Valuations

October 24, 2014

JEAN-LOUIS LENARD[1]

Abstract

Is logic empirical? Is logic to be found in the world? Or is logic rather a convention, a product of conventions, part of the many rules that regulate the language game? Answers fall in either camp. We like the linguistic answer. In this paper, we want to analyze how a linguistic community would tackle the problem of developing a logic and show how the linguistic conventions adopted by the community determine the properties of the local logic. Then show how to move from a notion of logic that varies from community to community to a notion of logic that is in a sense universal. The framework is conventional up to a point: we have sentences, atomic and composite, the connectives are interpreted, values are computed, and the value of a composite sentence is a function of the values of its sub-sentences. Less conventional is the use of a plurality of truth values, and the sharp distinction we draw between sentences and statements, in the spirit of the distinction between propositions and judgments one may find in proof theory.

The linguistic community will face many choices. What are the good ones, the ones to avoid? Are there, in some sense, optimal choices? These are the kind of issues we are addressing. Where do we end up? With some kind of universal bivalent logic, ironically enough. We start from an arbitrarily large number of truth values, atomic sentences and connectives, construct a generic many-valued logic, recover more or less the usual results and issues, and, in the end, it all comes down to a positive bivalent logic with two connectives, 'and' and 'or', as if logic is nothing more than a mere accounting of possibilities.

Key words: Many-valued logic, satisfiability, Galois connection, logical connectives, truth, bivalence, universal logic, logic and information, AI.

"How is logical certainty possible?". That is the question, posed by W.V. Quine in *Carnap and Logical Truth* [29]. Is logic empirical [11, 28]? Is logic to be found in the world? Or is logic rather a convention, a product of conventions, part of the many rules that regulate the language game? Answers fall in either

[1] *jlouis.lenard@gmail.com*

camp. We like the linguistic answer: we believe that logic has more to do with the languages that talk of the actual world than with the actual world itself. In this paper, we want to analyze how a linguistic community would tackle the problem of developing a logic and show how the linguistic conventions adopted by the community determine the properties of the local logic. Then show how to move from a notion of logic that varies from community to community to a notion of logic that is in a sense universal. The framework is conventional up to a point: we have sentences, atomic and composite, the connectives are interpreted, values are computed, and the value of a composite sentence is a function of the values of its subsentences. Less conventional is the use of a plurality of truth values and the sharp distinction we draw between sentences and statements, in the spirit of the distinction between propositions and judgments one may find in proof theory [25]. In his first Sienna lecture [25], Martin-Löf stresses the need to distinguish two kind of entities:

> " ... we have the entities that the logical operations operate on, which we call propositions, and we have those that we prove and that appear as premises and conclusion of a logical inference, which we call assertions. It turns out that, in order to clarify the meanings of the logical constants and justify the logical laws, a considerable portion of the philosophical work lies already in clarifying the notion of proposition and the notion of assertion."

We are, as Martin-Löf, interested in the meanings of the logical constants and justifications for the logical laws. Our approach is not proof theoretical however; we concern ourselves with the transfer of information between a speaker and a listener and reconstruct logical notions from such a perspective. We do not have per se propositions and assertions here; we have sentences and statements, sentences as means of expression and statements as means of assertion. Rules of formation regulate how connectives operate on sentences. A natural language, English for example, does not typically distinguish statements from sentences: the utterance "Grass is green and snow is white" may be a statement of fact or it may be an example sentence given by a teacher to illustrate some fine points of grammar. It is left to the context to disambiguate. The languages of interest in this work are formal languages that distinguish statements from sentences, that explicit intended assertoric forces. Technically, we consider typed formal languages with explicit typing annotations (the so-called Church convention for typed languages [3]). Such languages are robot-friendly and close enough to natural languages to remain an option for homo sapiens.

Let's agree on some terminology. Hereafter, sentences are means of expression and the bearers of truth values. A statement[2] is a sentence with a truth value - mathematically, a couple (sentence, value). The members of the linguistic community use statements to assert, to report facts or opinions, to express

[2]Our usage of the word 'statement' is technical - a sentence with a truth value. 'Statement' is meant to connote report of facts or report of opinions, not the content of a sentence, logical content, judgeable content, or content of any kind. In our framework, both sentences and statements are first and foremost linguistic entities, as are truth values.

claims and judgements. Our argument is more transparent, not to mention more general, with a generic set of truth values, but the wary reader is naturally free to limit her working set to the traditional {True, False}. Truth values in this paper are mere tags, linguistic entities, and have no particular metaphysical implications.

In its attempt to develop a logic, the linguistic community will face many choices. What are the good ones, the ones to avoid? Are there choices that are, in some sense, optimal? These are the kind of issues we are addressing. Where do we end up? With some kind of universal bivalent logic, ironically enough. *Naturam expellas furca, tamen usque recurret, et mala perrumpet furtim fastidia victrix*[3]. We start from an arbitrarily large number of truth values, atomic sentences and connectives, construct a generic many-valued logic, recover more or less the usual results and issues, and, in the end, it all comes down to a positive bivalent logic with two connectives, 'and' and 'or', as if logic is nothing more than a mere accounting of possibilities.

This paper is a tale of two logics: one based on sentences, the other on statements. The logic based on sentences is highly relative: its laws change from community to community, the way identities change from algebraic system to algebraic system. The logic of statements, on the other hand, is the same for everybody. It is an universal logic of constraints; it is naturally bivalent, though not required to be so, and involves two traditional connectives, 'and' and 'or'. The main result of this work is a reconstruction of logic and of classical logic notions from a linguistic perspective, from a formal language perspective strictly speaking, based on an analysis of the transfer of information between a speaker and a listener, rather than on a notion of truth or truth transmission. The analysis of the transfer of information drives our argument and presentation.

The paper is structured as follows. Section 1 sets the stage. Section 2 introduces the many-valued setting we want to work with and discuss sentence equivalence. Section 3 moves to a consideration of statements, their reading as constraints and to a rephrasing of satisfiability as a constraint satisfaction problem. Sections 3.3 and 3.4 collects a few technical results on Galois connections. Section 4 is logic from a statement perspective. Section 5 discusses pre-image analysis, our take on tableau expansion, where two constraint language connectives, 'and' and 'or', find their calling. Section 6 discusses how to improve the expressivity of the language and section 7 consider normal forms. We conclude with some clarifications, limitations, and generalizations.

1 Setting the stage

Reporting

Your name is Alice and you are the famous explorer. The one that crossed the seas, went on to discover the secrets of the far away lands and came back to tell stories, one more unbelievable than the other. You may be mostly homo

[3]Horace, *Epistles* 1.10.*24-25*

sapiens or you may be a pile of rusty metal modules; our story does not say. But you can sense your environment and you can talk. As part of a linguistic community, your means of expression are codified. Your language provides ways to generate and recognize well-formed sentences: atomic sentences are widely available, composite sentences are generated at will, their structure is clear and unambiguous. The stories you tell follow the official format; it is the only way you have to express yourself, to connect with your audience. In the official format for stories, all sentences are followed by a value, a truth value, e.g.,

The lord of the land is a two headed goat like creature : true

I lost (5^3 - 12^2 + 4) digits to the blistering cold : (false, impossible)

There are vast gold deposits right under the feet of the natives : hearsay

They have no clue about our land and our ways : unknown

Most natives are either goat like creatures or a cross between a frog and a lion : no evidence to the contrary

They know the fifth universal law : improbable (p=0.12)

It is as if an English speaker would feel (grammatically) obligated to say "It is true that the cat is on the mat" rather than "The cat is on the mat" to assert the fact that, well, the cat is on the mat. In Alice's land, we end up with stories composed of statements, of judgments, where truth values appear explicitly. The bewildering array of possible truth values is naturally codified by the language; there is an official set. How many truth values there are, what their names are, and how they are related, are all details specified somewhere, but of no concern to us here. Truth values in our setting are tags that one attaches to sentences; they are nothing more than the elements of some suitable sublanguage. Depending on the interpretation, the matter, at hand, they may be seen as linguistic elements expliciting the assertoric force of a statement, as traditional logical truth values, Bedeutungen à la Frege, elements of some arbitrary lattice, signs in a signed tableau expansion [8, 21], or as the possible displays of a sensor - to mention only a few possibilities.

Alice's language is sophisticated enough for sentences to come in two forms: atomic and composite. As part of the linguistic convention of the community, the truth value attached to a composite sentence is computed from the truth values of its atomic subsentences - a compositionality assumption. A native speaker has at least two competencies: she is able to generate well-formed sentences and she is able to correctly value composite sentences.

While on assignment, Alice records her observations in a log book, whose entries are of the form:

$atomic\, sentence\, \underline{e} : truth\, value\, b$ or $\underline{e} : b$ for short.

The log book records available evidences in the form of valued atomic sentences[4]. To produce a report, Alice uses the log book entries to compute composite sentence values as needed. Anybody having access to her log book would be able to reproduce her valuations because the way values are propagated from the atomic sentences is codified by the language, there is only one correct end

[4]Anticipating the mathematical representation, a log book is nothing but a partial atomic valuation.

result. Note that, in Alice's world, the value of a composite sentence is never observed, it is always computed, whereas the value of an atomic sentence is (mostly) observed, never computed[5].

I like to view the situation in terms of a run-of-the-mill cognitive agent. Let's downgrade Alice to a lowly robot status. The robot has sensors, a whole series of them; the sensors have names and display values. The log book is a long list of sensor readings recorded as:

sensor name $X(\underline{e})$: *value* b or $X(\underline{e}) : b$, $X_i : b$ for short

The raw data may be terabytes of values, billions of pixels, a level of details that no audience should have to bear. The data may be proprietary or the customer may be interested only in a few salient results. Instead of sharing its whole database, our little robot computes the value of some composite quantities of interest and communicates its findings:

$(X_1 + X_5 + X_8)/3 : b$

$F(X_4, X_8) : b'$

This is a report and this is how our little robot talks. The way the values are computed is known to all parties involved, the raw data may not, and they are, in general, not present at the report level.

Comments

(1) For Alice the lowly robot, it is presumably the environment, the world, reality, that dictates the values displayed by the sensors. A more capable cognitive agent such as Alice the explorer is free to attach any value she sees fit to an atomic sentence. Our linguistic community does not regulate how to valuate at the atomic level.

(2) How does Alice determine if an atomic sentence is true? That we do not say, because we do not have to. Our setting nicely dissociates truth determination and value propagation. How values propagate decides the properties of the local logic. The particular theory of truth the community may have chosen to valuate atomic sentences, the verification protocol, or the confirmation and falsification criteria adopted, does not affect the local logic - and even less the global one. Logic does not have to be a theory of truth or a theory of truth transmission. What truth or a truth value is, is not really that relevant to our reconstruction of logic. That values are propagated, i.e., the compositionality restriction, is the key assumption.

(3) Given a log book, the speaker is free to choose the sentences she wants in the report[6]. What she says, hides, emphasizes is hers to decide. The rest, however, is convention. The values attached to the sentences appearing in the report are fully determined by the log book used and the conventions of the language.

[5]In principle, not all atomic sentence values have to be observed. Alice may rely on some particular deductive systems or on some particular scientific theories to relate atomic sentence values. Such capabilities will however not be considered here.

[6]Would Alice be more of a database, she may alternatively limit herself to answer queries.

The listener perspective

Upon reception of the report, what should Bob, the listener, do? Read it, of course, and then as a competent and responsible member of the linguistic community, he may feel obligated to:

(1) determine the grammaticality of the report,

(2) determine the satisfiability of the report.

Are all sentences appearing in the report well-formed according to the official grammar of the language? Yes, proceed to (2). Are all values well computed? Log book *log* is said to satisfy report Σ if report Σ can be generated by a competent speaker starting from *log*. Given a particular *log*, there is at most one value that can be attached to a sentence e; either *log* does not provide enough information or the computation can proceed and will terminate. Report Σ is said to be satisfiable if at least one log book satisfies Σ.

Of course, Bob could judge the report relative to any set of criteria and challenge it on various grounds. He may believe the report. He may find it useful. He may appreciate the way the report was put together and the amount of work and skill that went into it. The typesetting or the cover may be particularly pleasing. He may remember the admonitions of Grandfather to never believe anything with an official stamp or, contrariwise, the cautionary tales of his teachers, to only trust refereed reports. Alice the explorer may be known in the community for her colorful and fanciful testimonies. Or he may simply ignore the report because its findings are inconvenient. Bob's state of mind, biases and prejudices are not of interest here. How Bob's predispositions, conception of the world, past experiences, or intentions may influence his opinion of Alice's work would be a fascinating digression, but well beyond the confines of our little analysis. We will not even allow Bob to challenge the entries of Alice's log book. He did not go with Alice on her far-reaching exploration of the world; he has no basis to challenge the veracity of her atomic statements, notwithstanding his preconceptions of how the world is or what the sensors should display. Our limited listener can check however if the report follows the conventions of the linguistic community and use in any way he sees fit the information disclosed by the report.

How should Bob determine satisfiability? The language does not say. Another linguistic community may provide rules to determine satisfiability but not this one. Bob is left to his sagacity and his understanding of how the language works. How Bob comes up with a log book that satisfies report Σ is up to him. Yet, once the claim is made that a particular *log* satisfies Σ, anybody can check the claim. It is a simple matter of value propagation. Bob's predicament is however far from hopeless, for his language is simple enough to ensure that satisfiability can always be determined by checking a finite number of log book candidates.

