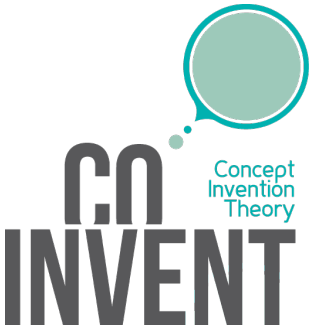


D6.2
Mathematical blends in the heterogeneous KR&R framework

Authors	Ewen Maclean, Alan Smaill, Danny Gómez-Ramírez, Jacques Fleuriot
Reviewers	Marco Schorlemmer

Grant agreement no.	611553
Project acronym	COINVENT - Concept Invention Theory
Date	October 1, 2015
Distribution	PU

Disclaimer

The project COINVENT acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open Grant number 611553.

Abstract

This deliverable presents a series of developments in mathematics and related domains that make use of some parts of the computational machinery underlying work in the Coinvent project. In particular, this work makes use of the HETS system to validate correctness of the blends described. We have also relied on the ONTOHUB to support uniform library and look-up mechanisms within these developments. Most of the work is described in publications, which are referenced as appropriate.
Keyword list: **concept invention**, **mathematical blends**, **creativity**

Executive Summary

- Section 1 gives a general background;
- Section 2 describes some simple examples where simple combinatorial combinations via blending give mathematically interesting results.
- Section 3 looks at the blending of finite and infinite notions so as to achieve a consistent theory with aspects of both.
- Section 4 looks at a development from algebra that produced a "surprising" definition of an algebraic concept.
- Section 5 looks at the use of blending to identify analogies between theories that can serve as aids in proof tasks.
- Section 6 describes a more extended development that builds the main concepts of Galois theory from a small numbers of simple input concepts.

Contents

1 Introduction

This document presents a series of developments using conceptual blending in mathematics, widely understood. In addition, a couple of developments that go in different ways outside of the mathematical domain are also reported.

The various cases presented use the Coinvent computational resources in different ways.

- In all cases, the logical theories studied are given in the CASL language;
- Computation of the blended theory from input theories and a generic theory is realised by HETS.

On the other hand:

- In some cases, the generic theory is computed by the HDTP module, perhaps using successive solutions for such generic theory.
- In other cases, the generic theory is given simply in terms of type declarations common to the input theories.
- In other cases, the generic theory is given by hand.

This will be clarified in each case.

The developments here lean on notions of search which are of course central to the aims of the project. Again, this is apparent to different extents in the different developments.

Working in the richer representation language that is given by CASL is needed in the mathematical domain if we are to work with humanly-comprehensible theories. This rules out direct use of the amalgams approach used elsewhere in the project, and reliance on HDTP to find generic theories.

This brings with it issues of undecidability in the worst case, and infeasibility of consistency checking in typical cases. A discussion of search issues in this context, with a suggested semi-effective algorithm is given in Martinez et al. (2016).

There is work involving the *reformation* of logical theories carried out in association with Coinvent work, not further described in this document, that is discussed in A. Bundy and Mitrovic (2016) and Alan Bundy and Maclean (2016).

2 Incremental Theory Building

This section presents work done on constructing mathematical theories by applying blending recursively, using simple initial "seed" theories. Here we present examples in this section using simple concepts inspired from the notion of *image schema*. A more complex development is described in §6.

A main challenge in this work is controlling the search in blending. We describe the challenges here in §2.2, where we discuss automation.

2.1 Simple Theory Building

Let us introduce some very simple concepts in CASL syntax. For example a simple version of the natural numbers:

spec NATSUC =
  sort *Num*
  ops $zero : Num$;
    $\_\_+\_\_ : Num \times Num \to Num$;
    $suc : Num \to Num$
  pred $\_\_<\_\_ : Num \times Num$
  $\forall\, x, y, z : Num$
  • $\neg\, x < x$
  • $x < y \wedge y < z \Rightarrow x < z$
  • $x < y \vee x = y \vee y < x$
  • $x < y \Leftrightarrow suc(x) = y \vee \exists\, sx : Num \bullet suc(x) = sx \wedge sx < y$
  • $suc(x) = y \wedge suc(x) = z \Rightarrow y = z$
  • $suc(x) = suc(y) \Rightarrow x = y$
  • $\exists\, a : Num \bullet suc(x) = a$
  • $\neg\, suc(x) = zero$
  • $suc(x) = y \Rightarrow zero < y$
  • $x < y \Rightarrow suc(x) < suc(y)$
  • $zero + y = y$
  • $suc(x) + y = suc(x + y)$
end

A notion of magnitude:

spec MAGNITUDE =
  sort X
  pred $\_\_>\_\_ : X \times X$
  op $zerosize : X$
  $\forall\, x, y, z : X$
  • $\neg\, zerosize = x \Rightarrow x > zerosize$
  • $x > y \wedge y > z \Rightarrow x > z$
end

from which we could develop the notion of line:

spec LINE =
MAGNITUDE **with** $X \mapsto Dist$
then sort X
preds $rightof : X \times X$;
$leftof : X \times X$
op $distance : X \times X \to Dist$
$\forall x, y : X$
• $rightof(x, y) \Leftrightarrow leftof(y, x)$
• $\neg\ rightof(x, x) \wedge \neg\ leftof(x, x)$
• $distance(x, y) = distance(y, x)$
• $distance(x, x) = zerosize$
end

