CDM

Equational Logic

Klaus Sutner

Carnegie Mellon University

30-equational 2017/12/15 23:21

1. Equational Logic

- Equational Reasoning

With a view towards implementation, let's take a closer look at the syntactic aspects of equations (also called identities in this context).

We need a simple language to construct terms.

- A collection Var of variables, written x, y, z ...,
- A collection Σ of function symbols, written f, g, h ...
- A map ar : $\Sigma \to \mathbb{N}$ that provides an arity for each function symbol in Σ.

$\langle \Sigma, \mathsf{ar} \rangle$ is called a graded alphabet.

To lighten notation we use subscripts, superscripts and the like.

Also, function symbols of arity 0 are really constants and we write a, b, c ... for them.

Definition

We define the set of terms $\mathcal{T} = \mathcal{T}(\Sigma) = \mathcal{T}(\mathrm{Var}, \Sigma)$ over Var and Σ by induction:

- Every variable $x \in \mathrm{Var}$ is a term, and
- given terms $t_1, \ldots, t_n$ and a function symbol $f \in \Sigma$ of arity n, $f\, t_1 \ldots t_n$ is a term.

Note that we use prefix notation here.

This has the great advantage that we do not need auxiliary symbols such as parentheses or commata, terms are just strings over the alphabet $\Sigma \cup \mathrm{Var}$. We can safely assume the latter two sets to be disjoint.

It is straightforward given a well-formed term t to determine its components f, $t_1, \ldots, t_n$. We can also check whether a given string is indeed well-formed.

Of course, prefix notation is not all that suitable for humans.

So, later on we will use parens and commas and write

$$f(t_1, \ldots, t_n)$$

In particular for binary function symbols we will even use infix notation and write $x \oplus y$ and so on.

So far, a term is a purely syntactical object. In order to attach meaning to these expressions we need to interpret them over a concrete structure, a domain of discourse.

Suppose $\Sigma = \{f_1, \ldots, f_k\}$. Then a suitable structure has the form

$$\mathcal{A} = \langle\, A, F_1, F_2, \ldots, F_k \,\rangle$$

where $F_i : A^{\mathsf{ar}(f_i)} \to A$: F_i is an actual function of the right arity that is used to interpret the function symbol f_i.

Again: f_i is a purely syntactical object, and F_i is its meaning in the real world (over a particular structure $\mathcal{A}$).

When one thinks of a mathematical structure the integers, reals, matrices, trees, graphs and so on come to mind.

No doubt these are the important examples, but there is another class of structures that is very important in developing logic: structures whose elements are terms (of equivalence classes thereof).

More precisely, we can think of $\mathcal{T} = \mathcal{T}(\Sigma)$ as a structure: the carrier set is the set of terms. Suppose we have a binary function symbol f, then the corresponding map is

$$F(s,t) = fst$$

Clearly $\mathcal{T}$ is a suitable structure for our signature, albeit a particularly sterile one.

Definition

A valuation or assignment is a map that assigns elements in a structure to variables:

$$\sigma : \mathrm{Var} \to A$$

By a straightforward induction we can extend a valuation to all terms (rather than just variables):

$$\sigma(f(t_1, \ldots, t_n)) = F(\sigma(t_1), \ldots, \sigma(t_n)).$$

This is just the standard process of evaluation of an expression, given values for all variables. f here is just a syntactic object, a symbol, whereas F is the actual function (the implementation, perhaps even a piece of executable code).

Theorem

Suppose $\sigma : \mathrm{Var} \to A$ *is a valuation over* $\mathcal{A}$*. Then* σ *can be extended uniquely to a homomorphism from the term model* $\mathcal{T}$ *to* $\mathcal{A}$*.*

In other words, once we assign values to the free variables we can assign values to all terms, in a canonical fashion.

Note that this provides an elegant way to specify the corresponding homomorphism, in particular when we are dealing with finitely many variables.

Definition

An equation or identity is an expression

$$s \approx t$$

where s and t are terms.

Note that this is just syntactic sugar, we might as well have defined an identity to be a pair (s, t).

Also, we have deliberately chosen to write $\approx$ rather than the customary and hopelessly overloaded equality sign $=$ to make clear that we are defining a special class of syntactic objects.

Later we will be sloppy and write $s = t$.

Now consider an equation $s \approx t$ and a valuation σ over some structure $\mathcal{A}$.

Definition

$s \approx t$ is valid over $\mathcal{A}$ with respect to σ if $\sigma(s) = \sigma(t)$. We also say that $\mathcal{A}$ is a model of $s \approx t$. $s \approx t$ is valid over $\mathcal{A}$ if it is valid for all valuations.