The local logician

Our basic setting has one last character: Uncle George, the logician. What is a logician you may ask? Speakers and listeners are a dime a dozen in any linguistic community. Who needs a logician to be a competent speaker or a good listener? As long as Alice can follow the linguistic conventions, her reports will be well-formed and well-valued. She can generate complex sentences at will, value them correctly, follow all the rules of the language, tell her stories, spin a tale as well as anybody. And with sufficient diligence and determination, Bob can check the grammaticality and satisfiability of any report.

Alice and Bob need a logician as much as they need a grammarian. For the sake of definitiveness, let's agree that a grammarian is someone who studies the grammatical features of naturally occurring languages and of not so naturally occurring languages, provides ways to approach such endeavors, and comes up with formal languages that have suitable properties including simple construction and recognizability. Likewise, a logician studies the logical properties of languages, provides ways to approach such endeavors, and comes up with formal languages that have suitable properties including easy determination of values and satisfiability[7].

Our local logician George would like to understand the language of his linguistic community. In particular, he would like to know:

(1) Which two sentences end up with the same value regardless of the log book used?

(2) Which two reports are equivalent in the sense that if one is satisfied by a log book the other is too?

Eventually, he wants to be able to:

(1) Streamline means of expressions and means of reporting by providing normal forms.

(2) Easily recognize tautological and contradictory reports.

(3) Provide improvements on the existing linguistic conventions.

George's working language is not going to be Alice's language. If he wants to talk about the logical properties of Alice's language, he needs a suitable medium, a metalanguage so to speak. George may find the use of a mathematical framework propitious. And so will we.

2 Logic at the sentence level

2.1 The means of expression

Without restricting ourselves much, let's assume, for definitiveness, that the language of our linguistic community is a many sorted term language L. What we have in mind is a formal language built from a set of atoms, $a \in Atoms(L)$, and a set of connectives, $c \in Conn(L)$, as is customary in logic, universal

[7]Any realistic account of what logicians do would have to mention 'validity of argument', 'logical consequence', 'proof theory', and so on. We have tailored the definition to our needs. Our approach is purely semantical.

algebra or programming language theory [13, 16, 7, 2, 18]. Connectives are typed, $c : (T_1, T_2, \ldots \to T_m)$, so are atomic terms, $(a) : T$.

New terms are built and types are propagated according to the grammar of the language, specified for example by some formation rules, one per connective:

$$\frac{e_1 : T_1,\ e_2 : T_2,\ \ldots,\ c : (T_1, T_2, \ldots \to T_m)}{(c\, e_1\, e_2 \ldots) : T_m}$$

The types of a many sorted language have typically no particular structure[8]. They are sorts, $T \in Sort(L)$. The formation rules ensure that all terms of the language are typed, and that the type associated with a well-formed term is a sort. By default, we will use the familiar S-expression format to display terms, e.g., $(c\, e_1\, (c'\, e_3\, e_2))$, allowing for infix, postfix, mixfix, or other variants.

We are not assuming the sorts to be distinct. It is perfectly fine to assume that all the T's are the same, would one want to work in a monosorted setting. In the case of a single sort language, the grammar of the language simplifies to a consideration of arities, e.g. in BNF style,

$e := (a) \,|\, (c_1\, e) \,|\, (c_2\, e\, e) \,|\, \ldots$

This is the kind of term language one typically encounters in the first few chapters of a logic textbook, e.g.,

$e := (a) \,|\, (\neg\, e) \,|\, (\wedge\, e\, e) \,|\, (\vee\, e\, e) \,|\, \ldots \,|\, (\Box\, e) \,|\, (\Diamond\, e)$

For predicate logic, it is more convenient to use a term language with at least two sorts, e.g., one sort for propositions and one for individuals.

The set of terms of a many sorted term language L is canonically endowed with a structure of term algebra. The elements of $Conn(L)$ are connectives, i.e., linguistic entities, not algebraic operations. Nonetheless, one may associate an algebraic operation, Γ_c, with each connective and view the formal language L as an algebraic system if so inclined. Let's denote by $L(T)$ the terms of L that have type T. And let's denote by $A(L)$ the term algebra associated with the formal language L, i.e., $(L(T),\ )_{T \in Sort(L)}\ \hat{}\ (\Gamma_c,\ )_{c \in Conn(L)}$[9].

If connective c has type $(T_1, T_2, \ldots, T_n \to T_m)$, then operation Γ_c takes in n arguments, the first one from set $L(T_1)$, the second from $L(T_2)$,.., the last one from $L(T_n)$, and returns a term from $L(T_m)$, or $\Gamma_c :\ L(T_1), L(T_2), \ldots, L(T_n) \to L(T_m)$ with

$\Gamma_c(e_1,\, e_2, \ldots,\, e_n) = (c\, e_1\, e_2 \ldots\, e_n)$

The operation is reversible, technically, Γ_c is injective.

Comments

(1) For us, a term is more a tree than a string. What its exact mathematical representation is in this mathematical framework depends on the details of the mathematical representation used, e.g., a tuple, a partial order, a graph, or some categorical theory construct. We omit the details; they play no role.

[8]The linguistic community may enjoy much richer languages and elaborate type systems. Except for the added technicalities and the more complex notation, our analysis unfolds in similar ways.

[9]The caret indicates concatenation of the two families. This is the operation one needs to recover a tuple, the usual notation for an algebraic system.

(2) Whenever convenient either technically or philosophically, we will assume that $Atoms(L)$ and $Conn(L)$ are finite. The reader could also assume that the complexity of the terms is limited in some ways (e.g., by requiring the height of a term to be less than 1000), allowing him or her to work with a finite corpus, but we will not. Our finitistic qualms are not that extreme.

(3) With these assumptions, the only source of infinity in this work is the ever-expanding formal language, a set in becoming if any.

2.2 H-interpretation and the propagation of values

Let's associate with each sort T a range of values, B. The set B may be a range of truth values[10]. It may also be understood as nothing more than the set of values a sensor may display. And with each connective $c : (T_1, T_2, \ldots \rightarrow T_m)$, let's associate an algebraic operation $H_c : (B_1, B_2, \ldots \rightarrow B_m)$ of matching type.

An assignment is an element of the cartesian product $(\times)_{a \in Atom(L)} B_a$. An assignment assigns to each atomic term (a) a value. Notation: the set of all assignments is $Asg(L, H)$, $Asg(L, H) = (\times)_{a \in Atom(L)} B_a$.

An assignment, let's say α, is a function defined on the atomic terms. We want to extend α to all the terms of the language. There are quite a few ways of getting there. The value associated with composite term $(c\, e_1\, e_2 \ldots)$ relative to assignment α may be computed by firing the computation rules that correspond to the formation rules of the language:

$$\frac{e_1 : b_1,\ e_2 : b_2,\ \ldots,\ c : (B_1, B_2, \ldots \rightarrow B_m)}{(c\, e_1\, e_2 \ldots) : H_c(b_1, b_2, ...)}$$

This is a bottom-up computation: the values are propagated from the leaves to the root.

More traditionally, let's define by structural recursion the value of term e at assignment α, $val(e, \alpha)$,

$val((a), \alpha) = \alpha(a)$

$val((c\, e_1\, e_2 \ldots), \alpha) = H_c(val(e_1, \alpha), val(e_2, \alpha), ...)$

Whenever convenient, we will lighten the notation to $val(\alpha)(e)$, $val_\alpha(e)$ or even $val(e)$. And extra sets of parenthesis will be elided, e.g., $val(a)$ rather than $val((a))$.

Comments

(1) All computations are finite even if the sets $Atom(L)$, $Conn(L)$, or B_T are not - because a term has only so many connectives.

(2) All computations are done within the algebraic system $H(L)$, i.e., within $(B_T,)_{T \in Sort(L)}\ \hat{}\ (H_c,)_{c \in Conn(L)}$.

(3) Technically, $val(\alpha)$ is a morphism of algebraic systems, $val(\alpha) : A(L) \rightarrow H(L)$. We could as well have defined $val(\alpha)$ as the morphism that canonically extends the function α:

[10] The elements of B may be seen as some kind of generalized bits, hence the adopted notation, $b \in B$.

$(val(\alpha) \circ \Gamma_c)(e_1, e_2, \ldots, e_n) = H_c(val(\alpha)(e_1), val(\alpha)(e_2), ...)$
or,
$val(\alpha) \circ \Gamma_c = H_c \circ val(\alpha)^{\otimes}$

(4) The computation of values does not have to be bottom-up. The values do not have to be propagated from the leaves to the root. One could easily switch to a top-down computational scheme, working for example with a stack of function calls or generating goals and subgoals.

(5) To keep the number of assignments finite and the operations H_c finitely presentable, one can always assume the sets B_T finite.

(6) The couple (L, H) may be seen as an interpreted language.

2.3 The equivalence of two terms

Terms e_1 and e_2 of L are said to be (L, H)-equivalent, $e_1 \simeq_{(L,H)} e_2$ or, for short, $e_1 \simeq e_2$, if the computation of values always ends up with $val_\alpha(e_1)$ and $val_\alpha(e_2)$ equal, i.e.,
$\forall\alpha,\ val_\alpha(e_1) = val_\alpha(e_2)$

With the sets B_T and $Atom(L)$ finite, the quantification is over a finite number of assignments. The binary relation $\simeq$ is an equivalence relation on terms compatible with the formation rules of the language, i.e., $\simeq$ is a congruence on $A(L)$. Terms e_1 and e_2 are equivalent if their associated functions, $val(e_1) : Asg(L, H) \rightarrow B$ and $val(e_2) : Asg(L, H) \rightarrow B$, are equal on $Asg(L, H)$, $val(e_1) = val(e_2)$. The quotient algebraic system $A(L)/\simeq$ is the Lindenbaum-Tarski algebra of the local logic. Note that $\simeq$ is typed.

The study of the properties of the congruence $\simeq$ is a study of the algebraic system $H(L)$, in particular, a study of its identities.

2.4 Polynomial terms

With any term e of L, one may associate a polynomial term $P(e)$, a polynomial[11] of $H(L)$:
$P(c\, e_1\, e_2 \ldots) = (H_c\, P(e_1)\, P(e_2) \ldots)$
$P(a) = (a)$

Note that the atoms of L play now the role of polynomial variables and the names of the operations of $H(L)$ are used as connectives - leaving as usual to the context the delicate task of disambiguating the various usages.

The value of polynomial term $P(e)$ at α, $val(P(e), \alpha)$, is defined as for the case of a general term:

$$\begin{aligned} val(P(c\, e_1\, e_2 \ldots), \alpha) =& val((H_c\, P(e_1)\, P(e_2) \ldots), \alpha) \\ =& H_c(val(P(e_1), \alpha),\ val(P(e_2), \alpha), \ldots) \\ val(P(a), \alpha) =& \alpha(a) \end{aligned}$$

[11]For us, a linguistic entity on a par with a term, not a function, though it is straightforward to view a polynomial as a function on assignments - see below.

where the interpretation of polynomial term connective H_c is the operation H_c. The function $P(e)$ associated with the polynomial term $P(e)$ is $P(e) : Asg(L, H) \rightarrow B_e$ where $P(e)(\alpha) = val(P(e), \alpha)$. Two polynomials are equivalent if they are equal when seen as functions. Note that $val(P(e), \alpha) = val(e, \alpha)$ or, as functions, $P(e) = val(e)$. (L, H)-computing with L-terms is computing with $H(L)$-polynomials. Algebraic equivalence of terms is functional equality of polynomials is equality of functions:

$e_1 \simeq e_2 \;\Leftrightarrow\; \forall\alpha,\, val_\alpha(e_1) = val_\alpha(e_2) \;\Leftrightarrow\; \forall\alpha,\, P(e_1)(\alpha) = P(e_2)(\alpha) \;\Leftrightarrow\; P(e_1) = P(e_2) \;\Leftrightarrow\; val(e_1) = val(e_2)$

Comment: Polynomial terms may be represented mathematically in many ways. We have adopted a version that fits our framework nicely. We could relate in a somewhat similar way $H(L)$-polynomials and $A(L)$-polynomials. If we were looking for logical forms, $A(L)$-polynomials would fit that role just fine, though adding variables to the language L would be more transparent.

2.5 Functionally complete interpretation

With any term e of L, one may thus associate a function on assignments, $val(e)$, $P(e) : Asg(L, H) \rightarrow B_e$, $\alpha \mapsto val(e, \alpha)$. The function val sends $Term(L)$, $Term(L) = \bigcup_{T \in Sort(L)} L(T)$, into $\bigcup_{T \in Sort(L)} (Asg(L, H) \rightarrow B_T)$. An interpretation H is said to be function on assignments complete, or functionally complete, if all functions on assignments, $f : Asg(L, H) \rightarrow B_T$, are representable by polynomial terms,

$\forall f : Asg(L, H) \rightarrow B_T,\ \exists e \in L(T),\ f = P(e)$

Functionally complete interpretations are naturally quite nice to work with; they provide maximal expressivity from a function on assignment point of view. There is no feature[12] the speaker cannot express.

Comment: The usual notion of functional completeness pertains to algebraic systems; all operations[13] on the carrier set are required to be representable as polynomial functions. The two notions are quite close though not identical.

2.6 Examples

2.6.1 Classical propositional logic, the bivalent case: Boolean valuations.

Consider the single sort term language L, with one unary and two binary connectives,

$e := (a) \mid (\neg\, e) \mid (\wedge\, e\, e) \mid (\vee\, e\, e)$,

and the classical algebraic system $\mathbb{B}$ or $(\{0, 1\}, \ominus, \otimes, \oplus)$. The operations of $\mathbb{B}$ are the customary Boolean operations: negation, product, and sum on two elements.