We can also construct the concept of rotation by 180 degrees:

spec ROTATION180 =
sorts $Elem, Size$
preds $left : Elem$;
$right : Elem$
ops $origin : Elem$;
$rotate : Elem \times Elem \to Elem$;
$size : Elem \to Size$
$\forall x : Elem$
• $left(x) \Leftrightarrow right(rotate(origin, x))$
• $size(rotate(origin, x)) = size(x)$
• $rotate(origin, origin) = origin$
• $\neg\ left(origin) \wedge \neg\ right(origin)$
• $\neg\ x = origin \Rightarrow \neg\ (left(x) \wedge right(x))$
• $\neg\ x = origin \Rightarrow left(x) \vee right(x)$
end

Given these theories, the challenge is to automatically compute blends, and recursively compute with the given blends to discover new theories which are creative. An algorithm for doing this automatically is given in §2.2. For now in this section we present a simple combination of blends which result in an interesting result - that being a version of the integers. The HETS development graph can be seen in Figure 1.

2.1.1 Blending Naturals with Line

The creativity demonstrated here is the combination of very simple concepts. The first blend to be considered is by blending the Natural numbers with the concept of a line. The arbitrary sort underlying the specification on a line is associated via the Generic space with the sort of the Natural numbers. The blend forms a number line — i.e. a line with a concept of number imposed on it.

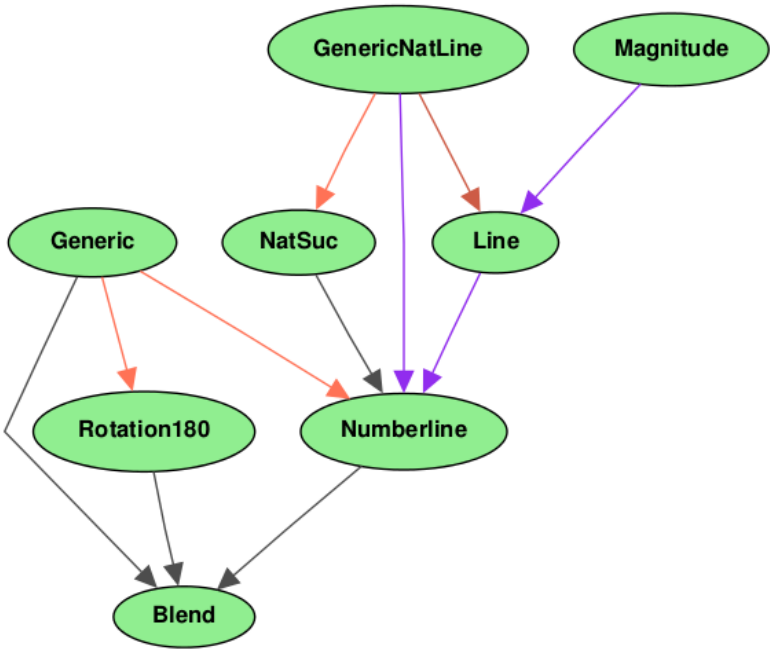


Figure 1: HETS development graph for constructing a version of the integers from simple theories

2.1.2 Blending Rotation with a Numberline

Given a concept describing rotation by 180 degrees, we can construct a blend by associating the concept of a fulcrum with the natural number 0, and identifying the sort of Natural numbers with that of rotation by 180 degrees. This constructs a numberline which can be reflected about 0 —- thus identifying the concept of a negative number.

2.2 Search and Automation

In order to construct a blend recursively, one need to consider an algorithm for how to search. Figure 2 shows a graph of how theory construction is propagated.

The main branch points in the search are:

- On choice of generic theory (HDTP or Amalgams;
- On revising input theories to logically weaker theories (e.g. if found to be inconsistent on evaluation).

In the examples treated here, the second choice point is explored before trying alternatives to the first (in the case HDTP is used). As soon as output theories become available for subsequent blending, the combinatorics of the sitaution become challenging.