We use the same terminology for a set E of equations: E is valid over $\mathcal{A}$ if all the equations in E are so valid.

Notation:

$$\mathcal{A} \models_\sigma s \approx t$$
$$\mathcal{A} \models s \approx t$$

For a set of equations E we require that all the equations in E hold over $\mathcal{A}$:

$$\mathcal{A} \models E$$

Example

The usual associative law has $\Sigma = \{*\}$ where $\mathsf{ar}(*) = 2$:

$$x * (y * z) \approx (x * y) * z \qquad (\mathrm{assc})$$

Then (assc) is valid over $\langle\, \mathbb{N}, + \,\rangle$ and $\langle\, \Gamma^\star, \cdot \,\rangle$ but not over $\langle\, \mathbb{N}, x^y \,\rangle$.

Example

To write equations in elementary arithmetic we could use 5 function symbols $\Sigma = \{\oplus, \otimes, S, \mathbf{0}, \mathbf{1}\}$ where $\mathsf{ar}(\oplus) = \mathsf{ar}(\otimes) = 2$, where $\mathsf{ar}(S) = 1$ and $\mathsf{ar}(\mathbf{0}) = \mathsf{ar}(\mathbf{1}) = 0$.

Typical examples of equations are

$$x \oplus y \approx y \oplus x$$
$$x \oplus \mathbf{0} \approx x$$
$$x \oplus S(y) \approx S(x \oplus y)$$

They are all valid over $\langle\, \mathbb{N}, +, S, 0 \,\rangle$.

To pin down monoids we choose a graded alphabet $\Sigma = (*, 1)$ with arities $(2, 0)$.

The axiom system (Mon) for monoids has just three equations:

$$x * (y * z) \approx (x * y) * z$$
$$x * 1 \approx x$$
$$1 * x \approx x$$

Then a structure $\mathcal{A}$ of type $(2, 0)$ is a monoid iff $\mathcal{A} \models (\mathrm{Mon})$.

As we have seen, any monoid also satisfies $((a * b) * c) * d = a * (b * (c * d))$.

Example

Words over an alphabet under concatenation form a monoid.

Example

Functions from A to A under composition form a monoid.

Example

The following Cayley table defines a monoid (by specifying the binary operation, the identity element can be extracted from it).

$*$	e	a	b	c	d	f
e	e	a	b	c	d	f
a	a	c	f	e	b	d
b	b	f	a	d	e	c
c	c	e	d	a	f	b
d	d	b	e	f	c	a
f	f	d	c	b	a	e

Many important monoids satisfy another property:

$$\forall\, x\, \exists\, y\, (x * y \approx 1 \wedge y * x \approx 1)$$

Definition

The element y above is called an inverse of x. A monoid with this additional property is a group.

The quantified formula is a bit more complicated than equations, but we can get rid of it by changing the signature: use $\Sigma = (*, {}^{-1}, 1)$ with arities $(2, 1, 0)$. Then we can add two more identities to (Mon):

$$x * x^{-1} \approx x^{-1} * x \approx 1$$

So groups also have an equational characterization.

More precisely, suppose we have a monoid that satisfies $\forall\, x\, \exists\, y\, (x * y \approx 1 \wedge y * x \approx 1)$. Let's write $\mathsf{inv}(x, y)$ as abbreviation for $x * y \approx 1 \wedge y * x \approx 1$. Then we can prove the following from (Mon):

Claim

The inverse element is always unique:

$$\mathsf{inv}(x, y) \wedge \mathsf{inv}(x, z) \;\rightarrow\; z \approx y$$

Hence it makes sense to introduce a new, unary function $^{-1}$ that denotes this uniquely determined y: we are defining an abbreviation

$$x^{-1} \approx y \iff \mathsf{inv}(x, y)$$

Here is a purely equational proof.

Proof. Assuming $\mathsf{inv}(x, y) \wedge \mathsf{inv}(x, z)$ we have

$$\begin{aligned} z &\approx 1 * z \\ &\approx (y * x) * z \\ &\approx y * (x * z) \\ &\approx y * 1 \\ &\approx y \end{aligned}$$

□

Exercise

Determine precisely the steps in this argument.

We can then prove the following lemma which describes the interaction between $^{-1}$ and multiplication.

Lemma

$$(x * y)^{-1} \approx y^{-1} * x^{-1}$$

Exercise

Prove the last lemma making sure you only use the group axioms and the abbreviation for the inverse operation.