[12]'feature' as understood in data analysis, let's say.

[13]An n-ary operation may be seen as an unary function on n-ary tuples. Relative to a ranking of the atoms, a sufficiently long tuple is an assignment (modulo some typing constraints), $tuple \circ ranking = assignment$, and an assignment is a tuple of length $\#Atoms(L)$, $tuple = assignment \circ ranking^{(-1)}$.

Relative to $\mathbb{B}$, a connective c of L is said to be a $\otimes$-connective or a conjunction if $H_c = \otimes$, similarly, connective c is a $\oplus$-connective or a disjunction if $H_c = \oplus$ and a negation if $H_c = \ominus$. A language may have in general many conjunctions, disjunctions, and negations; it makes little difference to its logical properties.

With the habitual interpretations $H_\wedge = \otimes$, $H_\vee = \oplus$ and $H_\neg = \ominus$, one recovers the laws of classical propositional logic as algebraic equivalence of terms, e.g., $(e_1 \wedge e_2) \simeq (e_2 \wedge e_1)$, $(e \wedge e) \simeq e$, or $\neg\neg e \simeq e$.

The function $val(e)$ sends $Atoms(L) \to \mathbb{B}$ into $\mathbb{B}$, $val(e) : (Atoms(L) \to \mathbb{B}) \to \mathbb{B}$.

2.6.2 Propositional logic, the N-valued case: Post valuations

The language is the same, $e := (a) \,|\, (\neg\, e) \,|\, (\wedge\, e\, e) \,|\, (\vee\, e\, e)$, the interpretation is different [33]. Consider the algebraic system $(\{0, 1, \ldots, N-1\}, C, min, max)$, where $\{0, 1, \ldots, N-1\}$, $N \geq 1$, is ordered as usual, max and min are binary operations that pick respectively the maximum and the minimum of two elements and C is an unary operation that cycles through the elements:

$C(x) = (x+1)mod(N)$ or $C(0) = 1$, $C(1) = 2$,..., $C(N-1) = 0$

Take $H_\wedge = max$, $H_\vee = min$ and $H_\neg = C$.

Some laws of classical propositional logic are still present, e.g., $(e_1 \wedge e_2) \simeq (e_2 \wedge e_1)$ or $(e \wedge e) \simeq e$, some take a new form, e.g., $e \vee \neg e \vee \neg\neg e \vee \ldots \vee (\neg\neg \ldots \neg e) \simeq e' \vee \neg e' \vee \neg\neg e' \vee \ldots \vee (\neg\neg \ldots \neg e')$, some are no more, e.g., $e \wedge \neg e \simeq e' \wedge \neg e'$ does not correspond to any polynomial identity unless $N \leq 2$.

The function $val(e)$ sends $Atoms(L) \to N$ into N, $val(e) : (Atoms(L) \to N) \to N$.

2.6.3 Modal valuation, the Boolean algebra with operators version [6, 19]

Consider the term language,

$e := (a) \,|\, (\neg\, e) \,|\, (\wedge\, e\, e) \,|\, (\vee\, e\, e) \,|\, (\Box\, e\,) \,|\, (\Diamond\, e)$,

and a set W, the set of worlds, assumed finite for computational reasons. The power set of W is canonically endowed with a Boolean algebra structure. Let's add to that Boolean algebra two unary operations, $m : \wp(W) \to \wp(W)$ and $l : \wp(W) \to \wp(W)$, typically a closure operator and an interior operator. Relative to that algebraic system, the set of worlds associated with a modal sentence is defined recursively by:

$val((a), \alpha) = \alpha(a)$

$val((\neg\, e), \alpha) = W - val(e, \alpha)$

$val(e_1 \wedge e_2, \alpha) = val(e_1, \alpha) \cap val(e_2, \alpha)$

$val(e_1 \vee e_2, \alpha) = val(e_1, \alpha) \cup val(e_2, \alpha)$

$val((\Box\, e), \alpha) = l(val(e, \alpha))$

$val((\Diamond\, e), \alpha) = m(val(e, \alpha))$

The function $val(e)$ sends $Atoms(L) \to \wp(W)$ into $\wp(W)$, $val(e) : (Atoms(L) \to \wp(W)) \to \wp(W)$.

2.6.4 Predicate logic [19, 1]

Consider a set of variables, $Var = \{x, x', ...\}$, a language of formula,
$e := (a) \,|\, (\neg\, e) \,|\, (\wedge\, e\, e) \,|\, (\vee\, e\, e) \,|\, (\forall x\, e\,) \,|\, (\exists x\, e) \,|\, (\forall x'\, e\,) \,|\, (\exists x'\, e) \,|\, \ldots\,,$
and a set D, the domain of quantification. Consider the set of object assignments, $Var \rightarrow D$. We want to interpret a formula as a function, a function from the set of object assignments, $Var \rightarrow D$ to a set L. If the range L is the binary set $\{0, 1\}$, the interpretation is a characteristic function, hence a subset. Let's focus on the binary case, and, for computational reasons, let's assume that Var and D are finite.

The algebraic system of interest is the Boolean algebra with operators, $(\wp(Var \rightarrow D), \cap, \cup, \complement, m_x, l_x, m_{x'}, l_{x'}, \ldots)$ where the operations $m_x, l_x, m_{x'}, l_{x'}, \ldots$, are particular cylindrification operators.

...
$val((\forall x\, e\,), \alpha) = l_x(val(e, \alpha))$
$val(((\exists x\, e\,), \alpha) = m_x(val(e, \alpha))$
...

The function $val(e)$ sends $Atoms(L) \rightarrow ((Var \rightarrow D) \rightarrow L)$ into $((Var \rightarrow D) \rightarrow L)$, in the binary case, $Atoms(L) \rightarrow (\wp(Var \rightarrow D))$ into $\wp(Var \rightarrow D)$.

Comment: If we rank the variables, an assignment is also a tuple and, after removing the extraneous dimensions, we recover a traditional interpretation - the extension of a predicate as a set of tuples.

2.6.5 Functional Programming

A value can also be a term, in which case, a valuation becomes a term language translation, a function from terms to terms. The initial interpretation associates the operation Γ_c with the connective c, or $H_c = \Gamma_c$. The algebraic system $A(L)$ is the finest interpretation one may invoke. $A(L)$-valuations are term substitutions.

The transformation of expressions is functional programming forte. For example, working with S-numerals, a coarse but useful characterization of a term is its connective height,
$val(a) = 0$
$val(c\, e_1\, e_2\, \ldots) = 1 + max(val(e_1), val(e_2), \ldots)$

The connective structure of a term is already a less coarse characterization, it erases all information at the leaves, leaving only the connective skeleton,
$val(a) = (0)$
$val(c\, e_1\, e_2\, \ldots) = \Gamma_c(val(e_1), val(e_2), \ldots)$
where 0 is some distinguished atom. The connective structure of a term is a term language transliteration, a particularly transparent valuation. Many transformations of expressions in functional programming are straightforward valuations.

The function $val(e)$ sends $Atoms(L) \rightarrow Terms(L')$ into $Terms(L')$, $val(e) : (Atoms(L) \rightarrow Terms(L')) \rightarrow Terms(L')$.

2.6.6 Typing as valuation [26, 3]

In a typed language, types may be, among other things, propagated or inferred.

Example: Applicative language, i.e., a term language with a single binary connective (.),

$val(\cdot\, e_1\, e_2) = val(e_1) * val(e_2) = (A \to B) * A = B$

An assignment is now a typing assignment and propagation of values is propagation of types [14]. The closeness of typing and valuation is naturally not fortuitous. They both propagate, going up the parse tree. A type and a value[15] can both be seen as the coarsening of a term: a type is also a collection of its inhabitants and a value is, in the preimage sense, a collection of terms. The move is the same: a partitioning of a set of terms resulting in a coarser picture and a loss of information.

2.6.7 Statistical reduction of experimental data

Consider a language of arithmetical expressions

$e := (X_i) \mid (plus\, e\, e) \mid (times\, e\, e) \mid ((1/n)\, e\,)$

and the field of rational numbers as intended interpretation. The valuation scheme is naturally

$val((X_i), \alpha) = \alpha(X_i)$

$val((plus\, e_1\, e_2), \alpha) = val(e_1, \alpha) + val(e_2, \alpha)$

$val((times\, e_1\, e_2), \alpha) = val(e_1, \alpha) * val(e_2, \alpha)$

$val(((1/n)\, e\,), \alpha) = (1/n) * val(e, \alpha)$

A report may now includes a variety of statistical quantities, exploratory data analyses, estimators, e.g.,

$val((mean\, X_1\, X_2\, X_3\, X_4\,), \alpha) = (1/4)*(val(X_1, \alpha)+val(X_2, \alpha)+val(X_3, \alpha)+ val(X_4, \alpha))$

Naturally, the statistical reduction of a set of data or data analysis is not traditionally within the province of logic, but from our point of view we are merely propagating values. Broadly speaking, the construction of a report is nothing more than a repackaging of information.

2.7 Conclusion

(1) The linguistic community settles on a many sorted language and an interpretation. The elements of the language are seen in the light of their algebraic denotata. A term of the language is reduced to a (polynomial) function on assignments; two terms are equivalent, $e_1 \simeq e_2$, if their associated functions are equal. The congruence $\simeq$ tells the speaker which two sentences have the same meaning, i.e., play the same role within the interpreted language, within a report, and opens the door to a normalization of the means of expression, e.g., sum of products in classical propositional logic.

[14]And type inference is satisfiability determination.

[15]'value' as used in this paper, not as used in type theory contexts.

The (L, H)-equivalence of two sentences is a key notion in classical logic. The congruence is fully determined by the choice of an interpretation, i.e., by the algebraic system $H(L)$. And that choice is to a large extent arbitrary. It is an agreement, a convention of the linguistic community. Logical results as $e_1 \simeq e_2$ are relative to the choice made.

(2) Modern logic quickly moves from computation within a single algebraic system to the representation of equivalence relations, consequence relations, and classes of sentences by classes of algebraic systems, in general, by classes of models.

$e_1 \simeq_{(L,\mathcal{H})} e_2 \quad \Leftrightarrow \quad \forall H \in \mathcal{H}, e_1 \simeq_{(L,H)} e_2$

If the class of algebraic systems $\mathcal{H}$ is equationally defined, we get a particularly nice congruence $\simeq_{(L,\mathcal{H})}$. Alternatively, the linguistic community may specify the operative congruence by listing a few primitive identities or the operative consequence relation by listing a few sequents and sequent operations. Mathematically, the way to go, but our linguistic community just want to compute. They have no particular congruence, consequence relation, or set of sentences to represent. They just have a single algebraic system relative to which they compute the values of composite sentences.

(3) The congruences $\simeq_{(L,H)}$ and $\simeq_{(L,\mathcal{H})}$ are relations on sentences, statements have not yet entered the fray.

(4) There is room to argue that some algebraic systems are natural choices[16], that some operations are logical and some are not. Where should the line between logical values and non-logical values be drawn, if at all? Where should the line between logical operations and non-logical operations be drawn, if at all? Criteria of logicality vary [22, 30, 32, 31]. We have no need for a demarcation line here; logicality criteria will not be discussed much further - see however section 7.4: *Distinguished value reexpression.* Not to mention, once we have the two statements connectives 'and' and 'or', sentence connectives become redundant - see for example section 7.1: *An accounting of possibilities,* there are no sentence connectives in a list of alternatives.

(5) At this point, a many-valued logic framework would introduce a set of designated values and a notion of validity, logical entailment, and logical equivalence, relative to that set [20, 24]. Alternatively, it may consider an order on the values, e.g., $H(L)$ as a lattice, and compare functions, in particular polynomial functions, hence sentences, using that order [12]. Either way, one recovers a logical formalism. We are not taking these roads. A listener needs to be able to assess satisfiability. A study of the satisfiability of reports is what we are after; a welcome outcome is yet another instance of a logical formalism.

A focus on information transfer, an assumption of compositionality, and a distinction between means of expression and means of assertion, gives us a plurality of logics, a plurality of logical formalisms. In this paper, we have emphasized two of them for they are, in the context of the paper, the most natural to us. But others are possible and are common place; logical entailment

[16]Functionally complete algebraic systems are particularly interesting as far as expressivity is concerned

constructed relative to a set of distinguished values is a perfectly adequate instance of logical entailment. How these different versions of logical entailment are related is discussed in section 7.6: *Logical notions by hook or by crook.*

(6) Diagrams and functors. Although we have developed the theory along traditional lines using algebraic systems and morphisms, this mathematical formalism does not really do justice to the simplicity of the situation. We really have only three structures - a term language, a term algebra, and an algebraic system - and a morphism. Terms upon substitutions move from one structure to another. A computer science perspective[17] or a category theory perspective may help: a term is a diagram with a tree structure, its leaves labeled by elements of $Atom(L)$ and its internal nodes labeled by connectives. It is transparent to replace the content of the nodes and computation is propagation along edges. A polynomial term is now nothing more than the connectives replaced by operation names; a polynomial term may be seen as a kind of partial computation. Such a framework clearly separates form and content, and would be a good fit for a streamlined and more visual presentation of logic at the sentence level.

3 Statements

3.1 Weighted terms, log books, reports

Let's forget about sentences and structure for a while and let's consider statements and reports[18]. Hereafter, we will represent statements mathematically by couples, by elements of $\bigcup_{T \in Sort(L)} L(T) \times B_T$, or $ST(L,H)$. A statement is a sentence with a value, sterm for short, that we will display as '(e,b)', as '$e : b$' or equivalent.