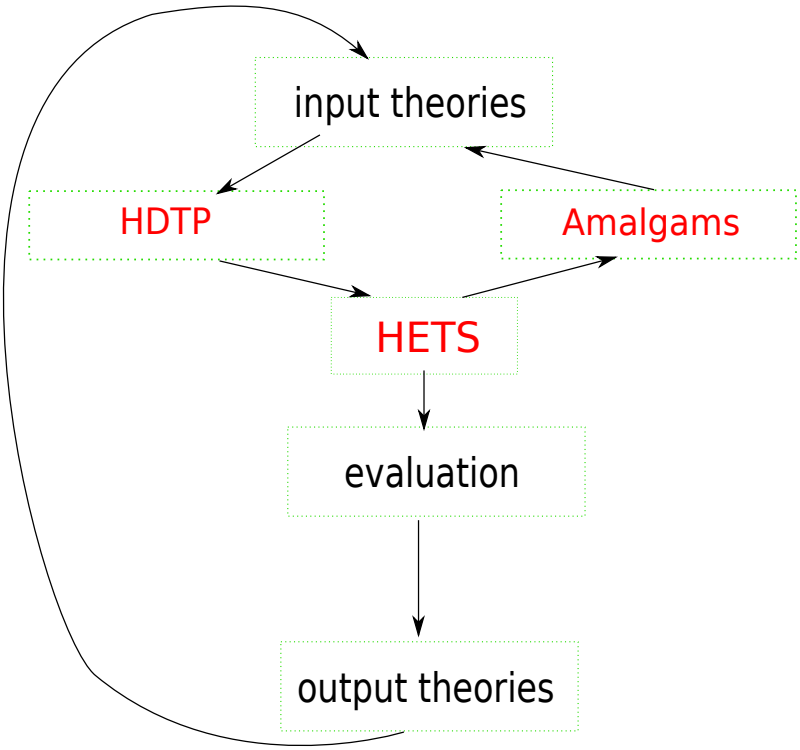


Figure 2: The components of the system used for recursively discovering concepts

3 Notions of Infinity

This work is reported in the publication Bou et al. (2015), linked to the bibliography.

The work of Lakoff and Núñez (2000) provides a wide range of examples of metaphorical reasoning in mathematics, while stressing the embodied cognition involved in basic mathematical experience.

Some of the ideas of Lakoff and Núñez (2000) have been reworked by the authors, with increased emphasis on conceptual blending. In particular, the analysis of mathematical infinity, given in metaphorical form as the "Basic Metaphor of Infinity" (BMI) in Lakoff and Núñez (2000), is represented in blend form in Núñez (2005) as the "Basic Mapping of Infinity" (so, still "BMI").

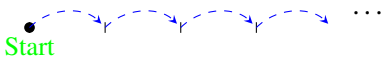
A particular case descibed by Núñez involves reasoning about infinite ordered sets, and transfinite ordinals; he analyses the situation from a cognitive point of view in (nunez93,nunez05).

We show here how this blend works out in our setting. The BMI suggests that the notion of completed infinity, in particular the possibility of transfinite numbers in the sense of Cantor, comes from a blend of the notion of completed, finite process with that of a potentially infinite and endless process.

Thus take two corresponding input spaces, given by CASL specifications **FinEnd** and **Inf** corresponding to the following diagrams

FinEnd:

Inf:

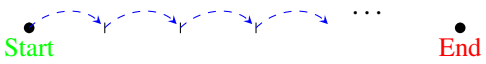


- **FinEnd:** Completed Iterative Processes are those that from some initial state, terminate in a final state after a finite number of state transitions. One such case is chosen.
- **Inf:** Infinite Iterative Processes are those that continue indefinitely to change state.

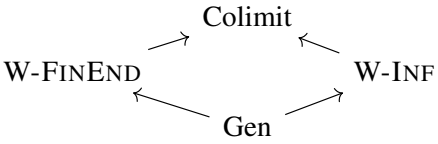
The generic space **Gen** simply identifies the start states, the notion of process step, and the linear ordering of states. This blend is *inconsistent*, for the following two reasons:

1. the number of states is finite (from **FinEnd**), and infinite (from **Inf**);
2. there both is an end state (from **FinEnd**) and is no end state (from **Fin**).

Search through the possibilities of weakening the input spaces by omitting as few axioms as possible among those involved in an inconsistency reveals the possibility of a structure with infinitely many states (from **Inf**) and an end state (from **FinEnd**). Computing the colimit from the weakened input spaces **W-FinEnd**, **W-Inf** gives a theory corresponding to this diagram:



Thus we have a blend as in the earlier examples:



4 Prime Ideals

This work is also published in Bou et al. (2015).

The mathematical reuse of the terminology "prime", originally associated with natural numbers, applied to certain well-behaved subsets of algebraics structures called rings, is not an accident. It shows inter-theory analogy in practice.

We indicate here how this is captured via the blending process. Here, we give a standard mathematical account of the theories involved. The background concepts can be found in Eisenbud (1995). In carrying through identification of a blend between *prime numbers* and *ideals* of commutative rings, the machinery gave a new definition, unknown to the authors, and apparently not generally known to the community of algebraists — this surprise is evidence of hte creativity of the process. We present a first blend involving weakening, followed by a second blend from fuller input spaces, where the emergent concept of CDR appears.

4.1 The first conceptual space

Let $(R, +.*, 0, 1)$ be a commutative ring with unity (see the formal definition and examples in Eisenbud (1995)). Now, R can be understood as the sort containing the elements of the corresponding commutative ring with unity. An ideal I is a subset of R satisfying the following axiom:

$$(\forall i, j \in I)(\forall r \in R)(i + (-j) \in I \wedge r * i \in I).$$

Let us define a unary relation (predicate) *isideal* on the set (sort) of subsets of $P(R)$ corresponding to this definition. Now, we define

$$\mathrm{Id}(R) = \{A \in P(R) : \mathit{isideal}(A)\}.$$

Ideals are "multiplied" using the following definition:

$$I \cdot_\iota J = \left\{ \sum_{k=1}^{n} i_k \cdot j_k : n \in \mathbb{N} \wedge i_1, \ldots, i_n \in I \wedge j_1, \ldots, j_n \in J \right\}.$$

The key property that we want to keep in the blend is the one saying that this operation $\cdot_\iota$ has a neutral element 1_ι, which can be seen as an additional notation for the ring. On the other hand, we want to see the containment relation $\subseteq$ as a binary relation over the sort $\mathrm{Id}(R)$.