The additive group of integers:

A	integers
$*$	addition
$^{-1}$	unary minus
1	zero

The multiplicative group of positive reals:

A	positive reals
$*$	multiplication
$^{-1}$	inverse
1	one

The 6-element monoid given by a table above.

$$\langle\, \mathbb{Z}; +, 0 \,\rangle$$

$$\langle\, \mathbb{Q}^{+}; \cdot, 1 \,\rangle$$

$$\langle\, \mathbb{R}^{+}; \cdot, 1 \,\rangle$$

$$\mathbb{Z}_n = \langle\, \{0, 1, \ldots, n-1\}; \oplus, 0 \,\rangle$$

$$\mathbb{Z}_p^* = \langle\, \{1, \ldots, p-1\}; \otimes, 1 \,\rangle$$

Here $x \oplus y$ is $x + y \bmod n$. Unsigned ints in C behave this way for $n = 2^{32}$.

$x \otimes y$ is $x \cdot y \bmod p$ and p is required to be a prime (otherwise we have a monoid, but not a group).

All the examples of groups we have seen so far are commutative.

Could it be true that commutativity follows somehow from the group axioms?

No, there are lots of non-commutative groups. Here is, again, a table of the smallest non-commutative group. It has 6 elements.

$*$	e	r	s	a	b	c
e	e	r	s	a	b	c
r	r	s	e	b	c	a
s	s	e	r	c	a	b
a	a	c	b	e	s	r
b	b	a	c	r	e	s
c	c	b	a	s	r	e

Checking the group properties requires work, as usual, but non-commutativity is easy to see. Note the 3 by 3 square in the upper left hand corner: $\{e, r, s\}$ is a commutative subgroup.

Though the equations are nearly trivial, just to understand their finite models took a huge amount of effort (classification of finite simple groups, completed 1985, some 15,000 pages of proofs; initially complete with minor errors).

The theorem says in essence that all finite groups can be decomposed into just 5 basic types. Two of these basic groups are very straightforward, e.g., cyclic groups of prime order or alternating groups, two are a bit more messy (certain Lie-type groups) but the 5th class is utterly bizarre: it consists of 26 rather strange groups.

The largest one is called the monster. It is a group of rotations, albeit in 196883-dimensional space. Its size is

$$808017424794512875886459904961710757005754368000000000 \approx 8.08 \cdot 10^{53}$$

How on earth does this number come out of the group axioms?

On the other hand, any purely equational set of axioms always has a model.

Proposition

Every system of equations has a model.

Proof.

A trivial model can be constructed by choosing $A = \{\bullet\}$ and setting all functions to be constant with value $\bullet$.

□

Of course, this trivial model is utterly useless, we need a carrier set of size at least 2 for an equation to be interesting. Things change drastically if we allow inequalities or implications between equations. For example, the field axioms force the existence of at least two elements zero and one.

Exercise

Meditate deeply on the nature of $\bullet$.

Suppose we have a system E of equations over some graded alphabet Σ. We already know that the term model $\mathcal{T}$ works when there are no equations in E.

Otherwise, define the following equivalence relation $\widetilde{E}$ on $\mathcal{T}$. $\widetilde{E}$ is the least equivalence relation that

- contains E, construed as a relation on $\mathcal{T}$, and
- is closed under substitution in the following two ways:
  - if $s \mathrel{\widetilde{E}} t$ then $s[u/x] \mathrel{\widetilde{E}} t[u/x]$, and
  - if $s \mathrel{\widetilde{E}} t$ then $u[s/x] \mathrel{\widetilde{E}} u[t/x]$.

  Here u is any term and x any variable.

So $\widetilde{E}$ can be a bit bigger than E, it includes all "conclusions" that can be drawn from the equations in E.

We write $\mathcal{T}_E$ for the structure obtained from $\mathcal{T}$ by factoring with respect to $\widetilde{E}$. It is easy to see that $\mathcal{T}_E$ is again a suitable structure.

Theorem

$\mathcal{T}_E$ is the free E-structure generated by Var*.*

More precisely, if we have any model $\mathcal{A}$ of E and a map $\sigma : \mathrm{Var} \to A$ then there is a unique homomorphism extending σ from $\mathcal{T}_E$ to $\mathcal{A}$.

As we have seen for the associative law, it may well happen that the validity of one equation in a structure entails the validity of others. More generally, suppose E is a system of equations and e a single equation.

Definition

e is a semantic consequence of E if for all structures $\mathcal{A}$ such that $\mathcal{A} \models E$ we also have $\mathcal{A} \models e$.