Notation: If $s = (e,b)$, then its term is $trm(s) = e$ and its value is $wgt(s) = b$.

A log book is a partial assignment: a set of sterms is a log book if all its terms are atomic and a given atomic term appears at most once. A set of sterms is a report if a given term appears at most once - nothing more than a functional set of couples. Let's denote the set of all reports by $Rep(L,H)$. One may view a report as a partial function, a partial element of $(\times)_{e \in Term(L)} B_e$, i.e., an element of $(\times)_{e \in L'} B_e$ for some $L' \subseteq Term(L)$. $Rep(L,H)$ is partially ordered by inclusion and is closed under intersection

Mathematically, we are moving away from a study of algebraic systems such as $A(L)$ and $H(L)$ and toward a study of posets. The vocabulary, tools and imagery will shift accordingly.

For simplicity, let's start by assuming that log books and reports are finite.

[17]Abstract datatypes, functional languages, semantics of programming languages.

[18]Actually, we want to be able to work at both levels: sentences for compositionality and statements for satisfiability.

3.2 The satisfiability problem

Given an assignment α and a few terms, it is a simple matter of computation to produce a report. Consider an assignment α and a set of terms L', the report obtained by propagating the values according to the interpretation H will be denoted by $rep_H(L', \alpha)$ or variants thereof where

$rep_H(L', \alpha) = \{(e, val(e, \alpha)) \,|\, e \in L'\} = val(\alpha)_{/L'}$

Propagating values is simple; the converse is, however, less so[19]. Consider a report Σ and its associated set of terms $trm(\Sigma)$ or L', to solve the equation $rep(L', \alpha) = \Sigma$ for α is the satisfiability problem[20]. We'll say assignment α satisfies report Σ, $Sat(\alpha, \Sigma)$, if α is a solution of the equation.

More terminology. Let's consider all the assignments that satisfy a report, $Asg(\Sigma)$ or $sol(\Sigma)$,

$Asg(\Sigma) = sol(\Sigma) = \{\alpha |\, rep(trm(\Sigma), \alpha) = \Sigma\}$

If $Asg(\Sigma)$ is empty, we say that Σ is unsatisfiable, if not, Σ is satisfiable. If all assignments satisfy Σ, the report is tautologous.

The satisfiability problem is a constraint satisfaction problem. A report plays the role of a set of constraints, and a satisfying assignment is a solution to the constraints. We have a set of trees with values at their tips, and we inquire about values at the leaves that upon propagation could reproduce the values at the tips.

How do we tackle the satisfaction problem?

1. We could solve one tree at a time and collect the common solutions, $Asg(\Sigma) = \bigcap_{s \in \Sigma} Asg(s)$ - a divide and conquer approach.

2. We could go through all the assignments, one assignment at a time, compute the values at the tips, and compare with the constraints - an exhaustive search algorithm.

3. We could transform the set of constraints, possibly one constraint at a time, into an equivalent problem easier to solve - the motivation behind a tableau expansion, see below.

There are many ways to solve a set of constraints. Is there an optimal way? What is a good quantification of the quality of an approach? How do we compare two approaches? This is a theory of algorithms. Computability theory and modern logic share many issues and tools. We will not concern ourselves too much with such computability issues however, our focus is on satisfiability per se, not satisfiability optimality.

Note that solving constraints is solving polynomial equations (in algebraic system $H(L)$),

$val(e, \alpha) = b \;\Leftrightarrow\; P(e)(\alpha) = b$

If the mathematical reader recognizes a few algebraic geometric notions here and there, it should be no surprise, for we are, at the end of the day, dealing with nothing more than solutions of polynomial equations.

[19]In the same way that solving a polynomial equation is less direct than computing the value of a polynomial at a given assignment.

[20]We will not dwell here on the important difference between proving the existence of a solution and constructing a solution. And we will not mention the $P \neq NP$ angle either.

3.3 Some properties of *sol*

We are collecting here a few properties of the function $sol : Rep(L, H) \to \wp(Asg(L, H))$. The reader familiar with the theory of Galois connections and its use in logic can safely skip this section and the next.

As a function, *sol* associates set of assignments with set of sterms, $sol : Rep(L, H) \to \wp(Asg(L, H))$.[21]

1. *sol* is a map between two posets.

2. Less constraints, more solutions, *sol* is antitone
$\Sigma_1 \subseteq \Sigma_2 \;\Rightarrow\; sol(\Sigma_1) \supseteq sol(\Sigma_2)$

3. The solutions of $\Sigma_1 \cup \Sigma_2$ are the solutions common to Σ_1 and to Σ_2,
$sol(\Sigma_1 \cup \Sigma_2) = sol(\Sigma_1) \cap sol(\Sigma_2)$

4. And since a report is the union of singletons, we recover
$sol(\Sigma) = \bigcap_{s\in\Sigma} sol(s)$

5. The assignment α is also a report and $sol(\alpha) = \{\alpha\}$. Singleton solutions are representable in the world of reports. The situation is however different for sets of solutions, e.g., $\{\alpha, \alpha', \alpha''\}$, and that is the crux of the matter (see maximal expressivity section).

6. A minimal report - in the sense of inclusion and ignoring the empty report, a lattice atom- is a singleton $\{s\}$ or $\{e : b\}$, a minimal report with atomic term would be a singleton $\{(a) : b\}$ and $sol(\{(a) : b\}) = \{\alpha \mid \alpha(a) = b\}$, i.e., a cylindrical set of solutions.

7. An empty report does not say much, $sol(\emptyset) = Asg(L, H)$, all assignments remain possible.

8. The maximal reports - in the sense of inclusion - are the infinite reports[22] $rep(L, \alpha)$, one per α, $rep(L, \alpha) = val(\alpha)$. Note that an infinite report such as $rep(L, \alpha)$ says the same thing as the finite report α, $sol(\alpha) = sol(rep(L, \alpha)) = \{\alpha\}$

9. *sol* is an antitone semi-lattice morphism,
$sol : (Rep(L, H), \cup, \emptyset) \to (\wp(Asg(L, H)), \cap, Asg(L, H))$

10. At some point, the finiteness restriction on reports becomes a liability. It is true that Alice will never produce more than a finite number of statements and restricting oneself to finite sets is philosophically appealing, but mathematically the restriction is awkward. The set of sterms $rep(L, \alpha)$ is not a report, though it should, and if we want to work with families of reports, e.g., $sol(\bigcup_i \Sigma_i) = \bigcap_i sol(\Sigma_i)$, the set of sterms $\bigcup_i \Sigma_i$ is not necessarily a report either per our present definition. To extend the function *sol* from $Rep(L, H)$ to $\wp(ST(L, H))$, from finite reports to arbitrary set of sterms, is not mathematically difficult: we just need to agree that for any set of sterms S but the empty one, $sol(S) = \bigcap_{s\in S} sol(s)$, and for the empty set, $sol(\emptyset) = Asg(L, H)$. We loose the computational angle but gain in mathematical convenience.

11. Let's work with arbitrary set of sterms.

[21] We may view an assignment as a report hence $sol(\Sigma)$ as a subset of $Rep(L, H)$: reduced to its bare bones, *sol* is a function from a set S to power set $\wp(S')$, $S' \subseteq S$.

[22] An infinite report is not per our present finiteness restrictions a report - see below however.

12. If $sol(S) = \emptyset$, S is said to be unsatisfiable, otherwise satisfiable. One expects most sets, finite or infinite, to be unsatisfiable.

13. The set of all solutions sets has a simple structure and that is a key point: all solution sets are generated by generalized intersection from the solution sets of single statements $sol(s)$, $s \in ST(L,H)$, except possibly the full solution set $Asg(L,H)$, since, by definition, $sol(S) = \bigcap_{s\in S} sol(s)$.

14. If $Asg(L,H)$ is finite, the situation is even simpler: the set of all solution sets is also finite.

3.4 The Galois connection between solutions and constraints

The binary relation 'assignment $\breve{\alpha}$ satisfies statement $\breve{s}$', or Sat, generates a Galois connection[23] [5, 14, 10]. In term of Formal Concept Analysis [17], we have an incidence on $Asg \times ST$. The associated polarities are:

$sol : ST \to \wp(Asg)$, $sol(s) = \{\alpha |\, Sat(\alpha, s)\}$

$ctr : Asg \to \wp(ST)$, $ctr(\alpha) = \{s |\, Sat(\alpha, s)\}$

and are suitably extended to sets of statements and sets of assignments:

$sol(S) = \{\alpha |\, \forall s \in S,\, Sat(\alpha, s)\} = \bigcap_{s\in S} sol(s)$

$ctr(A) = \{s |\, \forall \alpha \in A,\, Sat(\alpha, s)\} = \bigcap_{\alpha\in A} ctr(\alpha)$

sol collects the solutions to a set of constraints and dually *ctr* collects the constraints that are satisfied by a set of assignments. Notation: $solIm = \{sol(S) |\, S \subseteq ST\}$ and $ctrIm = \{ctr(A) |\, A \subseteq Asg\}$. The collection of statements $ctr(A)$ is the theory of the set A.

1. The power sets $\wp(Asg)$ and $\wp(ST)$ are Boolean algebras, posets under inclusion.

2. *sol* and *ctr* are antitone

3. If one defines $Sat(A, S)$ as $\forall(\alpha, s) \in A \times S$, $Sat(\alpha, s)$, we recover[24] the usual characterization of an antitone Galois connection.

$A \subseteq sol(S) \;\Leftrightarrow\; Sat(A, S) \;\Leftrightarrow\; A \times S \subseteq Sat \;\Leftrightarrow\; S \subseteq ctr(A)$

4. $S \subseteq (ctr \circ sol)(S)$ and $A \subseteq (sol \circ ctr)(A)$

5. $(ctr \circ sol)(ctr(A)) = ctr(A)$ and $(sol \circ ctr)(sol(S)) = sol(S)$

6. The operators $\langle\rangle_H : \wp(ST) \to ctrIm$ with $\langle S\rangle_H = (ctr \circ sol)(S)$ and $\langle\rangle_H : \wp(Asg) \to solIm$ with $\langle A\rangle_H = (sol \circ ctr)(A)$ are expansive, monotone and idempotent. They are closure operators.

7. The sets of images $solIm$ and $ctrIm$ are closure systems on Asg and ST respectively - from $Asg = sol(\emptyset)$ and $ST = ctr(\emptyset)$ and closure under intersection,

$\bigcap_{i\in I} sol(S_i) = sol(\bigcup_{i\in I} S_i)$

$\bigcap_{i\in I} ctr(A_i) = ctr(\bigcup_{i\in I} A_i)$

8. Comparison of closure operators:

$\langle S\rangle_H = (ctr \circ sol)(S) = \bigcap\{X \in ctrIm \,|\, S \subseteq X\}$

$\langle A\rangle_H = (sol \circ ctr)(A) = \bigcap\{X \in solIm \,|\, A \subseteq X\}$

9. The sets of images $\{sol(S) |\, S \subseteq ST\}$ and $\{ctr(A) |\, A \subseteq Asg\}$ are complete lattices under

[23]There are many Galois connections in Logic.

[24]Proof: An exercise in quantifier permutation, $\forall \alpha \in A$, $\forall s \in S, ... \Leftrightarrow \forall(\alpha, s) \in A \times S, ... \Leftrightarrow \forall s \in S,\ \forall \alpha \in A,\ ...$

$\bigwedge_{i\in I} sol(S_i) = \bigcap_{i\in I} sol(S_i)$
$\bigvee_{i\in I} sol(S_i) = \bigcap\{X \in solIm |\, \forall i \in I,\, sol(S_i) \subseteq X\} = (sol{\circ}ctr)(\bigcup_{i\in I} sol(S_i))$
$\bigwedge_{i\in I} ctr(A_i) = \bigcap_{i\in I} ctr(A_i)$
$\bigvee_{i\in I} ctr(A_i) = \bigcap\{X \in ctrIm |\, \forall i \in I,\, ctr(A_i) \subseteq X\} = (ctr{\circ}sol)(\bigcup_{i\in I} ctr(A_i))$

10. The sets of images $\{sol(S) |\, S \subseteq ST\}$ and $\{ctr(A) |\, A \subseteq Asg\}$ are isomorphic as complete lattices[25] with $ctr : solIm \to ctrIm$ and $sol : ctrIm \to solIm$ as morphism. From,
$\bigcap_{i\in I} sol(S_i) = sol(\bigcup_{i\in I} S_i) = sol \circ ctr \circ sol(\bigcup_{i\in I} S_i) = sol(\bigvee_{i\in I} S_i)$
hence
$sol(\bigvee_{i\in I} ctr(A_i)) = \bigwedge_{i\in I} sol(ctr(A_i))$
and
$\bigvee_{i\in I} sol(ctr(A_i)) = sol{\circ}ctr(\bigcup_{i\in I} sol(ctr(A_i))\,) = sol(\bigcap_{i\in I}(ctr{\circ}sol{\circ}ctr(A_i))\,)$
hence
$sol(\bigwedge_{i\in I} ctr(A_i)) = \bigvee_{i\in I} sol(ctr(A_i))$

11. A formal concept is a couple $(sol(S),\, ctr(sol(S))\,)$, alternatively $(sol(ctr(A)),\, ctr(A)\,)$. The set of all formal concepts, $Concept(L, H)$, is canonically endowed with a complete lattice structure.

