Theory Presentation Combinators

Jacques Carette, and Russell O'Connor

McMaster University

CICM 2012, Bremen, Germany

12th July 2012

Motivation

- As part of MathScheme, we wish to efficiently encode mathematical knowledge.
  1. Efficient for the library developper
  2. Efficient for the user
  3. Efficient for processing
- Focus first on Theory Presentations
  - Over a (dependently) typed expression language
  - Syntax for meaningful content

Theories

```
Monoid := Theory {
  U : type;
  * : (U, U) -> U;
  e : U;
  axiom rightIdentity_*_e : forall x:U. x*e = x;
  axiom leftIdentity_*_e : forall x:U. e*x = x;
  axiom associative_* : forall x,y,z:U. (x*y)*z=x*(y*z)
}
```

Theories

```
Monoid := Theory {
  U : type;
  * : (U, U) -> U;
  e : U;
  axiom rightIdentity_*_e : forall x:U. x*e = x;
  axiom leftIdentity_*_e : forall x:U. e*x = x;
  axiom associative_* : forall x,y,z:U. (x*y)*z=x*(y*z)
}

CommutativeMonoid := Theory {
  U : type;
  * : (U, U) -> U;
  e : U;
  axiom rightIdentity_*_e : forall x:U. x*e = x;
  axiom leftIdentity_*_e : forall x:U. e*x = x;
  axiom associative_* : forall x,y,z:U. (x*y)*z=x*(y*z)
  axiom commutative_* : forall x,y,z:U. x*y=y*x
}
```

Theories

```
Monoid := Theory {
  U : type;
  * : (U, U) -> U;
  e : U;
  axiom rightIdentity_*_e : forall x:U. x*e = x;
  axiom leftIdentity_*_e : forall x:U. e*x = x;
  axiom associative_* : forall x,y,z:U. (x*y)*z=x*(y*z)
}

AdditiveMonoid := Theory {
  U : type;
  + : (U, U) -> U;
  0 : U;
  axiom rightIdentity_+_0 : forall x:U. x+0 = x;
  axiom leftIdentity_+_0 : forall x:U. 0+x = x;
  axiom associative_+ : forall x,y,z:U. (x+y)+z=x+(y+z)
}
```

Theories

```
Monoid := Theory {
  U : type;
  * : (U, U) -> U;
  e : U;
  axiom rightIdentity_*_e : forall x:U. x*e = x;
  axiom leftIdentity_*_e : forall x:U. e*x = x;
  axiom associative_* : forall x,y,z:U. (x*y)*z=x*(y*z)
}

AdditiveCommutativeMonoid := Theory {
  U : type;
  + : (U, U) -> U;
  0 : U;
  axiom rightIdentity_+_0 : forall x:U. x+0 = x;
  axiom leftIdentity_+_0 : forall x:U. 0+x = x;
  axiom associative_+ : forall x,y,z:U. (x+y)+z=x+(y+z)
  axiom commutative_+ : forall x,y,z:U. x+y=y+x
}
```

Combinators for theories

Extension:

```
CommutativeMonoid := Monoid extended by {
  axiom commutative_* : forall x,y,z:U. x*y=y*x }
```

Combinators for theories

Extension:

```
CommutativeMonoid := Monoid extended by {
  axiom commutative_* : forall x,y,z:U. x*y=y*x }
```

Renaming:

```
AdditiveMonoid := Monoid[ * |-> +, e |-> 0 ]
```

Combinators for theories

Extension:

```
CommutativeMonoid := Monoid extended by {
  axiom commutative_* : forall x,y,z:U. x*y=y*x }
```

