Algebraic Systems

Formal Modeling (refrain)

Formal = Atoms + Formations + Transformations + Axioms

A ***formal system*** (a mathematical system) consists of

1. several sets of labels (for objects, functions, relations) called constants,
2. rules for building compound sentences (or equations or expressions), and
3. rules for evaluating and simplifying compound expressions.
4. some axioms or assumptions which assert equivalence sets

A ***calculus*** is a formal transformation system with variables.

Mathematical Data Structures

truth values	`0,1`	arithmetic of logic
propositions	`a,b,c`	algebra of logic
sets	`{},{a},{a,b}`	set theory
ordered pairs	`(a,b),(a,c)`	functions, relations
nested pairs	`((a,b),c),((a,c),d)`	binary functions and relations
nested pairs	`(a,(a,b)),(b,(b,c))`	graphs

Morphism Functions

A ***function*** is a constrained relation between two sets, the Domain and the Range.

An ***algebraic system*** is a Set (the Domain of a function) and at least one binary function on that Set: `(S,f)` where `S` is the Domain, and `f` is a binary function.

A ***homomorphism*** is a special type of function which maps one algebraic system onto another. Given a system `(S,f)` and a system `(T,g)`, the homomorphic function `h` is:

```
All s1, s2 inS .  h(f(s1,s2)) = g(h(s1),h(s2))
```

The morphism function **h** preserves the structure of the two systems. When it exists, we know that the two systems are in some way *functionally identical*. Isomorphic systems are algebraically indistinguishable.

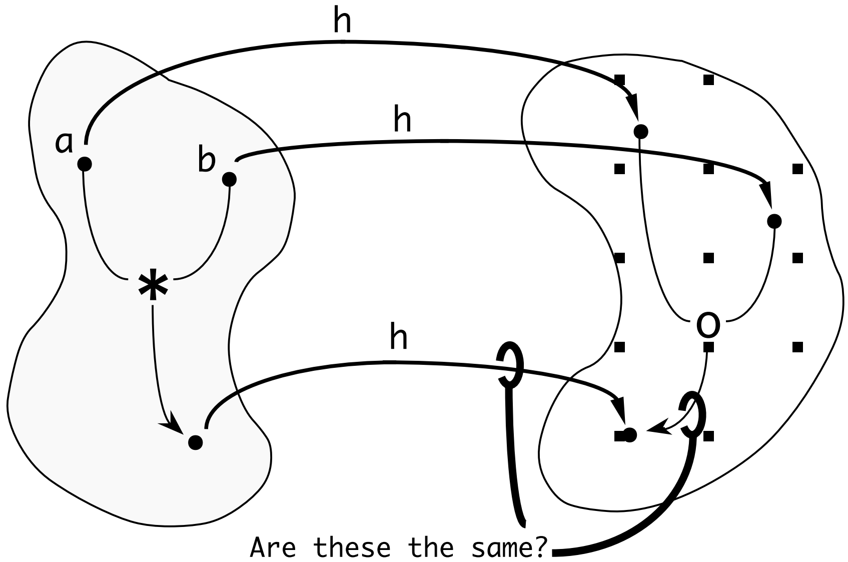
Other types of morphism functions preserve other types of functional structure.

Epimorphic: h preserves the onto characteristic.

Monomorphic: h preserves the one-to-one characteristic

Isomorphic: h preserves one-to-one correspondence

Morphism Diagram



Examples of Morphic Systems

*h[a*b] = h[a] • h[b]*

Affine

System 1: (integers,+)
System 2: (integers,+)

Morphism h[x] = 2x

Proof: h[a+b] = 2(a+b) = 2a + 2b = h[a] + h[b]

Logarithm

System 1: (integers,+)
System 2: (reals,*)

Morphism h[x] = e^x

Proof: h[a+b] = e^(a+b) = e^a * e^b = h[a] * h[b]

Signs in Multiplication

System 1: (integers,+)
System 2: ({1,-1},*)

Morphism
```
h[x] =  1 if x is even
     = -1 if x is odd
```

Proof:
```
case a,b even:    h[a+b] =  1    h[a]*h[b] = even*even = 1
case a,b odd:     h[a+b] =  1    h[a]*h[b] = odd*odd = even = 1
case a,b differ:  h[a+b] = -1    h[a]*h[b] = odd*even = odd = -1
```

Group Theory

Algebraic systems (`(S,f)`, where `S` is a set and `f` is a binary function on that set) can be classified into groups having similar structural characteristics. This additional level of abstraction is called **group theory**, or modern algebra.

The essential distinguishing characteristics of algebraic systems `(S,f)`:

Let `a,b,c inS` and `e`, the identity element, `inS`

Closed binary operation: `f(a,b) = c`

Associativity: `f(f(a,b),c) = f(a,f(b,c))`

Identity element: `Exists e inS. f(e,a) = f(a,e) = a`

Inverse element: `Exists y inS. f(a,y) = f(y,a) = e`

Commutativity: `f(a,b) = f(b,a)`

Types of Algebraic Systems

Groupoid: `(S,f)` such that `S =/= { }`

Loop: Groupoid and

```
All a,b,c in S. if f(a,b) = f(a,c) then b=c
                if f(a,c) = f(b,c) then a=b
```

Semigroup: Groupoid and
`S` is closed under `f`
`f` is associative on `S`

Monoid: Semigroup and
`(S,f)` has an identity element

Group: Monoid and
every element in `S` has an inverse.

Each type can be combined with the commutative property, to give

commutative loop
commutative groupoid
commutative semigroup
commutative monoid
commutative group (boolean algebra is an example in this category)

Boolean Algebra

Boolean algebra is an algebraic system, `{K,A,V,'}` consisting of

K	a set of elements
A	the *meet* operation
V	the *join* operation
'	the *complement* operation

Boolean Algebra Axioms

Let • be either AND or OR:

associative	a·(b·c) = (a·b)·c
commutative	a·b = b·a
distributive	a·(b*c) = (a·b)*(a·c)
zero element	a V 0 = a
one element	a A 1 = a
complement	a V a' = 0 a A a' = 1

Boolean Algebra Morphisms

Domain	*meet*	*join*	*complement*	*zero*	*one*	*less-than*
Boolean algebra	meet	join	complement	0	1	<
algebra of sets	union	intersection	complement	Phi	Universe	subset
switching circuits	series	parallel	opposite	open	closed	if-then
propositional logic	and	or	not	false	true	if-then
integer divisors	gcd	lcm	largest/x	1	largest	divides