12. The lattice of formal concepts, the lattice of solution sets, and the lattice of *ctr*-images are isomorphic as complete lattices.

Galois connection wise, a report is a set of solutions. a set of solution is a theory, and a theory is a set of solutions. The Galois connection gives us a systematic way of transferring constructs and problems back and forth between $Asg(L, H)$ and $ST(L, H)$, providing a bridge between semantics and syntax, as it is put. In this work, the world of solution sets is particularly simple, all sets are finite and easily presentable. The world of reports, on the other hand, is already less giving: the number of reports is infinite and sets of statements of interest are typically non finite. Not surprisingly, we find ourselves working first and foremost with solution sets and transferring structures as needed from the world of assignments to the world of reports.

Comment: Though we are working with the particular binary relation Sat, the constructions and results of this section are highly generic. They apply ad verbatim to any binary relation[26] R, $R \subseteq M \times C$. They do not belong per se to a theory of satisfiability, and one should therefore not expect a theory of satisfiability to be limited to Galois connection results.

3.5 (L, H)-theories

A (L, H)-theory is simply a *ctr* image, i.e., an element of $\{ctr(A) |\, A \subseteq Asg\}$. Note that $ctr(\alpha)$ is $rep(L, \alpha)$ hence $val(\alpha)$.
$ctr(A) = \bigcap_{\alpha\in A} ctr(\alpha) = \bigcap_{\alpha\in A} rep(L, \alpha) = \bigcap_{\alpha\in A} val(\alpha)$

Theories are generated from the maximal reports by generalized intersection, except for the trivial theory $ctr(\emptyset)$, in exactly the same way that solutions

[25]Technically, modulo a lattice dual.

[26]For example, if we change the binary relation 'assignment $\breve{\alpha}$ satisfies statement $\breve{s}$' to 'the value of sentence $\breve{e}$ at assignment $\breve{\alpha}$ is a designated value' , or $val(\breve{e}, \breve{\alpha}) \in D$, D a set of value, one per sort, we get another interesting Galois connection.

sets are generated by generalized intersection from the solution sets of single statements $sol(s)$, $s \in ST(L,H)$.

1. $ctr(\emptyset) = ST$, the top theory is unsatisfiable and it is the only one to be so.
2. If $A \neq \emptyset$ then $sol(ctr(A)) = \langle A \rangle_H = sol(ctr(\langle A \rangle_H))$ and $\langle A \rangle_H \neq \emptyset$
3. The smallest theory is $ctr(Asg)$; it collects all the tautologies of the language (L,H) and is included in all other theories.
4. Theories are in general infinite sets by contrast to solution sets.
5. Membership is nonetheless easily determined. To assess if (e,b) is element of the set $rep(L,\alpha)$, compute $val(e,\alpha)$ and compare with b:
$(e,b) \in rep(L,\alpha) \;\Leftrightarrow\; val(e,\alpha) = b$
6. Notation: $rep(L,A) = ctr(A) = \bigcap_{\alpha \in A} rep(L,\alpha)$
7. $sol(rep(L,A)) = \langle A \rangle_H = \bigcap_{s \in rep(L,A)} sol(s)$

There are many sets of interest in $ST(L,H)$ besides finite sets and (L,H)-theories. For example, a set of sterms may be functional, closed under various algebraic operations, closed under various co-algebraic operations, or representable as an axiomatic system.

3.5.1 Functional sets of sterms

The set of sterms S is said to be functional, $S \in PF(L,H)$, if:
$\forall e, b, b',\ (e : b) \in S,\ (e : b') \in S \;\Rightarrow\; b = b'$

$PF(L,H)$ is essentially collecting all partial functions. A report is a functional set of sterms, $Rep(L,H) \subseteq PF(L,H) \subseteq \wp(ST(L,H))$. The system of sets $PF(L,H)$ is ordered by inclusion and closed under generalized intersection. It is a closure system. The functional set S is maximal if all terms e have a value. A maximal theory is therefore also a maximal functional set, a function. All theories but the trivial one are functional; most functional sets are not satisfiable.

3.5.2 Sets of sterms closed under the computation rules

The set of sterms S is said to be closed under computational rule r, where r is:

$$\frac{e_1 : b_1,\ e_2 : b_2,\ \ldots,\ c : (B_1, B_2, \ldots \to B_m)}{(c\, e_1\, e_2 \ldots) : H_c(b_1, b_2, \ldots)}$$

if
$\forall e_1, b_1, e_2, b_2, \ldots,\ (e_1 : b_1) \in S,\ (e_2 : b_2) \in S, \ldots$
$\Rightarrow\ ((c\, e_1\, e_2 \ldots) : H_c(b_1, b_2, \ldots)) \in S$

A (L,H)-theory is closed under all the computational rules of the language - opening the door to an algebraic approach to the study of theories.

3.5.3 Sets of sterms closed under decomposition (e.g. Hintikka sets)

The set of sterms S is said to be closed under c-decomposition, where c is a connective of the language, if
$\forall e_1, e_2, \ldots, \forall b,\ ((c\, e_1\, e_2 \ldots) : b) \in S$
$\Rightarrow\ \exists b_1, b_2, \ldots,\ (e_1 : b_1) \in S \wedge (e_2 : b_2) \in S \wedge \ldots \wedge H_c(b_1, b_2, ...) = b$

The trivial theory is closed under c-decomposition, the maximal theories are closed but one cannot expect all (L, H)-theories to be closed under c-decomposition.

Example: Consider $val((c\,e), \alpha_1) = b$ and $val((c\,e), \alpha_2) = b$ or $((c\,e) : b) \in rep(L, \{\alpha_1, \alpha_2\})$. It is perfectly possible that $val(e, \alpha_1) = b_1$, $val(e, \alpha_1) = b_2$, $H_c(b_1) = b$, $H_c(b_2) = b$, and $b_1 \neq b_2$.

4 Logic at the statement level

Satisfiability considerations allow us to recover a very familiar looking logical formalism.

4.1 Entailment or (L,H)-implication

We say report Σ_1 (L, H)-implies report Σ_2 or $\Sigma_1 \models_{(L,H)} \Sigma_2$ if $sol(\Sigma_1) \subseteq sol(\Sigma_2)$, i.e., an assignment that satisfies Σ_1 will satisfy Σ_2, a solution of Σ_1 is a solution of Σ_2. By reporting Σ_2, you are not reporting anything new. (L, H)-implication is entailment or logical implication from a semantics point of view as usually understood:

> "Generalized Tarski Thesis (GTT): An argument is valid$_\mathrm{x}$ if and only if in every case$_\mathrm{x}$ in which the premises are true, so is the conclusion."[4, p 29]

We carry along a reference to the language L and the interpretation H to belabor the point we are trying to make: logical results are relative, they depend on the composite terms we have and the conventions we use to value them. For example,
$\{e_1 : b_1,\ e_2 : b_2,\ \ldots,\} \models_{(L,H)} \{(c\, e_1\, e_2 \ldots) : H_c(b_1, b_2, ...)\}$

The binary relation $\models_{(L,H)}$ is set inclusion[27] on $\wp(Asg(L, H))$ transferred to $Rep(L, H)$. It is a preorder[28] and has the properties[29] one expects [27, 15, 23], including:

1. Reflexivity: $\Sigma \models \Sigma$
2. Transitivity :$\Sigma_1 \models \Sigma_2$, $\Sigma_2 \models \Sigma_3 \Rightarrow \Sigma_1 \models \Sigma_3$
3. Inclusion to entailment is antitone: $\Sigma_1 \subseteq \Sigma_2 \Rightarrow \Sigma_2 \models \Sigma_1$
4. Premiss dilution: $\Sigma_1 \subseteq \Sigma_3$, $\Sigma_1 \models \Sigma_2 \Rightarrow \Sigma_3 \models \Sigma_2$

[27] More to the point, set inclusion on the lattice of solution sets, $solIm$.

[28] Note that we now have two preorders on $Rep(L, H)$.

[29] The binary relation $\models_{(L,H)}$ may be highly dependent on L and H, but the properties of $\models_{(L,H)}$, the ones we are typically interested by, are, on the other hand, quite generic.

5. Conclusion pruning[30]: $\Sigma_4 \subseteq \Sigma_2,\ \Sigma_1 \models \Sigma_2 \Rightarrow \Sigma_1 \models \Sigma_4$

6. Cut[31]: $\Sigma_1 \models \Sigma_2,\ \Sigma_3 \models \Sigma_4 \Rightarrow ((\Sigma_3 - \Sigma_2) \cup \Sigma_1) \models \Sigma_4$

7. In terms of closure, $\Sigma_1 \models_H \Sigma_2 \;\Leftrightarrow\; \Sigma_2 \subseteq \langle \Sigma_1 \rangle_H$

8. (L,H)-implication can be extended without difficulties to arbitrary set of sterms:

$S_1 \models_H S_2 \;\Leftrightarrow\; sol(S_1) \subseteq\; sol(S_2) \Leftrightarrow\; S_2 \subseteq \langle S_1 \rangle_H$

4.2 (L,H)-equivalence

Report Σ_1 is (L,H)-equivalent to report Σ_2 if $\Sigma_1 \models \Sigma_2$ and $\Sigma_2 \models \Sigma_1$ or $sol(\Sigma_1) = sol(\Sigma_2)$, notation $\Sigma_1 \equiv_{(L,H)} \Sigma_2$. The binary relation $\equiv_{(L,H)}$ is set equality on $\wp(Asg(L,H))$ transferred to $Rep(L,H)$. Two reports are (L,H)-equivalent if they are satisfied by the same assignments. This equivalence is central to logic, more so than the equivalence of terms we would argue, though the two equivalences are naturally related,

$e_1 \simeq e_2 \;\Leftrightarrow\; \forall b,\ (e_1 : b) \equiv (e_2 : b)$

The relation $\equiv_{(L,H)}$ determines the reports that tell the same story, allowing the speaker to choose a formulation that suits her tastes or goals and providing ways for the listener to reduce the report to a normal form.

The local logician ends up with a finite partition - would $Asg(L,H)$ be finite - and a naturally occurring equational system to sort out. With a finite number of assignments, let's say N, there are only so many different stories that can be told, at most 2^N in general, exactly 2^N if the range of sol is $\wp(Asg(L,H))$ itself.

4.3 (L,H)-consequence

Sterm s is said to be a (L,H)-consequence of report Σ, $(s = \,|\, \Sigma)$, or, in topological term, s is near the set Σ, if $\Sigma \models \{s\}$ - an assignment that satisfies Σ will satisfy s. By adding consequence s to the report, nothing more is said. Mathematically, logical consequence can be seen as an extension or a coarsening of membership:

$s \in \Sigma \;\Rightarrow\; s = \,|\, \Sigma$

$(s = \,|\, \Sigma) \;\Leftrightarrow\; \Sigma \models_H \{s\} \;\Leftrightarrow\; s \in \langle \Sigma \rangle_H \;\Leftrightarrow\; s \in_H \Sigma$

and is nothing more than a specialization of (L,H)-implication to a singleton conclusion. Properties of logical nearness include:

1. Explosion: any sterm follow from an unsatisfiable report,

$\Sigma \notin SatRep \Rightarrow (\Sigma \models s)$

2. A tautological sterm is a consequence of any report:

$\{s\} \in TautRep \Rightarrow \forall \Sigma,\ (\Sigma \models s)$

3. Logical implication from logical nearness:

$(\forall s \in \Sigma_2,\ \Sigma_1 \models s) \;\Leftrightarrow\; \Sigma_1 \models \Sigma_2$

4. If we collect all the consequences of a report Σ, what do we get? The set of sterms $\{s \,|\, \Sigma \models_H s\}$ which is, not surprisingly, a theory:

[30] Proofs 2-5: Inclusion is transitive on $\wp(Asg(L,H))$.

[31] Proof: From $\alpha \in sol((\Sigma_3 - \Sigma_2) \cup \Sigma_1)$ infer $\alpha \in sol(\Sigma_1)$, $\alpha \in sol(\Sigma_2)$, $\alpha \in sol(\Sigma_3 - \Sigma_2)$, $\alpha \in sol(\Sigma_3)$, $\alpha \in sol(\Sigma_4)$

$\{s \mid \Sigma \models_H s\} = \langle \Sigma \rangle_H = \bigcap_{\alpha \in sol(\Sigma)} rep(L, \alpha)$

5 Pre-image analysis

So far, we have relied on the Galois decomposition $sol(\Sigma) = \bigcap_{s \in \Sigma} sol(s)$ to simplify matters; the set Σ is reduced to statements, and this is how far the Galois connection goes. We would like to go further, to be able to decompose $sol(s)$, whenever $s = (e : b)$ with e composite.

Question: When $e = (c\, e_1\, e_2..)$, how does $sol(e : b)$ depends on $sol(e_1 : b_1)$, $sol(e_2 : b_2)$, ...?