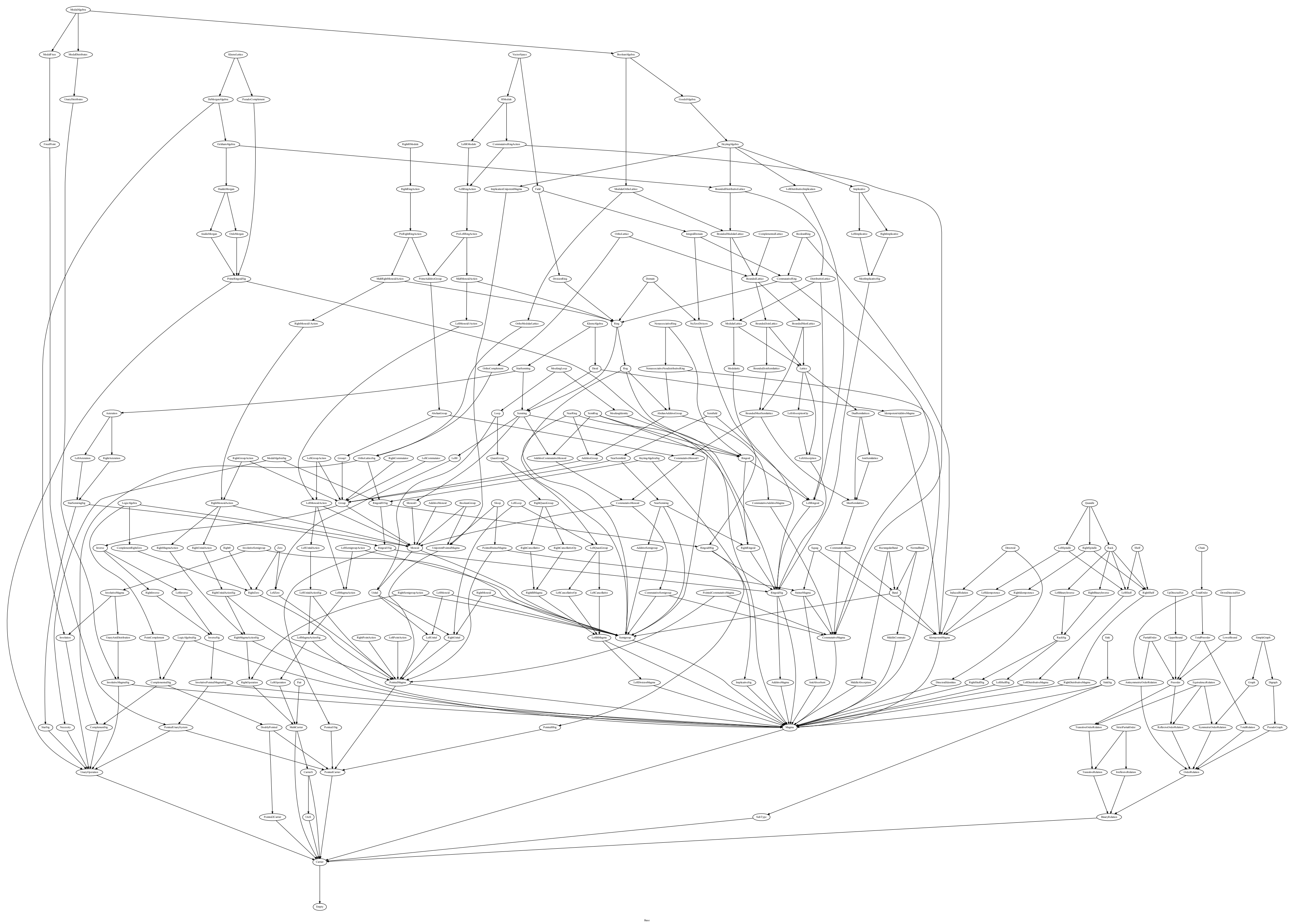
Renaming:

```
AdditiveMonoid := Monoid[ * |-> +, e |-> 0 ]
```

Combination:

```
AdditiveCommutativeMonoid :=
  combine AdditiveMonoid, CommutativeMonoid over Monoid
```