Summarizing, our first conceptual space consists of sorts $R, \mathrm{Id}(R)$ and $P(R)$; operations $+, *, 0_R, 1_R, 1_\iota$ and $\cdot_\iota$; and the relations $\subseteq$ and *isideal*.

Let us denote this space by $\mathbb{I}$.

4.2 The second conceptual space

Let $\mathbb{Z}$ be the set of the integer numbers. Here, we choose any partial axiomatization of them including at least the fact that $(\mathbb{Z}, *, 1)$ is a commutative monoid. We define also an upside-down divisibility relation $\lfloor$ defined as $e \lfloor g := g|e$, i.e. there exists an integer c such that $e = c * g$. Let us

define a unary relation *isprime* on $\mathbb{Z}$ as follows: for all $p \in \mathbb{Z}$, $isprime(p)$ holds if $p \neq 1$ and:

$$(\forall a, b \in \mathbb{Z}) \left((ab \lfloor p) \rightarrow (a \lfloor p \vee b \lfloor p)\right).$$

Besides, we define the set (sort) of the prime numbers as

$$Prime = \{p \in \mathbb{Z} : isprime(p)\}$$

In the CASL language, we consider $\mathbb{Z}$ as the sort of the integer numbers, $*$ as a binary operation, *prime* as a predicate and $\lfloor$ as a binary relation, any of them defined over the sort $\mathbb{Z}$. We denote this conceptual space by $\mathbb{P}$.

4.3 The Generic Space

The generic space $\mathbb{G}$ consists of a set (sort) G with a binary operation $*_G$, a neutral element S and a binary relation $\leq_G$.

4.4 The Blending Morphisms

The morphism to $\mathbb{I}$ uses:
$\varphi(G) = \mathrm{Id}(R), \varphi(*_G) = *_\iota, \varphi(S) = 1_\iota$ and $\varphi(\leq_G) = \subseteq$;
the morphism to $\mathbb{G}$ uses:
$\delta(G) = \mathbb{Z}, \delta(*_G) = *, \delta(S) = 1$ and $\delta(\leq_G) = \lfloor$.

4.5 The Axiomatization of the Blending

A straightforward colimit construction based on the input and generic spaces above yields a consistent space with properties inherited both from the prime elements into the integers and from the ideals of commutative rings; one of the concepts is a notion of prime ideals, another is that of CDR[1] Here we describe briefly a weakening of the given spaces that makes the resultant blend more generally applicable.

From the properties defining the integers we transfer into the blend only the fact that $\mathbb{Z}$ is a set with a binary operation $*$ having 1 as neutral element and $\lfloor$ as a binary relation, without taking into account its formal definition.

Now after computing the colimit, we obtain that $G = \mathrm{Id}(S)$ and that any element $P \in G$ (i.e., an ideal of S) satisfies the predicate *isprime* if and only if

$$P \neq S \wedge (\forall X, Y \in G \quad (X \cdot_\iota Y \subseteq P \rightarrow (X \subseteq P \vee Y \subseteq P)).$$

Thus, the predicate *isprime* turns out to be the predicate characterizing the primality of ideals of S and the set (sort) *Prime* turns out to be the set of prime ideals of S.

Using the weakened input spaces, the blending space consists of the axioms assuring that S is a commutative ring with unity, G is the set of ideals of S, *isprime* is the predicate specifying primality for ideals of S and *Prime* is the collection of all prime ideals of S.

[1] A ring R is a Containment Division Ring (CDR) if for all ideals I and J of R, $I \subseteq J$ if and only if J divides I (i.e. there exists an ideal U such that $I = U \cdot_\iota J$).

4.6 Implementation for prime ideals over CDR-s as a blend

The paper also describes a second version of the blend, where the axiom concerning divisibility in the second conceptual space are restored. See the paper for details: it is here that a potentially interesting new definition arises. The resultant blended theory is summarised by the HETS blend below.

This theory defining the blend corresponds to the axioms defining a CDR (S), the set of all its ideals (Generic), the set all its prime ideals (SimplePrime) and a primality predicate (IsPrime). We present in Listing 1 just the theory corresponding to the colimit (omitting details of ring axioms and ideal generation).

