Diamond$, $\Box$ and so on.

As for the self–dual symbols, some such symbols are also considered to be logical constants, notably the negation itself ($\neg p \Leftrightarrow \neg\neg\neg p$). In such cases we need to use some other symbol of the same category to prove the constancy. An example of a self–dual generalized quantifier is "more then a half of", if interpreted on a finite set of odd cardinality. As a general method, in such case we can use any symbol of the same category that is not self–dual, thus making at least one of the inferences which express self–duality to fail. So for example, replacing a self–dual quantifier with $\exists$ proves that it is a constant.

Proposition 5. Let L be closed under negation and $\Rightarrow$ a consequence relation. Let $k > 0$, and let $u \in A$ be a self–dual k–ary symbol. If there exists a non–self–dual $u' \in A$ of the same category as u, then u is a constant.

Proof. Since u is self–dual, we have $u(\psi_1, \ldots, \psi_k) \Leftrightarrow \neg u(\neg\psi_1, \ldots, \neg\psi_k)$. Let ρ be the replacement which is identity on $A \setminus \{u\}$ and $\rho(u) = u'$. Since u' is not self–dual, at least one of the inferences which express self–duality does not hold under the replacement ρ. □

In case a self–dual symbol is unique of its category, we can think of introducing as an abbreviation some other symbol of the same category which is not self–dual, and this is exactly what is done in [2] in the case of $\neg$ (which is, as a rule, the only unary Boolean connective in formal languages), by introducing a non–standard unary connective defined as "equal to false". For the choice of this introduced symbol we cannot have a general recipe, but solve it from case to case.

Finally, consider 0–ary symbols. For example, predicates or relational symbols in first–order logic, propositional variables in propositional logic or modal logic and so on, are not generally considered to be logical constants.[3] Indeed, replacing e.g. a propositional variable with another propositional variable preserves valid inferences. But truth values like $\top$ and $\bot$, if included in a language, are considered logical constants. And rightly so, because we have $\top \Rightarrow \neg\bot$, but $\top \not\Rightarrow \neg\top$. We can consider $\top$ and $\bot$ dual to each other if we allow $k = 0$ (there just isn't anything in their scope to negate, so dual is simply the negation).

5 A unified framework

The above ideas are easily generalized to the level of abstract logic, with an assumption – which is not too restrictive – that a set of sentences is indeed a language over an alphabet. In fact, for any abstract logic, we can make F trivially a language over an alphabet, namely by declaring that F is the alphabet, and thus all sentences are one–letter words. This is not at all useful if we want to apply ideas from previous sections, but it does show that we do not lose any generality by the following variant of the definition of abstract logic, in an attempt to unify frameworks discussed above.

Definition 10. An *abstract logic* is $L = (S, F, \models)$, where S is a class of *structures* or *models*, F is a language over some alphabet, elements of which are

[3]However, some of them, like equality, may be considered constants if we fix a (normal) interpretation of the symbol.

called *sentences* or *formulas* and $\models$ is a binary relation between elements of S and elements of F. We call $\models$ the *satisfaction relation.*

Again we assume symbols of abstract logics to be partitioned in what we can simply call classes of symbols. As noted, all notions regarding abstract logics apply in this slightly modified setting. But now, we can also use some notions regarding interpreted languages, which require some structure on F which the previous notion of abstract logic lacked, including notions of replacement and constants generated by a given consequence relation, due to [2], and also notions of arity and duals, as defined in the previous section. Propositions from the previous section are proved for this setting virtually without any changes.

References

[1] P. Blackburn, M. de Rijke, Y. Venema: *Modal Logic*, Cambridge University Press (2001)

[2] D. Bonnay, D. Westerståhl: Consequence mining, *Journal of Philosophical Logic* 41 (2012) 671–709.

[3] M. García Matos, J. Väänänen: Abstract model theory as a framework for universal logic, in: J.-Y. Beziau (ed.) *Logica Universalis: Towards a General Theory of Logic*, 2nd edition, Birkhäuser (2007) 19–33.

[4] D.J. Shoesmith, T.J. Smiley: *Multiple–conclusion Logic*, Cambridge University Press, 1978.

[5] Z. Šikić: Singular consequence relations and order relations, *Grazer Mathematische Berichte* 304 (1989) 118–127.