A fraction of the Algebraic Zoo



Library fragment 1

```
NearSemiring := combine AdditiveSemigroup , Semigroup , RightRingoid over R
NearSemifield := combine NearSemiring , Group over Semigroup
Semifield := combine NearSemifield , LeftRingoid over RingoidSig
NearRing := combine AdditiveGroup , Semigroup , RightRingoid over RingoidSi
Rng := combine AbelianAdditiveGroup , Semigroup , Ringoid over RingoidSig
Semiring := combine AdditiveCommutativeMonoid , Monoid1 , Ringoid , Left0 ov
SemiRng := combine AdditiveCommutativeMonoid , Semigroup , Ringoid over Rir
Dioid := combine Semiring , IdempotentAdditiveMagma over AdditiveMagma
Ring := combine Rng, Semiring over SemiRng
CommutativeRing := combine Ring , CommutativeMagma over Magma
BooleanRing := combine CommutativeRing , IdempotentMagma over Magma
NoZeroDivisors := Ringoid0Sig extended by {
  axiom onlyZeroDivisor_*_0: forall x:U.
    ((exists b:U. x*b = 0) and (exists b:U. b*x = 0)) implies (x = 0) }
Domain := combine Ring , NoZeroDivisors over Ringoid0Sig
IntegralDomain := combine CommutativeRing , NoZeroDivisors over Ringoid0Si
DivisionRing := Ring extended by {
  axiom divisible : forall x:U. not (x=0) implies
    ((exists! y:U. y*x = 1) and (exists! y:U. x*y = 1)) }
Field := combine DivisionRing , IntegralDomain over Ring
```

Library fragment 2

```
MoufangLoop := combine Loop, MoufangIdentity over Magma
LeftShelfSig := Magma[ * |-> |> ]
LeftShelf := LeftDistributiveMagma [ * |-> |> ]
RightShelfSig := Magma[ * |-> <| ]
RightShelf := RightDistributiveMagma[ * |-> <| ]
RackSig := combine LeftShelfSig, RightShelfSig over Carrier
Shelf := combine LeftShelf, RightShelf over RackSig
LeftBinaryInverse := RackSig extended by {
    axiom leftInverse_|>_<| : forall x,y:U. (x |> y) <| x = y }
RightBinaryInverse := RackSig extended by {
    axiom rightInverse_|>_<| : forall x,y:U. x |> (y <| x) = y }
Rack := combine RightShelf, LeftShelf, LeftBinaryInverse,
    RightBinaryInverse over RackSig
LeftIdempotence := IdempotentMagma[ * |-> |> ]
RightIdempotence := IdempotentMagma[ * |-> <| ]
LeftSpindle := combine LeftShelf, LeftIdempotence over LeftShelfSig
RightSpindle := combine RightShelf, RightIdempotence over RightShelfSig
Quandle := combine Rack, LeftSpindle, RightSpindle over Shelf
```

What we have

- A decent library of theories
- An expander
- Mostly complete export (of expanded version) to MMT/OpenMath
- Mostly complete export (of expanded version) to Matita
- In-progress: "export" to metaocaml and Template Haskell

But what does it mean?

- Intuitively: work in some category of signatures
  - extend: embedding
  - renaming: renaming!
  - combine: pushout
- Lots of precedent (Goguen and Burstall, D. Smith, and many many followers)

But what does it mean?

- Intuitively: work in some category of signatures
  - extend: embedding
  - renaming: renaming!
  - combine: pushout
- Lots of precedent (Goguen and Burstall, D. Smith, and many many followers)
- We don't think it works well enough!

```
T1 := Theory { n : Integer }
T2 := Theory { n : Natural }
T3 := combine T1, T2 over Empty
```

result(s):

But what does it mean?

- Intuitively: work in some category of signatures
  - extend: embedding
  - renaming: renaming!
  - combine: pushout
- Lots of precedent (Goguen and Burstall, D. Smith, and many many followers)
- We don't think it works well enough!

```
T1 := Theory { n : Integer }
T2 := Theory { n : Natural }
T3 := combine T1, T2 over Empty
```

result(s):

```
T3 := Theory {
  n$234 : Integer
  n$235 : Natural
}
```

But what does it mean?

- Intuitively: work in some category of signatures
  - extend: embedding
  - renaming: renaming!
  - combine: pushout
- Lots of precedent (Goguen and Burstall, D. Smith, and many many followers)
- We don't think it works well enough!

```
T1 := Theory { n : Integer }
T2 := Theory { n : Natural }
T3 := combine T1, T2 over Empty
```

result(s):

```
T3 := Theory {
  T1/n : Integer
  T2/n : Natural
}
```

But what does it mean?