Notation:

$$E \models e$$

Example

Equation $x * (y * (z * u)) \approx ((x * y) * z) * u$ is a semantic consequence of $x * (y * z) \approx (x * y) * z$.

But $x * y = y * x$ is not a consequence of associativity.

The two-element Boolean algebra $\mathbb{B}$ is just one particular model of the axioms (BA). One might wonder what the relationship is between equations that hold in $\mathbb{B}$ versus equations that are universally valid in all Boolean algebras.

Theorem

An equation is valid in $\mathbb{B}$ if, and only if, it is valid in all Boolean algebras.

Proof. Suppose $s \approx t$ is an equation that fails to hold in some algebra B. By Stone's theorem we may safely assume that $B \subseteq \mathfrak{P}(A)$ is a field of sets.

So we have for some valuation σ that $[\![s]\!]_\sigma \neq [\![t]\!]_\sigma \subseteq A$. Without loss of generality assume $a \in [\![s]\!]_\sigma - [\![t]\!]_\sigma$.

Define a homomorphism $f : B \to \mathbf{2}$ by $f(X) = \begin{cases} 1 & \text{if } a \in X, \\ 0 & \text{otherwise.} \end{cases}$

f shows that $s \approx t$ is not valid in $\mathbb{B}$. □

- Equational Logic

2. Equational Reasoning

Let us try to pin down how these arguments really work: we repeatedly substitute equal terms for equal terms and use the obvious properties of equality.

The given equations in (BA) and the basic properties of equality are the only source for the equalities that can be used in the argument.

As a consequence, the argument is easy to check for correctness: we just have to identify which axiom and what substitution was used.

In an annotated proof these would also be given explicitly so the correctness check comes down to pattern matching.

Alas, the real problem is not to check the proof for correctness but to find it in the first place.

Following Gödel's approach, we can define an equational proof of e (over some system E) to be a sequence of equations

$$e_1, e_2, \ldots, e_{n-1}, e_n = e$$

where each e_i is either in E, is a trivial identity $s \approx s$ or can be derived from a previous equation e_j, $j < i$ by a simple rule to be spelled out below. For example, we may switch the two sides of an equation.

There is one small exception: for transitivity of equality we need two premisses.

Also, a better representation of an equational proof (at least for humans) is a tree with e as the root. A node is the consequence of its children.

First we need rules that express the fact that equality is reflexive, symmetric and transitive.

$$\frac{-}{s \approx s} \qquad \frac{s \approx t}{t \approx s} \qquad \frac{s \approx t \quad t \approx u}{s \approx u}$$

The first rule has no premiss, the second one, and the last two.

Also, given a system of equations E as assumptions, we are allowed to use any equation $s \approx t \in E$:

$$\frac{-}{s \approx t}$$

Note that these rules are not quite enough: think about the associativity example from above.

Two rules relate to substitutions (instantiation and congruence):

$$\frac{s \approx t}{s\theta \approx t\theta} \qquad \frac{s \approx t}{f(\ldots s \ldots) \approx f(\ldots t \ldots)}$$

In other words, we are allowed to substitute arbitrary terms for variables on both sides of an equation $s \approx t$, thereby generating an instance $s\theta \approx t\theta$ of the equation. Here θ is a substitution, a map Var $\to \mathcal{T}$.

Moreover, if we have $s \approx t$ then we can replace s by t in an arbitrary term and obtain another equation (the ellipses here are sloppy notation for the same sequence of terms on botch sides).

Exercise

Check which of these rules were used in the equational proofs given so far.

Definition

e is a syntactic consequence of E if e can be derived from E by finitely many applications of these rules.

Notation:

$$E \vdash e$$

Theorem

The deduction system for equations is sound: every syntactic consequence of E is also a semantic consequence.

Proof. Straightforward induction on the length of the deduction. □

Exercise

Carry out a the details in the soundness proof.

Soundness is just a sanity check: our deduction system contains no rules that would produce invalid conclusions.

But how do we know that no rules are missing? Suppose $E \models e$. Is there always an equational proof of e from E? The answer is given by a famous theorem:

Theorem (Birkhoff 1935)

The deduction system for equations is complete: every semantic consequence of E is also a syntactic consequence.

Hence

$$E \models e \iff E \vdash e$$

So suppose we have a set of equations E over some graded alphabet Σ.

How do we check whether an equation e is a semantic consequence of E?

By Birkhoff's theorem, we can simply enumerate all possible equational proofs with axioms in E. If e follows from E we will discover a proof in finitely many steps. Hence, the problem is semi-decidable.