[!ht]

```
spec SPEC =
    sorts Generic, RingElt, SimplePrime, SubSetOfRing
    sorts SimplePrime < Generic,IdGeneric < SubSetOfRing
    ops 0,1,S : RingElt
    op __*__ : RingElt × RingElt → RingElt
    op __+__ : RingElt × RingElt → RingElt
    op __x__ : Generic × Generic → Generic
    pred IsIdeal : SubSetOfRing
    pred IsPrime : Generic
    pred __isIn__ : RingElt × SubSetOfRing
    pred gcont : Generic × Generic
    pred __generates__ : RingElt × Generic
    ∀ I : SubSetOfRing • I ∈ Generic ⇔ IsIdeal(I)
    ∀ x : Generic • x x S = x
    ∀ x : Generic • S x x = x
    ∀ A, B : Generic
    • gcont(A, B) ⇔ ∀ a : RingElt • a isIn A ⇒ a isIn B
    ∀ x, y : RingElt • x + y = y + x
    %% and further ring axioms ...
    ∀ I : SubSetOfRing
    • IsIdeal(I)
      ⇔ ∀ a, b, c : RingElt
        • ((a isIn I ⇒ a isIn S) ∧ 0 isIn I)
          ∧ (a isIn I ∧ c isIn S ⇒ c ∗ a isIn I)
          ∧ (a isIn I ∧ b isIn I ∧ c isIn S ∧ b + c = 0
            ⇒ a + c isIn I)
    ∀ a : RingElt; A : Generic
    %% and axioms for generates and x ...
    ∀ x, y : Generic • gcont(x, y) ⇔ ∃ c : Generic • x = y x c
    ∀ p : Generic • p ∈ SimplePrime ⇔ IsPrime(p)
    ∀ p : Generic
    • IsPrime(p)
      ⇔ (∀ a, b : Generic
        • gcont(a x b, p) ⇒ gcont(a, p) ∨ gcont(b, p))
        ∧ ¬ p = S
end
```

Listing 1: Colimit for prime ideals over CDR-s

5 Analogous Lemmas

The work we describe here has been published in Eppe et al. (2015), and is automated using the HETS and Amalgams system components.

5.1 Background explanation and motivation

In Automated Theorem Proving (ATP) one tries to apply mechanical techniques to prove the correctness of mathematical statements. One way of doing this is to apply inference rules of a particular logic, and rewrite rules pertaining to the theory. For example in the natural numbers one can apply the rewrite rules

$$0+N \Rightarrow N \quad (1)$$
$$s(M)+N \Rightarrow s(M+N) \quad (2)$$

Imagine now trying to solve the theorem

$$\forall x,y : x+y = y+x$$

which we do by induction on x. This sets a base case (which we ignore here but is easily solved) and a step case:

$$\forall y' : x+y' = y'+x \vdash s(x)+y = y+s(x)$$

comprised of the induction hypothesis on the left of the turnstile *vdash*, and the induction conclusion on the right. Now we can exploit the rewrite rule (2) to give

$$\forall y' : x+y' = y'+x \vdash s(x+y) = y+s(x)$$

now we can apply the induction hypothesis to give

$$\vdash s(y+x) = y+s(x)$$

which we can solve with a subsequent induction - the details of which we omit here.

5.2 Example

Now let us consider a slightly more complicated example pertaining to the natural numbers. Define a primitively recursive factorial function $fact$, and a tail-recursive factorial function $qfact$ as

$$fact(0) \Rightarrow s(0) \quad (3)$$
$$fact(s(N)) \Rightarrow s(N) \times fact(N) \quad (4)$$
$$qfact(0,M) \Rightarrow M \quad (5)$$
$$qfact(s(N),M) \Rightarrow qfact(N, s(N) \times M) \quad (6)$$

where multiplication is defined as

$$0 \times N \Rightarrow 0 \quad (7)$$
$$s(N) \times M \Rightarrow M + (N \times N) \quad (8)$$

Now let us imagine trying to prove a theorem stating a correctness property about the equivalence of factorial calculations of the $fact$ and $qfact$ functions:

$$\forall n : \mathbb{N}.\ fact(n) = qfact(n, s(0)) \tag{9}$$

Now by induction on n once again the base case is discharged easily and we ignore that here, focussing instead on the the step case:

$$fact(n) = qfact(n, s(0)) \vdash fact(s(n)) = qfact(s(n), s(0))$$

which can be rewrite with (4) and (6) to

$$fact(n) = qfact(n, s(0)) \vdash s(n) \times fact(n) = qfact(n, s(n) \times s(0))$$

which is not easily solveable by induction because the $s(0)$ in the hypothesis cannot be instantiated[2].

In order to solve this we need to employ a generalisation lemma. A lemma is another theorem which is used in the proof using the cut rule of inference:

$$\frac{\Gamma \vdash A \quad \Sigma, A \vdash \Delta}{\Gamma, \Sigma \vdash \Delta}$$

in this case Σ is empty, Δ is the theorem given in (9).

The *creative* step required is to discover the A such that the cut rule can be applied. The process of automatically discovering the A is an unsolved problem. There are various systems which propose techniques but it is undecideable and in general very challenging. For factorial the lemma we can apply which allows the proof to succeed is

$$\forall m, n : \mathbb{N}.\ fact(n) \times m = qfact(n, m)$$

We will refer to this lemma as the "Eureka Lemma" in the rest of the exposition.