- Intuitively: work in some category of signatures
  - extend: embedding
  - renaming: renaming!
  - combine: pushout
- Lots of precedent (Goguen and Burstall, D. Smith, and many many followers)
- We don't think it works well enough!

```
T1 := Theory { n : Integer }
T2 := Theory { n : Natural }
T3 := combine T1, T2 over Empty
```

result(s):

The problem:

1. theory does not distinguish between isomorphic presentations
2. humans distinguish them, to a point

The Semantics of Syntax qua Syntax

We need a semantics of our language(s) as syntax.

Requirements:

- Names matter in the presentation
- Arrows matter (categorical thinking)
- Independent of the underlying logic and type theory
- Coherent with semantics (aka models)
  - Induces transport of conservative extensions
  - Induces transport of theorems

Focus on: the intensional content of Theory Presentations

Crucial Observation

Observation

$\texttt{ThyPres} \simeq \texttt{Context}^{op}$

Theory Presentation + translations

```
Theory {
  U : type;
  * : (U, U) -> U;
  axiom associative_* : forall x,y,z:U. (x*y)*z=x*(y*z)
}
```

λ-calculus (or logical) context + substitutions

$$U : \texttt{type}, * : (U, U) \to U, \texttt{assoc} : \forall x, y, z : U.(x*y)*z = x*(y*z)$$

Basic definitions

Definition

A context Γ is a sequence of pairs of labels and types (or kinds or propositions), $\Gamma := \langle x_0 : \sigma_0; \ldots ; x_{n-1} : \sigma_{n-1} \rangle$, such that for $i < n$

$$\langle x_0 : \sigma_0; \ldots ; x_{i-1} : \sigma_{i-1} \rangle \vdash \sigma_i : \text{Type}$$

holds (resp. : Kind, or : Prop)

Notation: $\Gamma = \langle x : \sigma \rangle_0^{n-1}$ and $\Delta = \langle y : \tau \rangle_0^{m-1}$.

Definition

$\mathbb{C}$ has as objects contexts Γ, and morphisms $\Gamma \to \Delta$ are assignments $[y_0 \mapsto t_0, \ldots , y_m \mapsto t_{m-1}]$, abbreviated as $[y \mapsto t]_0^{m-1}$ where the $t_0, \ldots , t_{m-1}$ are terms such that

$$\Gamma \vdash t_0 : \tau_0 \qquad \ldots \qquad \Gamma \vdash t_{m-1} : \tau_{m-1} \, [y \mapsto t]_0^{m-2}$$

all hold, where $\tau \, [y \mapsto t]_0^i$ denotes substitution application.

More definitions

Definition

The category of nominal assignments, $\mathbb{B}$, has the same objects as $\mathbb{C}$, but only those morphisms whose terms are labels.

Definition

Those nominal assignments where every label occurs at most once will be called general extensions ($\approx$ no confusion).

$$\begin{array}{ccc} \Gamma^+ & \xrightarrow{f^+} & \Delta^+ \\ {\scriptstyle A}\downarrow & & \downarrow{\scriptstyle B} \\ \Gamma & \xrightarrow[f^-]{} & \Delta \end{array}$$

Definition

The category of general extensions $\mathbb{E}$ has all general extensions from $\mathbb{B}$ as objects, and given two general extensions $A : \Gamma^+ \to \Gamma$ and $B : \Delta^+ \to \Delta$, a morphism $f : A \to B$ is a commutative square from $\mathbb{B}$. We will denote this commutative square by $\langle f^+, f^- \rangle : A \to B$.