Of course, if were to implement proof search we would not simply generate potential proof sequences blindly but would try to "get from E to e" in some systematic way.

One seemingly reasonable approach would be try to generate longer and longer terms that ultimately lead to the desired target equation e.

Unfortunately, in the presence of the sufficiently ill-behaved axioms no such simple strategy is going to work: the proof may involve terms of arbitrary size.

Theorem

In general, it is undecidable whether e is a semantic consequence of E.

Furthermore, undecidability already rears its ugly head when the signature is very simple.

Here is one particularly small example, due to G. S. Tsentin.

There are 5 constants, a, b, c, d, e. We omit the associativity axiom and focus on axioms involving the constants.

4 axioms deal with commutation:

$$ac = ca, ad = da, bc = cb, bd = db.$$

The others are a bit strange:

$$abac = abacc, eca = ae, edb = be.$$

It is undecidable whether an equation between ground terms (no free variables) follows from these axioms.

So we have a deduction system that is perfect in principle but has computational problems. Indeed, in practice it turns out that using Birkhoff's rules directly often leads to arguments that are quite long and are often very, very tedious to discover – humans are not particularly good at this kind of task.

It helps to have a few other tools lying around that can help in constructing proofs. For example, here is a lemma that helps in establishing equalities in Boolean algebras.

Lemma (BA)

The complement $\overline{x}$ *is unique in the sense that* $x + y = 1$ *and* $x \cdot y = 0$ *implies* $y = \overline{x}$.

So instead of trying to prove $y = \overline{x}$ directly we can attempt to prove $x + y = 1$ and $x \cdot y = 0$ – which may well be easier.

Of course, we have to prove first that the lemma holds, using only equational reasoning.

Proof. Assume $x + y = 1$ and $x \cdot y = 0$.

$$\begin{aligned} y &= y \cdot 1 \\ &= y \cdot (x + \overline{x}) \\ &= y \cdot x + y \cdot \overline{x} \\ &= x \cdot y + y \cdot \overline{x} \\ &= 0 + y \cdot \overline{x} \\ &= x \cdot \overline{x} + y \cdot \overline{x} \\ &= (x + y) \cdot \overline{x} \\ &= 1 \cdot \overline{x} \\ &= \overline{x} \end{aligned}$$

Claim

$(\mathrm{BA}) \models \overline{\overline{x}} = x.$

Proof. By the lemma, we only have to show that x behaves like the complement of $\overline{x}$. But

$$\overline{x} + x = x + \overline{x} = 1 \qquad \text{and} \qquad \overline{x} \cdot x = x \cdot \overline{x} = 0.$$

Done! □

And so on and so forth.

Exercise

Establish all the other equations for Boolean algebras listed above.

In the 1930s E. V. Huntington discovered a very simple axiomatization for Boolean algebras. The signature is reduced to $+$ and $^-$ (since product can be defined in terms of these two via Morgan's law).

$$(x + y) + z = x + (y + z)$$
$$x + y = y + x$$
$$x = \overline{\overline{x} + y} + \overline{\overline{x} + \overline{y}}$$

According to Huntington's axioms, the operation plus is associative and commutative, and the relation between plus and complement is regulated by the third axiom.

It is easy to see that the third axiom holds in any Boolean algebra, but it requires more work to establish the converse.

In 1933 H. Robbins conjectured that the third axiom can be replaced by the slightly more cryptic:

$$x = \overline{\overline{x+y} + \overline{x+\overline{y}}}$$

Again it is easy to check that this equation holds in any Boolean algebra, but it is far from clear that the opposite direction also works.

It was shown in the 1970s that to prove this conjecture it suffices to show that Robbins' axioms imply the double negation property $\overline{\overline{x}} = x$ but no one knew how to do this.

Robbins' Conjecture was not proven until 1996, and then by the automatic prover EQP, not a human. See

`http://www.mcs.anl.gov/home/mccune/ar/robbins`

One annoying feature of this machine generated proof is that it provides no insight: it consists of just 16 steps of equational reasoning, applied to uncomfortably large expressions. The proof search took 8 days on a workstation.

It is a modest challenge even to just pretty-print the proof.

So, don't worry if you have problems finding some of these proofs.

One might wonder whether there is a way to axiomatize Boolean algebras with just a single equation. Surprisingly, the answer is yes.

To keep notation bearable, let us write xy for the NAND operation: $\neg(x \wedge y)$.

Here is a complete axiomatization for Boolean algebras:

$$(((xy)z)(x((xz)x))) = z$$

It is quite difficult to recover the standard axioms from this.