Catoids, Categories, Groupoids

Georg Struth

April 18, 2024

Abstract

This AFP entry formalises catoids, which are generalisations of single-set categories, and groupoids. More specifically, in catoids, the partial composition of arrows in a category is generalised to a multi-operation, which sends pairs of elements to sets of elements, and the definedness condition of arrow composition – two arrows can be composed if and only the target of the first matches the source of the second – is relaxed. Beyond a library of basic laws for catoids, single-set categories and groupoids, I formalise the facts that every catoid can be lifted to a modal powerset quantale, that every groupoid can be lifted to a Dedekind quantale and to power set relation algebras, a special case of a famous result of Jónsson and Tarski. Finally, I show that single-set categories are equivalent to a standard axiomatisation of categories based on a set of objects and a set of arrows, and compare catoids with related structures such as multimonoid and relational monoids (monoids in the monoidal category Rel).

Contents

1 Introductory Remarks

These Isabelle theories formalise results on catoids from [4, 2, 6] and groupoids from [3]. Catoids generalise single-set categories, as they can be found in Chapter XII of Mac Lane's book [8]. One particular result, namely that catoids can be lifted to (power set) relation algebras, is due to Jónsson and Tarski [7].

A wide-ranging formalisation of category theory based on single-set categories formalised as locales can already be found in the AFP [9]. The present type-class-based alternative might lend itself to a similar programme.

The multioperation $X \times X \to \mathcal{P}X$ in the definition of catoids is obviously isomorphic to a ternary relation $X \to X \to X \to 2$. Simple mathematical components for relational monoids, which are isomorphic (as categories with suitable morphisms) to catoids, can already be found in the AFP [5]. At this stage, I do not integrate the two components. They use different formalisations of quantales with Isabelle, which remain to be consolidated.

Catoids and groupoids admit many models. Those of catoids range from shuffle monoids and generalised effect algebras to base algebras of incidence and matrix algebras [6], whereas groupoids are so ubiquitous in mathematics that some mathematicians have argued for interchanging their names with groups, see [1] for a brief history, which goes decades beyond that of category theory.

The mathematical components in this AFP entry are also stepping stones towards the formalisation of (ω, p)-catoids, strict (ω, p)-categories and (ω, p)-quantales. Components for these structures will feature in a separate AFP entry. They contribute to a larger programme on the formalisation of higher rewriting techniques with proof assistants.

I am grateful for a fellowship at the Collegium de Lyon, Institute for Advanced Study, where this formalisation work has been done.

2 Catoids

theory *Catoid*
  imports *Main*

begin

2.1 Multimagmas

Multimagmas are sets equipped with multioperations. Multioperations are isomorphic to ternary relations.

class *multimagma* =
  fixes *mcomp* :: $'a \Rightarrow 'a \Rightarrow 'a\ set$ (**infixl** $\odot$ *70*)

begin

I introduce notation for the domain of definition of the multioperation.

abbreviation $\Delta\ x\ y \equiv (x \odot y \neq \{\})$

I extend the multioperation to powersets

definition *conv* :: $'a\ set \Rightarrow 'a\ set \Rightarrow 'a\ set$ (**infixl** $\star$ *70*) **where**
  $X \star Y = (\bigcup x \in X.\ \bigcup y \in Y.\ x \odot y)$

lemma *conv-exp*: $X \star Y = \{z.\ \exists x\ y.\ z \in x \odot y \wedge x \in X \wedge y \in Y\}$
  ⟨*proof*⟩

lemma *conv-exp2*: $(z \in X \star Y) = (\exists x\ y.\ z \in x \odot y \wedge x \in X \wedge y \in Y)$
  ⟨*proof*⟩

lemma *conv-distl*: $X \star \bigcup \mathcal{Y} = (\bigcup Y \in \mathcal{Y}.\ X \star Y)$
  ⟨*proof*⟩

lemma *conv-distr*: $\bigcup \mathcal{X} \star Y = (\bigcup X \in \mathcal{X}.\ X \star Y)$
  ⟨*proof*⟩

lemma *conv-distl-small*: $X \star (Y \cup Z) = X \star Y \cup X \star Z$
  ⟨*proof*⟩

lemma *conv-distr-small*: $(X \cup Y) \star Z = X \star Z \cup Y \star Z$
  ⟨*proof*⟩

lemma *conv-isol*: $X \subseteq Y \Longrightarrow Z \star X \subseteq Z \star Y$
⟨*proof*⟩

lemma *conv-isor*: $X \subseteq Y \Longrightarrow X \star Z \subseteq Y \star Z$
⟨*proof*⟩

lemma *conv-atom* [*simp*]: $\{x\} \star \{y\} = x \odot y$
⟨*proof*⟩

end

2.2 Multisemigroups

Sultisemigroups are associative multimagmas.