A pre-image analysis will give us the answer. Recall that,

$\alpha \in sol(e : b) \;\Leftrightarrow\; val(e, \alpha) = b$

By pre-image analysis we mean the decomposition of $f(a) = b$, where $f : A \to B$, as:

$f(a) = b \;\Leftrightarrow\; a = a_1 \; or \; a = a_2 \; or....$

where $a_1, a_2, \ldots$ are the preimages[32] of b by f. Nothing to it really, right to left it is function application, computation, and left to right it is decomposition, analysis, truth conditions, solving the equation $f(a) = b$ for a. If the function f has structured arguments, e.g., $f : A \times A' \to B$, the decomposition proceeds in two steps, e.g.,

$f(a, a') = b$

$\Leftrightarrow \; (a, a') = (a_1, a_2) \; or \; (a, a') = (a_3, a_4) \; or....$

$\Leftrightarrow \; (a = a_1 \; and \; a' = a_2) \; or \; (a = a_3 \; and \; a' = a_4) \; or \ldots$

Preimage analysis is the driving force behind tableau expansion, a kind of case analysis that is commonly seen in logic [9]. For example, using a classical binary interpretation for the connective $\wedge$,

$val(e_1 \wedge e_2) = 1 \;\Leftrightarrow\; (val(e_1) \otimes val(e_2)) = 1 \;\Leftrightarrow val(e_1) = 1 \; and \; val(e_2) = 1$

and

$val(e_1 \wedge e_2) = 0$

$\Leftrightarrow \; (val(e_1) \otimes val(e_2)) = 0$

$\Leftrightarrow \; (val(e_1) = 0 \, and \, val(e_2) = 0) \; or \; (val(e_1) = 0 \, and \, val(e_2) = 1) \; or \; (val(e_1) = 1 \; and \; val(e_2) = 0)$

In a tableau expansion one would see rather a branching based on the so called β-rule:

$e_1 \wedge e_2 : 0 \;\rightsquigarrow e_1 : 0 \; or \; e_2 : 0$

The preimage analysis formulation is equivalent, just more systematic, more rigid[33]. It applies to any operation H_c not just to the Boolean operations, $\otimes$, $\oplus$, ...

So, how does the structure of term e influence $sol(e : b)$?

$\alpha \in sol((c\, e_1\, e_2..) : b)$

$\Leftrightarrow \; val((c\, e_1\, e_2..), \alpha) = b$

$\Leftrightarrow \; H_c(val(e_1, \alpha), \, val(e_2, \alpha), ...) = b$

[32]The number of preimages is assumed finite. We do not want to work with witnesses here.

[33]The handling of signed formulas in tableau expansions [8], or, in general, many valued tableau expansions [21], is quite close in spirit and techniques with what we are doing here.

$\Leftrightarrow$ $(val(e_1),\, val(e_2), ...) = (b_1, b_2, \ldots)$ *or* $(val(e_1),\, val(e_2), ...) = (b'_1, b'_2, \ldots)$ *or* $\ldots$

$\Leftrightarrow$ $(val(e_1) = b_1$ *and* $val(e_2) = b_2$ *and* ...) *or* $(val(e_1) = b'_1$ *and* $val(e_2) = b'_2$ *and* ...) *or* $\ldots$

$\Leftrightarrow$ $(\alpha \in sol(e_1 : b_1)$ *and* $\alpha \in sol(e_2 : b_2)$ *and* ...) *or* $(\alpha \in sol(e_1 : b'_1)$ *and* $\alpha \in sol(e_2 : b'_2)$ *and* ...) *or* $\ldots$

The decomposition is driven by the preimages of the operations H_c and can be iterated until we reach the constraints on atomic terms. We start with a single constraint, $val(e, \alpha) = b$, and end up with a complex of constraints, a disjunction of conjunction of disjunctions of ... of conjunctions of atomic constraints, a tree with '*and*' and '*or*' at the nodes and atomic constraints as $val((a), \alpha) = b$ at the leaves.

The connectives '*and*' and '*or*' may be seen as part of the presentation language - here, English for mathematicians, loosely speaking -, or they may be seen as the connectives of a dedicated constraint language. Naturally, we would prefer to work with a formal constraint language, but, it turns out, we do not really have to introduce a specific language: maximal expressivity considerations will provide a suitable surrogate.

Alternatively, using for example a (possibly formal) calculus of classes:

$\alpha \in sol((c\, e1\, e2..) : b)$

$\Leftrightarrow$ $\alpha \in (sol(e_1, b_1) \cap sol(e_2, b_2) \cap ...) \cup (sol(e_1, b'_1) \cap sol(e_2, b'_2) \cap ...) \cup \ldots$

or, more primitively, the algebraic system $\mathbb{B}$,

$[sol((c\, e1\, e2..) : b)]_\alpha = ([sol(e_1, b_1)]_\alpha \otimes [sol(e_2, b_2)]_\alpha \otimes ...) \oplus ([sol(e_1, b'_1)]_\alpha \otimes [sol(e_2, b'_2)]_\alpha \otimes ...) \oplus \ldots$

where $[A]_\alpha = 1$ if α is an element of A and $[A]_\alpha = 0$ otherwise.

The $\{and, or\}$-tree of constraints, alternatively the $\{\cap, \cup\}$-tree of solution sets or the $\{\otimes, \oplus\}$-tree of Boolean bits, can be left as is or further processed, yielding for example a sum of products. Either way, we have the decomposition we are after: $sol(e : b)$ is reexpressible in term of $sol((a) : b')$, with $a \in Atoms(e)$. Rearranging the tree, migrating the '*or*' nodes to the root, removing redundancies, and pruning or closing branches with conflicting atomic constraints, is Boolean algebra at work. The moves can all be justified equationally via their Boolean algebra counterparts.

The solution set $sol(e : b)$ is reexpressible as an union of intersections, the intersections being over $sol((a) : b')$, with $a \in Atoms(e)$. The solution set $sol((a) : b')$ is a cylindrical set of assignments, $\{\alpha \,|\, \alpha(a) = b'\}$, the intersections , e.g. $sol((a_1) : b_1) \cap sol((a_2) : b_2)$, are also cylindrical sets, and $\{\alpha\} = (\cap)_{a \in Atoms(L)} sol((a) : \alpha(a))$.

What about the converse? Under which conditions is an union of intersections of $sol((a) : b')$, with $a \in Atoms(e)$, a solution set? A cylindrical set of assignments, e.g., $\{\alpha \,|\, \alpha(a_1) = b_1\, and\, \alpha(a_1) = b_1\}$, is always a solution set, for $sol(\{(a_1) : b_1,\, (a_2) : b_2\}) = \{\alpha \,|\, \alpha(a_1) = b_1\, and\, \alpha(a_1) = b_1\}$. If the interpretation is functionally complete, all sets of assignments are solution sets since all functions from assignments to values are representable [34]. At the other end of

[34]Proof, for example, by reductio ab absurdum: Assume set A is not a solution set, then a function f that has A as a preimage set cannot be represented by a polynomial term.

the spectrum, the solution sets under the initial interpretation $A(L)$ are limited to the cylindrical sets.

6 Maximal Expressivity

6.1 A Boolean language for reports

Where are we now? Our language community agrees on how to report information from the field: it agrees on a term language as the means of expression and on a way of computing the values of composite terms. The stories that can be told may differ greatly in their syntax and intent but as far as information content is concerned, there are as many stories as there are solution sets, as many stories as there are formal concepts. Depending on L and H, the number of formal concepts the language differentiates may not be very large.

Let's define the expressivity of the interpreted language (L, H) as its lattice of formal concepts (modulo isomorphism), or a numerical characterization thereof when convenient, e.g., its cardinal. This choice is driven by the central role that solution sets play and is, one would expect, more potent than a measure based on the number of term connectives or the number of truth values.

To increase expressivity, the local academy may decide to alter the language, adding connectives, changing some interpretations[35], or it may agree to enrich the lattice of solution sets in some other way, for example by changing what counts as a report. So far, a report has been a finite functional set of statements - a finite partial function. An interesting way of improving expressivity is via the introduction of connectives at the report level, iterating what we have done for the sentences. What's sauce for the goose is sauce for the gander. Note the shift in perspective though: with connectives at the report level, we now have to view statements and reports first and foremost as linguistic entities; connectives combines linguistic entities. Statements and reports as couples and finite set of couples, i.e., as mathematical objects, our old take on the matter, are reduced to the role of mathematical representations.

To set up a term language for reports, we need to specify the atoms (our former notion of reports for example), the connectives, and eventually the valuations of interest. Let's work out in some details a Boolean language of reports.

Term language on reports:
$\Phi := (\Sigma) \mid (\& \, \Phi \, \Phi) \mid (\vee \, \Phi \, \Phi) \mid (\neg \, \Phi)$
Intended interpretation:
$[(\Sigma)] = sol(\Sigma)$
$[(\& \, \Phi_1 \, \Phi_2)] = [\Phi_1] \cap [\Phi_2]$
$[(\vee \, \Phi_1 \, \Phi_2)] = [\Phi_1] \cup [\Phi_2]$
$[(\neg \, \Phi_1 \, \Phi_2)] = Asg - [\Phi]$
The valuation $[\;]$ is the algebraic extension of sol to composite reports.

[35] A functionally complete interpretation would be the limit of that approach and would provide maximal expressivity.

Why the interest in a Boolean language of reports? The set of all assignments happens to be naturally endowed with a Boolean algebra structure, a Boolean language of reports merely transfers that structure. We are using a structure[36] present at the solution set level, the operations $\cap$, $\cup$, and $(Asg - A)$, to interpret the newly introduced report connectives. Any operation on $\wp(Asg(L,H))$ could be used to the same effect. The Boolean structure is nonetheless of particular interest and plays a major role in our analysis of expressivity and in logic in general. With composite reports, the world of reports get richer, the set of solution sets grows in proportion, and expressivity increases.

In practice, the speaker may prefer to work with statements rather than reports. A language for Boolean statements would follow the same route:

$s' := (s) \mid (\& \, s' \, s') \mid (\vee \, s' \, s') \mid (\neg \, s')$

$[(s)] = sol(s)$

$[(\& \, s'_1 \, s'_2)] = [s'_1] \cap [s'_2]$

...

The differences between a language of composite reports and a language of composite statements is mostly cosmetic, since a report[37] is (L,H)-equivalent to the conjunction of its elements:

$\Sigma \equiv_{(L,H)} s_1 \& \, s_2 \& ...$ where $s_1, s_2, \ldots$ are the elements of Σ.

From $sol(\Sigma) = \bigcap_{s \in \Sigma} sol(s) = sol(s1) \cap sol(s2) \cap \ldots$

Composite reports and composite statements have the same expressive power[38] and are linguistically quite close; translating a composite report into a composite statement is fast and uneventful.

Our set of solution sets is now a Boolean algebra, the Boolean algebra generated from $\{sol(s) \mid s \in ST(L,H)\}$ plus Asg, to account for $sol(\emptyset)$, by intersection, union, and complementation. The original complete lattice of solution sets was generated from $\{sol(s) \mid s \in ST(L,H)\}$ by generalized intersection and the addition of Asg.

Which two Boolean reports tell the same story? Relative to the interpretation given above, we recover, not surprisingly, the usual calculus of classes, this time at the composite report level, mirroring once again the polynomial identities of the Boolean algebra $\mathbb{B}$, for example $(\Phi_1 \wedge \Phi_2) \equiv (\Phi_2 \wedge \Phi_1)$ or $(\Phi \wedge \Phi) \equiv \Phi$. With Boolean reports, we get a new layer of logical results.

Of note,

1. The Boolean statement $e : b_1 \;\vee\; e : b_2 \ldots$, where $\{b_1, b_2, \ldots\} = B_e$, is a tautology.

2. The negation connective is eliminable:

$\neg(e : b) \equiv_H e : b_1 \;\vee\; e : b_2 \ldots$ where $\{b_1, b_2, \ldots\} = B_e - \{b\}$

3. The conjunction $(e : b_1) \, \& \, (e : b_2)$ is not satisfiable if $b_1 \neq b_2$.

Comment

[36]A measure would be another interesting structure to transfer, the listener may want to valuate the likelihood of a story.

[37]the set of statement kind

[38]as determined by their respective lattice of solution sets

(1) Instead of working with a new layer of valuation, one may prefer a specification such as: “Assignment α satisfies report term $(\vee\,\Phi_1\,\Phi_2)$ iff α satisfies report term Φ_1 or α satisfies Φ_2”.

This is fine naturally; it is merely an extension of the binary relation *Sat* to composite reports and it it the format that is most often used in logic textbooks to explicit a semantics. The algebraic approach has the advantage of not relying on the meaning one may attach to a presentation language connective as “or”. What I, as an English speaker mean by “or” may not be what you, the reader, understand by “or”. The operation $\oplus$, on the other hand, takes only four entries to be fully defined, and the computation of $[(\vee\,\Phi_1\,\Phi_2)]_\alpha$ in term of $[\Phi_1]_\alpha$ and $[\Phi_2]_\alpha$ does not leave much room for personal interpretation,

$[(\vee\,\Phi_1\,\Phi_2)]_\alpha = [\Phi_1]_\alpha \oplus [\Phi_2]_\alpha$

(2) The setup sentence-statement can be iterated ad infinitum[39]. Connectives can be added at will, and so can interpretations.

6.2 Conjunction and disjunction

Although we start with an arbitrary stock of connectives, Boolean connectives are ubiquitous in this work. They appear:

- at the term level, e.g., $e_1 \wedge e_2$
- at the statement level, e.g., $s'_1 \& s'_2$
- at the constraint level where ‘*and*’ and ‘*or*’ are used to express composite constraints on assignments and may be seen as part of the presentation language or as part of a dedicated language for constraints.