5.3 Example with Blending

The idea is to use blending to discover these eureka lemmas for different theories by blending theories for which we know a eureka lemma with one with for which we don't. Let us initially take theory discussed so far about the natural numbers as one theory:

library *a*

spec NATSUC =
  sort *Nat*
  ops *zero* : *Nat*;
    $s : Nat \to Nat$
  op $fact : Nat \to Nat$
  op $qfact : Nat \times Nat \to Nat$

[2]This explanation can be expanded - the reason is complicated

op $plus : Nat \times Nat \to Nat$
op $times : Nat \times Nat \to Nat$
$\forall\, x, y : Nat$
$\bullet\ \exists\, a : Nat \bullet s(x) = a$
$\bullet\ \neg\, s(x) = zero$
$\bullet\ fact(zero) = s(zero)$
$\bullet\ fact(s(x)) = times(s(x), fact(x))$
$\bullet\ qfact(s(x), y) = times(qfact(x, s(x)), y)$
$\bullet\ qfact(zero, x) = x$
$\bullet\ plus(zero, x) = x$
$\bullet\ plus(s(x), y) = s(plus(x, y))$
$\bullet\ times(zero, y) = zero$
$\bullet\ times(s(x), y) = plus(y, times(x, y))$
$\bullet\ fact(x) = qfact(x, zero)$
%theorem to be proved
$\bullet\ times(fact(x), y) = qfact(x, y)$
%eureka lemma
end

now let us consider the theory of lists:

library *a*

spec LIST =
sort *El*
sort *L*
op $nil : L$
op $cons : El \times L \to L$
end

where we introduce some functions and definitions:

op $app : L \times L \to L$
op $rev : L \to L$
op $qrev : L \times L \to L$
$\forall\, x, y : L; h : El$
$\bullet\ app(nil, x) = x$
$\bullet\ app(cons(h, x), y) = cons(h, app(x, y))$
$\bullet\ rev(nil) = nil$
$\bullet\ rev(cons(h, x)) = app(rev(x), cons(h, nil))$
$\bullet\ qrev(nil, x) = x$
$\bullet\ qrev(cons(h, x), y) = qrev(x, cons(h, y))$
$\bullet\ rev(x) = qrev(x, nil)$
%theorem to be proved

Now let us say we want to find the eureka lemma for the theorem $rev(x) = qrev(x, nil)$ which states the equivalence of computation of reversal of a list of elements by a primitively recursive function *rev* and a tail-recursive function *qrev*. We want to use blending of the naturals and this theory of lists to find a colimit which states the eureka lemma for lists.

We want to find a generic space which consists of the constructors of the theory, the functions we want to behave analogously and any auxiliary functions. In this case we want the mapping between the input theories to be

$$\begin{aligned} 0 : Nat &\iff nil : List \\ s : Nat \to Nat &\iff cons : El \times List \to List \\ fact : Nat \to Nat &\iff rev : List \to List \\ qfact : Nat \times Nat \to Nat &\iff qrev : List \times List \to List \\ times : Nat \times Nat \to Nat &\iff app : List \times List \to List \end{aligned}$$

but this is problematic when you look at the types of the constructors of the List and Naturals.

We need to discover a way in which the constructor for the naturals can be made to be equivalent in arity to that of the Lists. Lists of identical elements are isomorphic to the naturals and app in lists is then equivalent to + in the naturals. In the list theory we can compute:

$$app([1,2],[3,4]) = app(cons(1,cons(2,nil)),cons(3,cons(4,nil)) = [1,2,3,4]$$

and

$$app([3,4],[1,2]) = app(cons(3,cons(4,nil)),cons(1,cons(2,nil))) = [3,4,1,2]$$

so app is not commutative. However on lists of identical elements

$$app([1,1],[1,1]) = app(cons(1,cons(1,nil)),cons(1,cons(1,nil)) = [1,1,1,1]$$

and

$$app([1,1],[1,1]) = app(cons(1,cons(1,nil)),cons(1,cons(1,nil))) = [1,1,1,1]$$

where app now behaves exactly as + does on the naturals.

We can exploit this to generalise the theory of naturals by adding an extra argument to the successor function, but only computing with the second argument:

$$s(n) \to s(c,n)$$

where c is some arbitrary element. The step definition of plus, for example, becomes

$$plus(s(c,n),m) = s(c,plus(n,m))$$

so the computation remains in the second element. This way the naturals can be made to look like lists. We then get a theory:

library *a*

spec NATSUC =
    sort *Nat*

sort *Element*
op *canonical_element* : *Element*
ops *zero* : *Nat*;
s : *Element* × *Nat* → *Nat*
op *fact* : *Nat* → *Nat*
op *qfact* : *Nat* × *Nat* → *Nat*
op *plus* : *Nat* × *Nat* → *Nat*
op *times* : *Nat* × *Nat* → *Nat*
∀ *x*, *y* : *Nat*
• ∃ *a* : *Nat* • *s*(*canonical_element*, *x*) = *a*
• ¬ *s*(*canonical_element*, *x*) = *zero*
• *fact*(*zero*) = *s*(*canonical_element*, *zero*)
• *fact*(*s*(*canonical_element*, *x*))
= *times*(*s*(*canonical_element*, *x*), *fact*(*x*))
• *qfact*(*s*(*canonical_element*, *x*), *y*)
= *times*(*qfact*(*x*, *s*(*canonical_element*, *x*)), *y*)
• *qfact*(*zero*, *x*) = *x*
• *fact*(*x*) = *qfact*(*x*, *zero*)
%**theorem** requiring lemma
• *times*(*fact*(*x*), *y*) = *qfact*(*x*, *y*)
%**eureka** lemma
• *plus*(*zero*, *x*) = *x*
• *plus*(*s*(*canonical_element*, *x*), *y*)
= *s*(*canonical_element*, *plus*(*x*, *y*))
• *times*(*zero*, *y*) = *zero*
• *times*(*s*(*canonical_element*, *x*), *y*) = *plus*(*y*, *times*(*x*, *y*))
end

and we can now define a generic space which is correctly typed:

library *a*

spec GEN =
sorts *H*, *G*
op *null* : *G*
op *constructor* : *H* × *G* → *G*
op *recfunc* : *G* → *G*
op *qrecfunc* : *G* × *G* → *G*
op *auxfunc* : *G* × *G* → *G*
end

Now the colimit contains the lemma as part of the definition

$$\forall x, y : List(El)\ .app(rev(x), y) = qrev(x, y)$$

which is precisely the "Eureka Lemma" we need. We then need to remove some axioms from the naturals so that the colimit theory is consistent.