Structure Theorem

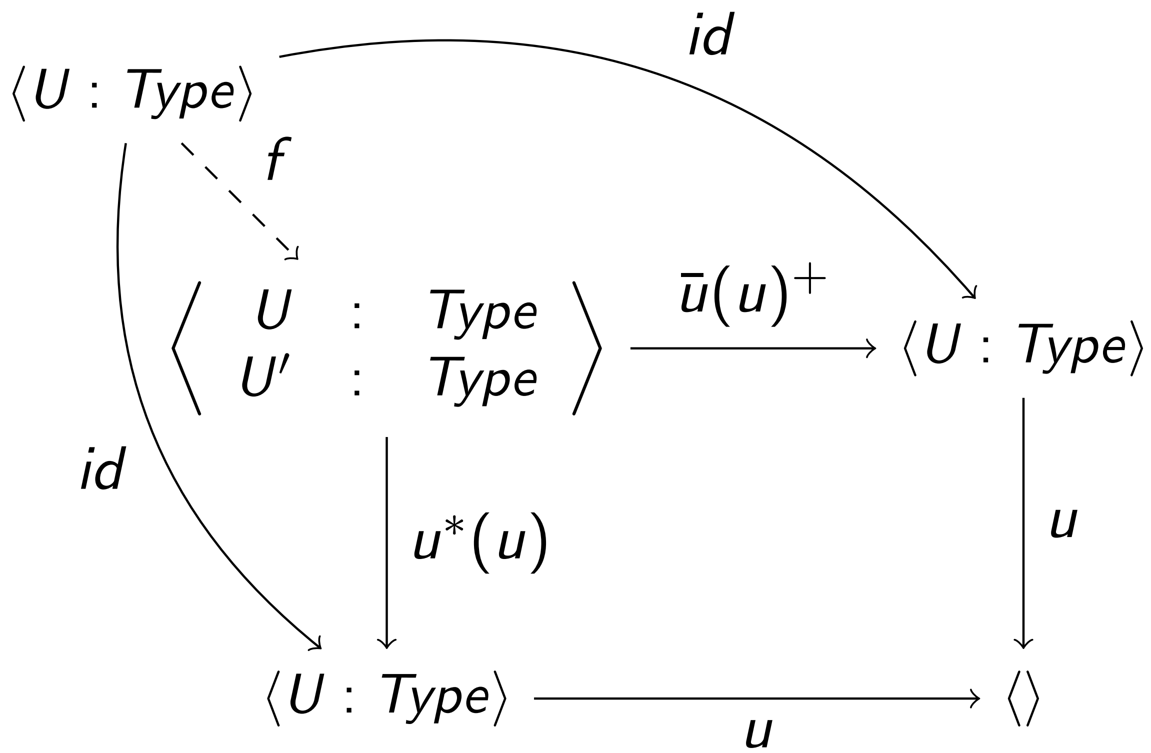
Theorem

The functor cod : $\mathbb{E} \to \mathbb{B}$ *is a fibration.*

Structure Theorem

Theorem

The functor cod : $\mathbb{E} \to \mathbb{B}$ *is a fibration.*



$$f \quad : \quad \langle U : \text{Type}\rangle \to \left\langle U : \text{Type}; U' : \text{Type}\right\rangle$$
$$f \quad := \quad \left[U \mapsto U, U' \mapsto U\right]$$

The MathScheme Theory Presentation Language

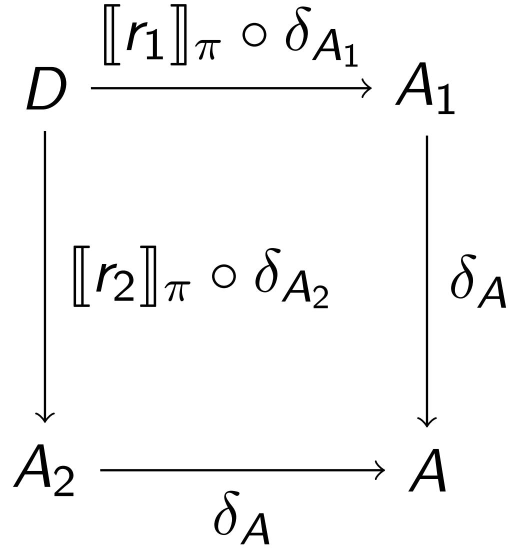
$$a, b, c \in \text{labels}$$
$$A, B, C \in \text{names}$$
$$l \in \text{judgments}^*$$
$$r \in (a_i \mapsto b_i)^*$$

$$\begin{aligned} \text{tpc} ::= \; & \text{extend } A \text{ by } \{l\} \\ | \; & \text{combine } A \; r_1, \; B \; r_2 \\ | \; & A \; ; \; B \\ | \; & A \; r \\ | \; & \text{Empty} \\ | \; & \text{Theory } \{l\} \end{aligned}$$

Note: completely generic over the underlying type theory.