Relative to the interpretation of the statement connectives given, the last two usages are closely related:

$(val(e_1,\alpha) = b_1 \text{ and } val(e_2,\alpha) = b_2 \text{ and } ...) \Leftrightarrow \alpha \in sol(\{e_1 : b_1,\ e_2 : b_2,\ \ldots\}) \Leftrightarrow \alpha \in sol((e_1 : b_1)\&(e_2 : b_2)\&\ldots)$

One usage can be defined in term of the other, e.g., for Boolean statements,

$\alpha \in sol(s'_1) \text{ and } \alpha \in sol(s'_2) \text{ and } \ldots \Leftrightarrow \alpha \in sol(s'_1 \,\&\, s'_2 \,\&\ldots)$

$\alpha \in sol(s'_1) \text{ or } \alpha \in sol(s'_2) \text{ or } \ldots \Leftrightarrow \alpha \in sol(s'_1 \vee s'_2 \vee \ldots)$

An $\{and,\ or\}$-tree of constraints corresponds naturally to a $\{\&, \vee\}$-composite statement. The correspondence is particularly transparent with a $\{\cap, \cup\}$-tree of solution sets format.

7 Wrapping Things up

7.1 An accounting of possibilities

We are reaching the end of the road here; there is nothing else to decompose. A statement s' is (L, H)-equivalent to a sum of products, to a disjunction of conjunctions of statements whose terms are atomic:

$s' \equiv_H (a_1 : b_1 \,\&\, a_2 : b_2 \,\&\, \ldots) \vee (a_2 : b_3 \,\&\, a_5 : b_4 \,\&\, \ldots) \vee \ldots$

[39]Typing with a hierarchy of universes, e.g., (e:type:sort:kind:...), illustrates the idea.

This sum of product is a set of alternatives, an accounting of possibilities. We start with one log book for the speaker, and end up with a set of log books for the listener: the set of possibilities as specified by the report.

Note that there is no negation connective in these normal forms. The range of values B_T accounts for all the possibilities. Logic, from a valuation point of view, is at heart two connectives, '*and*' and '*or*'. We need the 0-ary connective $\perp$ to express the absence of possibilities, and the 0-ary connective $\top$ can be added for convenience.

In term of solution sets, all solution sets $sol(s')$, s' composite statement, can be reexpressed as a finite union of cylindrical sets of assignments where a cylindrical set of assignment is the solution set of an alternative, e.g., $sol(a_2 : b_3 \,\&\, a_5 : b_4 \,\&\, \ldots)$.

Pretty much what we are used to with classical propositional logic, but for negation.

7.2 A language of alternatives

With hindsight, the linguistic community could have provided Alice with a streamlined version of the language:

$n = \perp \,|\, \top \,|\, (l) \,|\, (\vee\, l\, n)$

where the atoms l are log books. Technically, nothing more than lists[40] of log books. Assignment α satisfies report n if α includes one log book of n. Computationally:

$sat(\alpha, \perp) = 0$

$sat(\alpha, (l)) = [l \subseteq \alpha]$

$sat(\alpha, (\vee\, l\, n)) = sat(\alpha, (l)) \oplus sat(\alpha, n)$

Lists of assignments would play the same role, without the redundancies. In a list of log books, one log book may be included in another. A list of assignments is however a verbose format, not the format of choice when conciseness is the goal[41].

7.3 Sequents

At some point, the members of our linguistic community may want to work with a deductive system and would prefer one that allows elimination and introduction of term connectives. A tableau calculus mostly eliminates connectives[42]. Sequent calculus are typically more symmetric. In terms of sequent, working with statements, the introduction rule

$$e_1 : b_1,\ e_2 : b_2,\ \ldots \vdash (c\, e_1\, e_2 \ldots) : H_c(b_1, b_2, \ldots)$$

[40]'list' as used in computer science.

[41]Circuit minimization is a big issue in computer science. Our speakers are oblivious to such concerns. They have unlimited resources. A list of assignments format would be a fine option.

[42]More precisely, term connectives are eliminated and the constraint connectives '*and*' and '*or*' are introduced.

expresses the computational rule

$$\frac{e_1 : b_1,\ e_2 : b_2,\ \ldots,\ c : (B_1, B_2, \ldots \rightarrow B_m)}{(c\, e_1\, e_2 \ldots) : H_c(b_1, b_2, ...)}$$

And preimage analysis is implemented via an an elimination rule :

$$(c\, e_1\, e_2 \ldots) : b \vdash (e_1 : b_1\ \&\ e_2 = b_2\ \&\ ...)\ \vee\ (e_1 : b_1'\ \&\ e_2 = b_2'\ \&\ ...)\ \vee\ \ldots$$

where $b = H_c(b_1, b_2, ...) = H_c(b_1', b_2', ...) = \ldots$.

Term connectives may be thus introduced and eliminated at will within a sequent derivation. Note that we are giving matching sequent rules for the term connectives only, not for the statement connectives.

7.4 Distinguished value reexpression

We can collapse the sentence level and the statement level by using a distinguished value, let's say 1, and a few special purpose term connectives[43]. In a multi-sorted setting, it is notationally simpler to assume that the same value 1 is present in all B_T.

Consider the term connectives $[1/b]$, one per value b. The connective is unary and its interpretation $H_{[1/b]}$ is partially specified by the restriction:

$(e) : b \equiv_H ([1/b]e) : 1$

Spelling out the equivalence,

$val_\alpha([1/b]e) = 1 \ \Leftrightarrow\ H_{[1/b]}(val_\alpha(e)) = 1 \Leftrightarrow\ val_\alpha(e) = b$

The operation $H_{[1/b]}$ is similar to a Post unary operator (iterates thereof) [33], $H_{[1/b]}(b) = 1$. In fact, our distinguished value reexpression setting is pretty much a Post algebra setting. In a bivalent logic, the connective $[1/0]$ is the negation connective.

Let's assume that statement connectives & and $\vee$ are representable by sentence connectives, by $\tilde{\&}$ and $\tilde{\vee}$ let's say, in the following sense:

$(e_1 : 1\ \&\ e_2 : 1) \equiv_H (e_1\ \tilde{\&}\ e_2) : 1$

$(e_1 : 1\ \vee\ e_2 : 1) \equiv_H (e_1\ \tilde{\vee}\ e_2) : 1$

The constraints put on the interpretations $H_{\tilde{\&}}$ and $H_{\tilde{\vee}}$ are quite mild,

$H_{\tilde{\&}}(val(e_1, \alpha), val(e_2, \alpha)) = 1\ \Leftrightarrow\ val(e_1, \alpha) = 1\ \mathit{and}\ val(e_2, \alpha) = 1$

$H_{\tilde{\vee}}(val(e_1, \alpha), val(e_2, \alpha)) = 1\ \Leftrightarrow\ val(e_1, \alpha) = 1\ \mathit{or}\ val(e_2, \alpha) = 1$

If the value 1 is seen as the top value of the range, then $H_{\tilde{\&}} = min$ and $H_{\tilde{\vee}} = max$ work just fine.

This is enough structure to represent any composite statement s' as a sentence. We just have to propagate the above transformations from the leaves to the root. If the composite statement is already available as a (non empty) list of alternatives, the transformation is transparent:

$$s' \equiv_H (a_1 : b_1\,\&\, a_2 : b_2\,\&\, \ldots) \vee (a_2 : b_3\,\&\, a_5 : b_4\,\&\, \ldots) \vee \ldots$$
$$\Leftrightarrow\ s' \equiv_H (([1/b_1]a_1\,\tilde{\&}\,[1/b_2]a_2\,\tilde{\&}\,\ldots)\tilde{\vee}([1/b_3]a_3\,\tilde{\&}\,[1/b_4]a_5\,\tilde{\&}\,\ldots)\tilde{\vee}\ldots) : 1$$

[43]The special purpose term connectives are strong candidates for inclusion into a natural set of logical term connectives

Instead of attaching values to sentences and reporting the results, the speaker may equivalently communicate a single sentence. When the convention of the linguistic community is "Uttering is Stating", e.g., the convention in use in many natural languages, the speaker can still express herself, with no loss in expressivity.

7.5 Falling back on our two feet

If the interpretation is binary, polynomial equivalence becomes logical equivalence,

$e_1 \simeq e_2$
$\Leftrightarrow (e_1 : 0) \equiv (e_2 : 0)\, and\, (e_1 : 1) \equiv (e_2 : 1)$
$\Leftrightarrow (e_1 : 1) \equiv (e_2 : 1)$

hence the little need for two equivalence relations in classical logic, extensionally speaking. Intentionally, the two relations are naturally quite different, one talks of information processing, the other of satisfiability.

7.6 Logical notions by hook or by crook

Our version of logical entailment is based on satisfiability considerations,

$\Sigma_1 \models_{(L,H)} \Sigma_2 \quad \Leftrightarrow \quad sol(\Sigma_1) \subseteq sol(\Sigma_2)$

Let's consider an empty set of premisses. A statement s is a (L, H)-validity, or $\emptyset \models_H s$, if it is a (L, H)-consequence of the empty set - a tautology per the terminology we adopted earlier.

In many-valued logic, logical notions are more commonly introduced via the artifact of designated values. For example:

- sentence e is 1-valid, or $\emptyset \models_1 e$, if the H-value of e at any atomic assignment α is 1, i.e., if the range of the function on assignments $val(e)$ is limited to the singleton $\{1\}$.

- sentence e is D-valid, or $\emptyset \models_D e$, if the H-value of e at any atomic assignment α is an element of the designated set of values D, i.e., $Range(val(e)) \subseteq D \cap B_e$

These sentence based versions of logical notions are, nicely enough, definable in term of the satisfiability based logical notions,

$\models_1 e \quad \Leftrightarrow \quad \models_{(L,H)} (e : 1) \quad \Leftrightarrow \quad sol(e : 1) = Asg(L, H)$
$\models_D e \quad \Leftrightarrow \quad \models_{(L,H)} (\vee)_{d \in D}\, (e : d) \quad \Leftrightarrow \quad (\cup)_{d \in D}\, sol(e : d) = Asg(L, H)$

Conversely, once statements can be represented by sentences (see *Distinguished value reexpression* above), satisfiability based logical notions are definable in term of logical notions relative to a single designated value (one per sort). For example,

$\models_{(L,H)} (e : b) \quad \Leftrightarrow \quad \models_{(L,H)} ([1/b]e : 1) \quad \Leftrightarrow \quad \models_1 [1/b]e$

That all these versions of validity, entailment, ..., are so closely related is at first sight surprising, but it shouldn't really, for they are all based on the same notion: set of assignments. Entailment relations relative to a set of designated values involve set of preimages, set of assignments, and the entailment relation issued from satisfiability is inclusion of solution sets in disguise.

8 Clarifications, limitations, generalizations

This is where our analysis of Alice's language ends. We started with a generous language of terms and an unlimited supply of truth values, and found nevertheless the means of reporting limited. Alice expresses herself by attaching values to chosen sentences. Values of composite sentences are computed following agreed upon linguistic conventions. Values of atomic sentences are read from sensor displays, or, would our cognitive agents be more capable, may be the product of the deliberation of a committee of expert observers, the verdict of a jury, or the hunch of an oracle - subjective valuation if any. It does not really matter. The present analysis is all about information processing, how information is transformed, transferred, hidden, and recovered, not how it is acquired in the first place. The situation is pretty similar to image analysis. Once we have the pixels, it is filtering, convolution, segmentation, feature extraction, reconstruction of the original image, i.e., computation and inverse problems, one way or another. Where the pixels come from or what they correspond to is altogether a different matter. The original image is our log book. Transformed images, features of the original image, are the computed pieces of data. Reconstructing an image from features, from partial information, is the satisfiability problem.

Two sentences e_1 and e_2 are said to be equivalent if their associated functions, $val(e_1)$ and $val(e_2)$, are equal, $val(e_1) = val(e_2)$. Two sentences are equivalent if, upon computation, they always end up with the same values, if they are equal as polynomials. Which two sentences are equivalent is highly dependent on the interpretation of the connectives, on the local conventions. Equivalence of sentences is a theory of algebraic identities; it reduces logic to algebra, to a study of polynomial terms.

It is no small task to justify a particular choice of interpretation as the correct one. One is free naturally to choose any algebraic operation whatsoever; mathematically speaking, it makes no difference. But if the objective is the formal representation of connectives already present in the community in some informal way, then, choices have to be justified, some representations may be better than others. And, given a particular algebraic system, some informal connectives may just not have a good representative. The English connective 'if then' is case in point; controversies abound regarding its correct Boolean interpretation. One is free to choose a Boolean interpretation for the formal connective $'\supset'$, but to read it as the English connective 'if then' and claim that the chosen Boolean interpretation is or is related to the (English) meaning of the connective is quite another matter. Natural languages connectives and formal language connectives live in two different worlds; much may be lost in translation - or gained, e.g., conciseness and clarity. Extracting the principles of formal logic, including its connectives, from a study of natural languages is a common and interesting approach. It may not be the most profitable or natural approach to the subject however. Natural languages and formal languages yield to different forces; they follow different dynamics.

A report is a finite set of valued sentences, i.e., a finite set of statements. A report if satisfiable if it can be generated by a competent speaker from at least one assignment, one log book. Two reports are equivalent if they are satisfied by the same assignments. A report is a set of solutions and a set of solutions is a theory, Galois connection wise. Report equivalence and report entailment mirror quite faithfully our intuitions about logical equivalence and logical entailment, more so certainly than polynomial equivalence. Bob, the limited listener, hears a set of assignments. One expects a suitably capable speaker to adapt the amount of information transferred to the needs and status of the listener, in general, to tailor her speech to fulfill some objectives.