5.4 Generalisation Required

In order to render the two theories of Nat and List similar enough that signature morphisms can be defined between a generic space, we need to modify the definitions of the functions. The constructor for naturals is extended with a generic type, so that the constructor function $s(n)$ (of arity 1) becomes $s(g,n)$ where g is an element of a new type $\mathbb{G}$ which is a newly introduced general type:

$$s : \mathbb{N} \to \mathbb{N} \quad \longrightarrow \quad s : \mathbb{G} \to \mathbb{N}$$

Now we must extend the definitions of any functions which are defined according to these constructors. For example the definition of plus and times becomes:

$$\begin{aligned} \forall x,y : \mathbb{N}.\ plus(s(x),y) = s(plus(x,y)) \quad &\longrightarrow \quad \forall x,y : \mathbb{N}; g : \mathbb{G}.\ plus(s(g,x),y) = s(plus(g,x+y)) \\ \forall x,y : \mathbb{N}.\ times(s(x),y) = y + times(x,y) \quad &\longrightarrow \quad \forall x,y : \mathbb{N}; g : \mathbb{G}.\ times(s(g,x),y) = y + times(x,y) \end{aligned}$$

The element g always either is returned identically or removed from the definition. That is to say, whenever $s(g,n)$ appears, the first argument is *always* simply g.

Let us define a transformation $\mathscr{T}$ formally for this generalisation. We could write for any function:

$$\begin{aligned} &\mathscr{T}(\mathscr{F}(\vec{X}_i) = \mathscr{G}(\vec{Y}_i)) \equiv \\ &\quad \forall s(a_j) \in \vec{X}_i; s(b_k) \in \vec{Y}_i . \mathscr{F}(\mathscr{T}(\vec{X}_i \circ (s(g,a_j)/s(a_j)))) = \mathscr{G}(\mathscr{T}(\vec{Y}_i \circ (s(g,b_k)/s(b_k)))) \end{aligned}$$

6 Galois Theory

Here we summarise work showing how a fairly small combination of belnding steps is sufficient to build the complex, abstract notions that belong to the mathemtical theory associated wit Evariste Galois. A version of this work is published as Gomez-Ramirez, 2015. It extends the work on prime ideals described above. All this development has been checked via the *hets* system.

This work shows that it is possible to generate formally four of the most fundamental notions of Field and Galois Theory by means of nine recursively generated blends, starting from five basic mathematical notions coming from sev eral areas of mathematics as group theory, abstract algebra and topology.

As an example, one of the given input spaces is given by:

Definition 1 *A pointed (abelian) group is a set B with an binary operation $*$ and a distinguished element $b \in B$ such that $(B \setminus \{b\}, *_{|B\setminus\{b\}\times B\setminus\{b\}})$ is an (abelian) group, and $b*c = c*b = b$ for all $c \in B$.*

To summarise, the statring point is from the following five basic mathematical concepts, all of them supported by well-known examples and used very often implicitly and explicitly in modern mathematics: Abelian Group, Pointed Abelian Group, Distributive Space, Action of a group over a Set and Fixed Point Space.

Then each of these concepts are blended iteratively using very simple generic spaces inducing simultaneously the simplest kinds of analogical relation between the corresponding input spaces, namely, relations given by renaming sorts corresponding to sets, functions and relations. So, after doing 9 times the operation of conceptual blending with a formalization of colimits of theories in many-sorted first order logic, we obtain 4 of the most fundamental concepts of Fields and Galois theory, i.e., the mathematical notions of Fields, Field Extension, Group of Automorphisms of a Field and the Group of Automorphisms of a Field Extension fixing the base field. This last concept coincides with the notion of Galois Group when the corresponding extension is a Galois extension. We make and run explicitly all the related implementation in Hets

The figure 3 gives an idea of how this fits together.

This gives some foundational mathematical ‘evidence’ that formal conceptual blending plays a central role as a concrete ‘meta-mathematical’ operation allowing us to model more and more aspects of mathematical creativity. Now, we are not talking any more about isolated mathematical concepts but about the conceptual production of entire theories with formal conceptual blending.

While the small size of the theory graph is encouraging here, at this stage the complementary question of how to control a blending process in the face of huge freedom of choice is still to be addressed, for this sort of example. It is not claimed that the process as described is a plausible model of human mathematical invention.