Object-level semantics

$$
\begin{aligned}
[\![-]\!]_{\mathbb{B}} &: \mathsf{tpc} \rightharpoonup |\mathbb{B}| \\
[\![\mathsf{Empty}]\!]_{\mathbb{B}} &= \langle\,\rangle \\
[\![\mathsf{Theory}\ \{l\}]\!]_{\mathbb{B}} &\cong \langle l \rangle \\
[\![A\ r]\!]_{\mathbb{B}} &= [\![r]\!]_{\pi} \cdot [\![A]\!]_{\mathbb{B}} \\
[\![A; B]\!]_{\mathbb{B}} &= [\![B]\!]_{\mathbb{B}} \\
[\![\mathsf{extend}\ A\ \mathsf{by}\ \{l\}]\!]_{\mathbb{B}} &\cong [\![A]\!]_{\mathbb{B}} \mathbin{⨾} \langle l \rangle \\
[\![\mathsf{combine}\ A_1 r_1, A_2 r_2]\!]_{\mathbb{B}} &\cong D
\end{aligned}
$$



where $A = [\![A_1]\!]_{\mathbb{B}} \sqcap [\![A_2]\!]_{\mathbb{B}}$.

Object-level semantics

$$
\begin{aligned}
[\![-]\!]_{\mathbb{B}} &: \mathsf{tpc} \rightharpoonup |\mathbb{B}| \\
[\![\mathsf{Empty}]\!]_{\mathbb{B}} &= \langle\,\rangle \\
[\![\mathsf{Theory}\ \{l\}]\!]_{\mathbb{B}} &\cong \langle l \rangle \\
[\![A\ r]\!]_{\mathbb{B}} &= [\![r]\!]_{\pi} \cdot [\![A]\!]_{\mathbb{B}} \\
[\![A; B]\!]_{\mathbb{B}} &= [\![B]\!]_{\mathbb{B}} \\
[\![\mathsf{extend}\ A\ \mathsf{by}\ \{l\}]\!]_{\mathbb{B}} &\cong [\![A]\!]_{\mathbb{B}} \,⨾\, \langle l \rangle \\
[\![\mathsf{combine}\ A_1 r_1, A_2 r_2]\!]_{\mathbb{B}} &\cong D
\end{aligned}
$$

$$
\begin{array}{ccc}
D & \xrightarrow{[\![r_1]\!]_{\pi} \circ \delta_{A_1}} & A_1 \\
\downarrow{\scriptstyle [\![r_2]\!]_{\pi} \circ \delta_{A_2}} & & \downarrow{\scriptstyle \delta_A} \\
A_2 & \xrightarrow[\delta_A]{} & A
\end{array}
$$

where $A = [\![A_1]\!]_{\mathbb{B}} \sqcap [\![A_2]\!]_{\mathbb{B}}$.

```
T1 := Theory { n : Integer }
T2 := Theory { n : Natural }
T3 := combine T1 [n |-> m], T2
```