class *multisemigroup* = *multimagma* +
assumes *assoc*: $(\bigcup v \in y \odot z.\ x \odot v) = (\bigcup v \in x \odot y.\ v \odot z)$

begin

lemma *assoc-exp*: $(\exists v.\ w \in x \odot v \wedge v \in y \odot z) = (\exists v.\ v \in x \odot y \wedge w \in v \odot z)$
⟨*proof*⟩

lemma *assoc-var*: $\{x\} \star (y \odot z) = (x \odot y) \star \{z\}$
⟨*proof*⟩

Associativity extends to powersets.

lemma *conv-assoc*: $X \star (Y \star Z) = (X \star Y) \star Z$
⟨*proof*⟩

end

2.3 st-Multimagmas

I equip multimagmas with source and target maps.

class *st-op* =
fixes *src* :: $'a \Rightarrow 'a$ (σ)
and *tgt* :: $'a \Rightarrow 'a$ (τ)

class *st-multimagma* = *multimagma* + *st-op* +
assumes *Dst*: $x \odot y \neq \{\} \Longrightarrow \tau\ x = \sigma\ y$
and *s-absorb* [*simp*]: $\sigma\ x \odot x = \{x\}$
and *t-absorb* [*simp*]: $x \odot \tau\ x = \{x\}$

The following sublocale proof sets up opposition/duality.

sublocale *st-multimagma* ⊆ *stopp*: *st-multimagma* $\lambda x\ y.\ y \odot x$ *tgt src*
rewrites *stopp.conv* $X\ Y = Y \star X$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *ts-compat* [*simp*]: $\tau\ (\sigma\ x) = \sigma\ x$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *ss-idem* [*simp*]: $\sigma\ (\sigma\ x) = \sigma\ x$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *st-fix*: $(\tau\ x = x) = (\sigma\ x = x)$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *st-eq1*: $\sigma\ x = x \Longrightarrow \sigma\ x = \tau\ x$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *st-eq2*: $\tau\ x = x \Longrightarrow \sigma\ x = \tau\ x$
⟨*proof*⟩

I extend source and target operations to powersets by taking images.

abbreviation (**in** *st-op*) *Src* :: *'a set* ⇒ *'a set* **where**
Src ≡ *image* σ

abbreviation (**in** *st-op*) *Tgt* :: *'a set* ⇒ *'a set* **where**
Tgt ≡ *image* τ

Fixpoints of source and target maps model source and target elements. These correspond to units.

abbreviation (**in** *st-op*) *sfix* :: *'a set* **where**
sfix ≡ $\{x.\ \sigma\ x = x\}$

abbreviation (**in** *st-op*) *tfix* :: *'a set* **where**
tfix ≡ $\{x.\ \tau\ x = x\}$

lemma (**in** *st-multimagma*) *st-mm-rfix* [*simp*]: *tfix* = *stopp.sfix*
⟨*proof*⟩

lemma (**in** *st-multimagma*) *st-fix-set*: $\{x.\ \sigma\ x = x\} = \{x.\ \tau\ x = x\}$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *stfix-set*: *sfix* = *tfix*
⟨*proof*⟩

lemma (**in** *st-multimagma*) *sfix-im*: *sfix* = *Src UNIV*
⟨*proof*⟩

lemma (**in** *st-multimagma*) *tfix-im*: *tfix* = *Tgt UNIV*
⟨*proof*⟩

lemma (**in** *st-multimagma*) *ST-im*: *Src UNIV* = *Tgt UNIV*
⟨*proof*⟩

Source and target elements are "orthogonal" idempotents.

lemma (**in** *st-multimagma*) *s-idem* [*simp*]: $\sigma\ x \odot \sigma\ x = \{\sigma\ x\}$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *s-ortho*:
$\Delta\ (\sigma\ x)\ (\sigma\ y) \Longrightarrow \sigma\ x = \sigma\ y$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *s-ortho-iff*: $\Delta\ (\sigma\ x)\ (\sigma\ y) = (\sigma\ x = \sigma\ y)$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *st-ortho-iff*: $\Delta\ (\sigma\ x)\ (\tau\ y) = (\sigma\ x = \tau\ y)$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *s-ortho-id*: $(\sigma\ x) \odot (\sigma\ y) = (\mathit{if}\ (\sigma\ x = \sigma\ y)\ \mathit{then}\ \{\sigma\ x\}\ \mathit{else}\ \{\})$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *s-absorb-var*: $(\sigma\ y \neq \sigma\ x) = (\sigma\ y \odot x = \{\})$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *s-absorb-var2*: $(\sigma\ y = \sigma\ x) = (\sigma\ y \odot x \neq \{\})$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *s-absorb-var3*: $(\sigma\ y = \sigma\ x) = \Delta\ (\sigma\ x)\ y$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *s-assoc*: $\{\sigma\ x\} \star (\sigma\ y \odot z) = (\sigma\ x \odot \sigma\ y) \star \{z\}$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *sfix-absorb-var* [*simp*]: $(\bigcup e \in \mathit{sfix}.\ e \odot x) = \{x\}$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *tfix-absorb-var*: $(\bigcup e \in \mathit{tfix}.\ x \odot e) = \{x\}$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *st-comm*: $\tau\ x \odot \sigma\ y = \sigma\ y \odot \tau\ x$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *s-weak-twisted*: $(\bigcup u \in x \odot y.\ \sigma\ u \odot x) \subseteq x \odot \sigma\ y$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *s-comm*: $\sigma\ x \odot \sigma\ y = \sigma\ y \odot \sigma\ x$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *s-export* [*simp*]: $\mathit{Src}\ (\sigma\ x \odot y) = \sigma\ x \odot \sigma\ y$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *st-prop*: $(\tau\ x = \sigma\ y) = \Delta\ (\tau\ x)\ (\sigma\ y)$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *weak-local-var*: $\tau\ x \odot \sigma\ y = \{\} \Longrightarrow x \odot y = \{\}$
⟨*proof*⟩