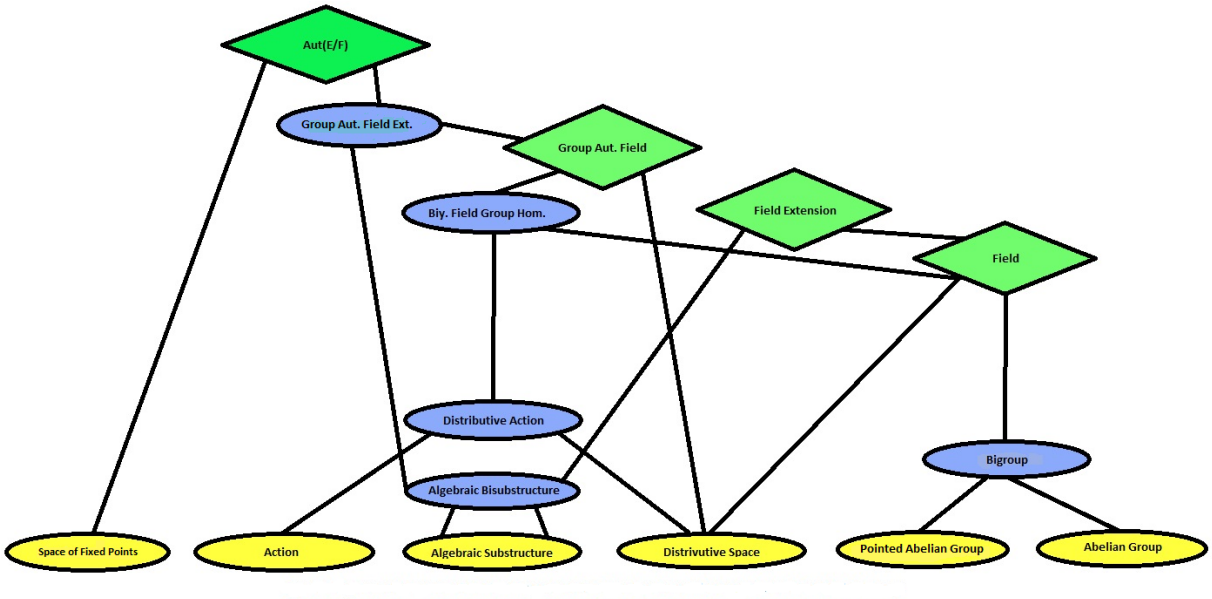


Figure 3: Diagrammatic Representation for the Generation of four Fundamental Conc epts of Fields and Galois Theory through Formal Conceptual Blending.

7 Conclusion

We have described a series of mathematical and related developments carried out in the course of the COINVENT project. These developments have made use, to different extents, of the various tools available to us (HETS, HDTP, Amalgams, and ONTOHUB as a flexible repository).

In so doing, we have used different search strategies, and representations, all within the framework of conceptual blending via categorical push-outs, instantiated in different ways.

The examples range from theories of simple data-structures to the advanced mathematics of Galois theory. This provides a proof of concept in this domain, showing that conceptual blends in mathematics are explicable in this way. More work is needed to evaluate the potential of this approach as support for creative mathematics — the new definition arising from the work in Galois theory gives grounds for hope in this direction.

References

Bou, F. et al. (2015). 'The role of blending in mathematical invention'. In: *Proceedings of the Sixth International Conference on Computational Creativity*. Ed. by H. Toivonen et al. Utah, USA: Brigham Young University, pp. 55–62.

Bundy, A. and B. Mitrovic (2016). *Reformation: A Domain-Independent Algorithm for Theory Repair*. Tech. rep. University of Edinburgh.

Bundy, Alan and Ewen Maclean (2016). 'The Use of Reformation to Repair Faulty Analogical Blends'. In: *UK Ontology Network 2016*. Newcastle. URL: http://www.research.ed.ac.uk/portal/files/25102904/poster.pdf.

Eisenbud, David (1995). *Commutative Algebra with a View Toward Algebraic Geometry*. Vol. 150. Graduate Texts in Mathematics. Springer-Verlag.

Eppe, M. et al. (2015). 'ASP, Amalgamation and the Conceptual Blending Workflow'. In: *Proceedings of the thirteenth international conference on Logic Programming and Nonmonotonic Reasoning*. Ed. by Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynksi. Vol. 9345. LNAI. Lexington USA: Springer.

Gomez-Ramirez, D. (2015). 'Conceptual Blending as a creative meta-generator of mathematical concepts: Prime Ideals and Dedekind Domains as a Blend'. In: *Proceedings of the workshop "Computational Creativity, Concept Invention, and General Intelligence"*. Ed. by T. R. Besold et al. Vol. 2-2015. PICS. Universität Osnabrück.

Lakoff, George and Rafael Núñez (2000). *Where Mathematics Comes From: How the Embodied Mind Brings Mathematics into Being*. New York: Basic Books.

Martinez, M. et al. (2016). 'Theory blending: Extended algorithmic aspects and examples'. In: *Annals of Mathematics and Artificial Intelligence*. URL: http://homepages.inf.ed.ac.uk/smaill/martinezEtAl16.pdf.

Núñez, Rafael (2005). 'Creating Mathematical Infinities: The Beauty of Transfinite Cardinals'. In: *Journal of Pragmatics* 37.10, pp. 1717–1741. URL: http://www.cogsci.ucsd.edu/~nunez/COGS260/JoP_InfinityR.pdf.