From a satisfiability point of view, a statement is a constraint on assignment, an equation in need of a solution. Report solution sets are intersection of statement solution sets. Preimage analysis tells us that, once normalized, statement solution sets are union of cylindrical sets of assignments. The converse does not necessarily hold however: the singleton $\{\alpha\}$ is a solution set, but the union of a few singletons may not be a solution set. Decomposition of a constraint into constraint complexes introduced two key connectives, 'and' and 'or'. These connectives are constraint language connectives, they allow us to express constraints involving composite sentences in term of (composite) constraints involving simpler sentences. We are not free to choose the interpretation attached to these connectives, their meaning is determined by their usage within a tableau expansion, they are in that sense absolute. On the other hand, the setting sentence-statement can be iterated, ad infinitum. The values attached to $[sol(e,b)]_\alpha$ do not have to be limited to 0 and 1; characteristic functions do not have to range into $\{0,1\}$. Statements are also linguistic entities, they are the sentences of some (other) formal language. As such, statements can be valued, statement connectives can be interpreted and quite arbitrarily so. An interpretation for 'and' and 'or' that respects the preimage analysis equivalences, i.e., the usage of 'and' and 'or' within a tableau expansion, is a perfectly suitable alternate reading of the connectives. In that sense, the meaning of 'and' and 'or' is not really absolute.

Attempts to increase the expressivity of the language led us to composite reports and composite statements, to statements connectives that can be read as the 'and' and 'or' connectives of constraint decomposition, and to normal forms. A composite report is logically equivalent to a disjunction of conjunctions of statements whose terms are atomic; this disjunction of conjunctions is a set of alternatives, an accounting of possibilities. Conversely, in a maximally expressive language, all such sets of alternatives are reportable.

Explorer Alice has by now quite a few ways of reporting her observations:

1. She can share her raw data.

2. She may report a summary of her observations; the log book is filtered, aggregated, averaged in various ways.

3. The report may be a finite set of statements or Alice may find it more appropriate to use the language of Boolean statements.

4. She may value transparency and opt for a list of alternatives format.

5. Alice's audience may be averse to statements and the convention that "Uttering is Stating" may be the law of the land. Subject to some conditions on the means of expression[44], this is not a limitation; Alice can translate any report into an equivalent sentence.

Option 1 transfers all the information. Option 2 allows some hedging. Options 3, 4 and 5 provide maximum expressivity. Option 2 favors direct access to highly integrated pieces of information, whereas option 4 privileges rawness.

Upon decomposition, the sentence level connectives fade away and two constraint language connectives appear. Option 4 provides a streamlined version of the decomposition; there are no sentence connectives in a list of alternatives. No matter how many truth values and sentence connectives we start with, we can always shift to a positive bivalent logic and two connectives, 'and' and 'or', or equivalent. There is no loss in the stories that can be told as long as the only way we have to judge stories is via satisfiability.

The main technical requirement for our results is, besides compositionality, the finiteness of the decomposition of the operations H_c: the preimage analysis may involve only a finite number of preimages. Similar results can be reached in quite different settings: any language that supports a tableau calculus is a fine candidate. Let's consider for example a multimodal language L with Kripke style interpretations for the modal connectives. The value of composite term e at world w is computed from the values of the subterms of e at world w and from the values of the subterms of e at worlds R-neighboring w, where R is some binary relation on worlds. The computation is maybe more elaborate than a simple call to some operation H_c but it is nonetheless a perfectly fine computation assuming the number of R-neighbors to world w finite - the total number of worlds does not have to be finite. Solutions are finite entities and the decomposition of a statement is a preimage analysis that can be iterated all the way down to atomic sentences. If the number of R-neighboring worlds is not finite, the situation is quite different. A typical example would be a quantified language with a non-finite domain of quantification. The computation of the value of a universally quantified sentence would require, if we follow a propagation of value protocol, the prior computation of the values of all its instances. That is a feat beyond our best technologies, our computers are limited to the processing of a finite number of values. Should we ever acquire a countable (or larger) number of values, we still would have the bother of aggregating them into a single value. Two tasks that are quite problematic by our contemporary computability standards. If we do not compute the value of a universal sentence from the values of its instances, how do we determine the value of a universal? In general, we don't. At best, we relate values of various sentences or we search for examples and counterexamples. On the decomposition side, the situation is less dire: through the judicious use of witnesses and expansion priorities, preimage analysis can grow into a tractable list of alternatives, although typically the list is not finite and the tableau expansion is never-ending. But that is fine for it is the tableaus

[44]For example, the availability of suitable sentence connectives or functional completeness in some sense.

that do end that are often of interest.

The speaker and the listener

Interestingly enough, the sharp distinction between sentences and statements, which at the start of this work we thought was fundamental, somewhat evaporates - see *Distinguished value reexpression*. The distinction may not be as cogent as initially envisioned but it is nonetheless a good distinction to keep. For one thing, sentences are processed by the speaker, statements by the listener and the difference between speaker and listener remains as relevant as ever. The speaker computes, combines, summarizes. The listener analyzes, decomposes, solves. It is algebra versus coalgebra, synthesis versus analysis, many to one versus one to many. One builds, the other breaks. The speaker propagates values, the listener propagates constraints. The speaker is a forward chainer, she leads the story; the listener manages goals and subgoals.

Information processing

Our analysis of the transfer of information between the speaker and the listener gives us two logics. Satisfiability issues are traditional logical concerns, information processing is somewhat of a novel topic. The two subjects are naturally closely related, satisfiability determination is the inverse problem of processing. Should we then consider logic as the study of information transfer, as a subfield of data analysis? The thesis seems mildly absurd in the face of 2500 hundred years of logic tradition that was never really concerned with information transfer, but it is also clear that reporting is nothing but information transfer. The question is naturally more pertinent in a setting that considers a plurality of values. The construction of a report is not the only way a log book can be put to good use. Data mining, computation of correlations, statistical reduction, discovery of relations, are all worthy and, nowadays, lucrative endeavors. It may be data processing beyond the confines of a particular algebraic model, computation beyond the operations provided by $H(L)$. But it is still computation.

Moreover, if thinking is information processing, how much of psychology is logic, how much of logic is psychology? A question put to rest in the nineteenth century that, interestingly, refuses to die.

It becomes a matter of what we mean by the word ‘logic’ rather than what logic is, and what is meant by the word has certainly evolved over the years.

What happened to truth?

Still there, but mostly irrelevant to our considerations. Even at the log book level there is not much truth. The sensors may be faulty or badly calibrated, Alice may have landed on the wrong planet, or a few sensor displays may have been misread. It does not really matter; the values could as well have been assigned at random. Our story begins once atomic sentences have a value; how the values are obtained is of no import. Which is not to say we are not interested by truth or shouldn't. It is just that our approach does not presume a notion of truth nor does it require one. Logic, as presented in this paper, is not grounded by reality, by features of natural languages or by any notions of truth. What we have rather is a formal language perspective: a (re)construction of logical notions from an analysis of the transfer of information between speakers and listeners. Logic is not per se a theory of truth nor a theory of truth transmission.

Contrast: Logic from empirical truth.

Let's assume that we have a way of determining truth, typically involving an actual world understood as the ultimate arbiter of what a fact is and a notion of correspondence between facts and sentences, a relation that is somehow accessible to the observer. The world has facts, some correspond to atomic sentences and some to composite sentences. Sentences are true or false by virtue of corresponding facts. In such a setting the value of a composite sentence is not computed, it is observed; at best one may recognize a functional dependence between sentence and subsentence values. Logical notions have their origins in the world, logical truth is seen as a special kind of empirical truth, and logical equivalences are natural laws - hence falsifiable and subject to revision. Logical results are what they are because the actual world is as it is.

However appealing and popular such an approach may be, it is not without its problems. For one, the world of facts may not be that giving. What are the facts corresponding to disjunctions, negations, universals, necessities, and, if logical truth is empirical truth, to tautologies and contradictions? We prefer an approach that computes values. There are no composite facts in Alice's world, no negative facts; at most, facts corresponding to atomic sentences. Alice does not have a privileged access to Truth: she merely records and computes.

Access to Reality

Our run-of-the-mill cognitive agent is going to have a finite number of sensors, certainly not a countable number of them, and will record a finite number of observations in its lifetime. This gives the agent a finite amount of information to work with. How much knowledge of the actual world can a cognitive agent have if its only access to that world is a finite amount of sensory data? How much knowledge of the actual world can a homo sapiens have?

So, is logic conventional or empirical?

Strictly speaking, the logics discussed in this work are neither. They describe local linguistic practices, they do not prescribe them. The rules of formation and the rules of value propagation are normative; the results of the local logician are not. Alice does not need a logician to be a competent speaker. As long as she complies with the formative and valuative rules of the language, her reports will be well-formed and well-valued. She does not need to be aware of, or to knowingly follow, any logical laws.

On the other hand, Uncle George is a scientist. He studies the properties of sentences, statements, reports, solution sets, theories, and the properties of the properties, the properties of the properties of the properties, and so on. If his medium of work is mathematics, his results are mathematical theorems; the local language becomes a mathematical object via some representation and its properties are derived within the confines of the mathematical standards of the times. Relative to such a representation, logical results have the certainty of a mathematical theorem. If the local logician is more of a field scientist, he will spend his days observing the local world of sentences and statements (tokens thereof), the local practices, collecting evidences, examples, counterexamples; his results are empirical.

The logic of sentences and the logic of statements presented here are not conventions; they are rather the product of conventions. The study of the properties of the local language can be, clearly enough, theoretical or empirical.

References

[1] H. Andréka, I. Németi, and I Sain. Algebraic logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 2, pages 133–247. Kluwer Academic Publishers, 2nd edition, 2001.

[2] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998.

[3] Hendrik Pieter Barendregt, Wil Dekkers, and Richard Statman. *Lambda Calculus with Types.* Perspectives in logic. Cambridge University Press, 2013.

[4] J.C. Beall and G. Restall. *Logical Pluralism.* Clarendon Press, 2006.

[5] Jean-Yves Béziau. *Logica Universalis: Towards a General Theory of Logic.* Birkhäuser, 2005.

[6] P. Blackburn, J.F.A.K. van Benthem, and F. Wolter. *Handbook of Modal Logic.* Studies in Logic and Practical Reasoning. Elsevier Science, 2006.

[7] Stanley N. Burris and H.P. Sankappanavar. *A Course in Universal Algebra.* Springer-Verlag, 1981.

[8] Marcello d'Agostino. Tableau methods for classical propositional logic. In Marcello D'Agostino, Dov M Gabbay, Reiner Hähnle, and Joachim Possega, editors, *Handbook of tableau methods*, pages 45–123. Springer Netherlands, 1999.

[9] Marcello D'Agostino, Dov M Gabbay, Reiner Hähnle, and Joachim Possega, editors. *Handbook of tableau methods.* Kluwer, 1999.

[10] B. A. Davey and H. A. Priestley. *Introduction to lattices and order.* Cambridge University Press, New York, second edition, 2002.

[11] Michael A. E. Dummett. *Truth and Other Enigmas.* Harvard University Press, Cambridge, MA, 1978.

[12] Michael A. E. Dummett. *The Logical Basis of Metaphysics.* Harvard University Press, 1991.

[13] H.B. Enderton. *A Mathematical Introduction to Logic.* Harcourt/Academic Press, 2001.

[14] M. Erné, J. Koslowski, A. Melton, and G. E. Strecker. A primer on galois connections. *Annals of the New York Academy of Sciences*, 704(1):103–125, 1993.

[15] John Etchemendy. *The Concept of Logical Consequence.* Harvard University Press, Cambridge, MA, 1990.

[16] D. Gabbay and F. Guenther, editors. *Handbook of Philosophical Logic.* Kluwer, Dordrecht, 2001-2005.

[17] Bernhard Ganter, Gerd Stumme, and Rudolf Wille. *Formal concept analysis: foundations and applications*, volume 3626 of *Lecture Notes in Artificial Intelligence.* Springer, 2005.

[18] Joseph A. Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1:49–67, 3 1991.

[19] Robert Goldblatt. Algebraic polymodal logic: A survey. *Logic Journal of the IGPL*, 8(4):393–450, 2000.

[20] Siegfried Gottwald. Many-valued logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Spring 2010 edition, 2010.

[21] Reiner Hähnle. Tableaux for many-valued logics. In Marcello D'Agostino, Dov M Gabbay, Reiner Hähnle, and Joachim Possega, editors, *Handbook of Tableau Methods*, pages 529–580. Springer, 1999.

[22] Lloyd Humberstone. Sentence connectives in formal logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Summer 2013 edition, 2013.

[23] Ramon Jansana. Propositional consequence relations and algebraic logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Spring 2011 edition, 2011.

[24] G. Malinowski. *Many-Valued Logics.* Clarendon Press, 1993.

[25] Per Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1):11–60, 1996.

[26] Benjamin C. Pierce. *Types and Programming Languages.* MIT Press, 2002.

[27] P. A. Planchette. Logical consequence. In Lou Goble, editor, *The Blackwell Guide to Philosophical Logic.* Blackwell, 2001.

[28] Hilary Putnam. Is logic empirical? *Boston Studies in the Philosophy of Science*, 5, 1968.

[29] W.V. Quine. Carnap and logical truth. *Synthese*, 12(4):350–374, 1960.

[30] W.V. Quine. *Mathematical Logic.* Harvard University Press, 1983.

[31] Yaroslav Shramko and Heinrich Wansing. Truth values. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Spring 2014 edition, 2014.

[32] A. Tarski. What are logical notions? *History and Philosophy of Logic*, 7:143–154, 1986.

[33] Alasdair Urquhart. Basic many-valued logic. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 2 of *Handbook of Philosophical Logic*, pages 249–295. Springer Netherlands, 2001.