Object-level semantics

$$
\begin{aligned}
[\![-]\!]_{\mathbb{B}} &: \mathsf{tpc} \rightharpoonup |\mathbb{B}| \\
[\![\mathsf{Empty}]\!]_{\mathbb{B}} &= \langle\,\rangle \\
[\![\mathsf{Theory}\ \{l\}]\!]_{\mathbb{B}} &\cong \langle l \rangle \\
[\![A\ r]\!]_{\mathbb{B}} &= [\![r]\!]_{\pi} \cdot [\![A]\!]_{\mathbb{B}} \\
[\![A; B]\!]_{\mathbb{B}} &= [\![B]\!]_{\mathbb{B}} \\
[\![\mathsf{extend}\ A\ \mathsf{by}\ \{l\}]\!]_{\mathbb{B}} &\cong [\![A]\!]_{\mathbb{B}} ⨟ \langle l \rangle \\
[\![\mathsf{combine}\ A_1 r_1, A_2 r_2]\!]_{\mathbb{B}} &\cong D
\end{aligned}
$$

$$
\begin{array}{ccc}
D & \xrightarrow{[\![r_1]\!]_{\pi} \circ \delta_{A_1}} & A_1 \\
\Big\downarrow {\scriptstyle [\![r_2]\!]_{\pi} \circ \delta_{A_2}} & & \Big\downarrow {\scriptstyle \delta_A} \\
A_2 & \xrightarrow[\delta_A]{} & A
\end{array}
$$

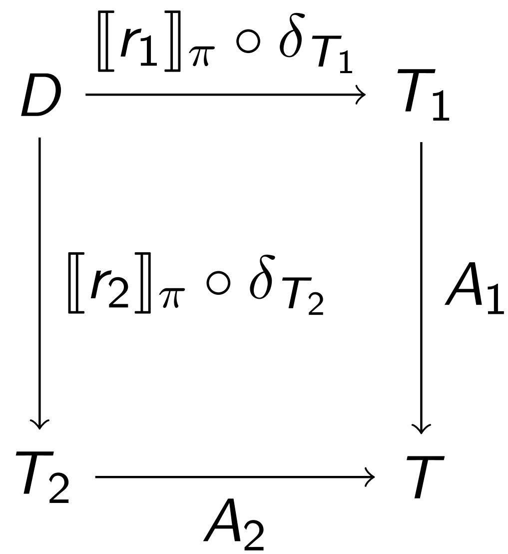
where $A = [\![A_1]\!]_{\mathbb{B}} \sqcap [\![A_2]\!]_{\mathbb{B}}$.

```
T1 := Theory { n : Integer }
T2 := Theory { n : Natural }
T3 := combine T1 [n |-> m], T2

T3 := Theory { m : Integer, n : Natural }
```

Morphism-level semantics

$$
\begin{aligned}
[\![-]\!]_{\mathbb{E}} &: \ \mathsf{tpc} \rightharpoonup |\mathbb{E}| \\
[\![\mathsf{Empty}]\!]_{\mathbb{E}} &= \ \mathsf{id}_{\langle\,\rangle} \\
[\![\mathsf{Theory}\ \{l\}]\!]_{\mathbb{E}} &\cong \ !_{\langle l \rangle} \\
[\![A\ r]\!]_{\mathbb{E}} &= \ [\![r]\!]_{\pi} \cdot [\![A]\!]_{\mathbb{E}} \\
[\![A; B]\!]_{\mathbb{E}} &= \ [\![A]\!]_{\mathbb{E}} \circ [\![B]\!]_{\mathbb{E}} \\
[\![\mathsf{extend}\ A\ \mathsf{by}\ \{l\}]\!]_{\mathbb{E}} &\cong \ \delta_A \\
[\![\mathsf{combine}\ A_1 r_1, A_2 r_2]\!]_{\mathbb{E}} &\cong \ [\![r_1]\!]_{\pi} \circ \delta_{T_1} \circ [\![A_1]\!]_{\mathbb{E}} \\
&\cong \ [\![r_2]\!]_{\pi} \circ \delta_{T_2} \circ [\![A_2]\!]_{\mathbb{E}}
\end{aligned}
$$



Future Work and Conclusion

Future Work

- Functorial semantics – diagram-level “constructions”
- Definitions [done]
- Port library to new semantics [ongoing]

Future Work and Conclusion

Future Work

- Functorial semantics – diagram-level “constructions”
- Definitions [done]
- Port library to new semantics [ongoing]

Conclusion

- There is a lot of structure in Mathematics, and it can be leveraged to simplify builder’s lives.
- Category theory can really help you
- Follow the math, don’t follow what you think the math should be