The following facts hold by duality.

lemma (**in** *st-multimagma*) *st-compat*: $\sigma\ (\tau\ x) = \tau\ x$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *tt-idem*: $\tau\ (\tau\ x) = \tau\ x$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *t-idem*: $\tau\ x \odot \tau\ x = \{\tau\ x\}$
⟨*proof*⟩

lemma (**in** *st-multimagma*)*t-weak-twisted*: $(\bigcup u \in y \odot x.\ x \odot \tau\ u) \subseteq \tau\ y \odot x$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *t-comm*: $\tau\ x \odot \tau\ y = \tau\ y \odot \tau\ x$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *t-export*: *image* $\tau\ (x \odot \tau\ y) = \tau\ x \odot \tau\ y$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *tt-comp-prop*: $\Delta\ (\tau\ x)\ (\tau\ y) = (\tau\ x = \tau\ y)$
⟨*proof*⟩

The set of all sources (and targets) are units at powerset level.

lemma (**in** *st-multimagma*) *conv-uns* [*simp*]: *sfix* $\star\ X = X$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *conv-unt*: $X \star$ *tfix* $= X$
⟨*proof*⟩

I prove laws of modal powerset quantales.

lemma (**in** *st-multimagma*) *Src-exp*: *Src* $X = \{\sigma\ x\ |x.\ x \in X\}$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *ST-compat* [*simp*]: *Src* (*Tgt* X) = *Tgt* X
⟨*proof*⟩

lemma (**in** *st-multimagma*) *TS-compat*: *Tgt* (*Src* X) = *Src* X
⟨*proof*⟩

lemma (**in** *st-multimagma*) *Src-absorp* [*simp*]: *Src* $X \star X = X$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *Tgt-absorp*: $X \star$ *Tgt* $X = X$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *Src-Sup-pres*: $Src\ (\bigcup \mathcal{X}) = (\bigcup X \in \mathcal{X}.\ Src\ X)$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *Tgt-Sup-pres*: $Tgt\ (\bigcup \mathcal{X}) = (\bigcup\ X \in \mathcal{X}.\ Tgt\ X)$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *ST-comm*: $Src\ X \star Tgt\ Y = Tgt\ Y \star Src\ X$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *Src-comm*: $Src\ X \star Src\ Y = Src\ Y \star Src\ X$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *Tgt-comm*: $Tgt\ X \star Tgt\ Y = Tgt\ Y \star Tgt\ X$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *Src-subid*: $Src\ X \subseteq sfix$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *Tgt-subid*: $Tgt\ X \subseteq tfix$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *Src-export* [*simp*]: $Src\ (Src\ X \star\ Y) = Src\ X \star Src\ Y$
⟨*proof*⟩

lemma (**in** *st-multimagma*) *Tgt-export* [*simp*]: $Tgt\ (X \star\ Tgt\ Y) = Tgt\ X \star\ Tgt\ Y$
⟨*proof*⟩

Locality implies st-locality, which is the composition pattern of categories.

lemma (**in** *st-multimagma*) *locality*:
assumes *src-local*: $Src\ (x \odot \sigma\ y) \subseteq Src\ (x \odot y)$
and *tgt-local*: $Tgt\ (\tau\ x \odot y) \subseteq Tgt\ (x \odot y)$
shows $\Delta\ x\ y\ = (\tau\ x = \sigma\ y)$
⟨*proof*⟩

2.4 Catoids

class *catoid* = *st-multimagma* + *multisemigroup*

sublocale *catoid* ⊆ *ts-msg*: *catoid* $\lambda x\ y.\ y \odot x$ *tgt src*
⟨*proof*⟩

lemma (**in** *catoid*) *src-comp-aux*: $v \in x \odot y \Longrightarrow \sigma\ v = \sigma\ x$
⟨*proof*⟩

lemma (**in** *catoid*) *src-comp*: $Src\ (x \odot y) \subseteq \{\sigma\ x\}$
⟨*proof*⟩

lemma (**in** *catoid*) *src-comp-cond*: $(\Delta\ x\ y) \Longrightarrow Src\ (x \odot y) = \{\sigma\ x\}$
⟨*proof*⟩

lemma (**in** *catoid*) *tgt-comp-aux*: $v \in x \odot y \Longrightarrow \tau\ v = \tau\ y$
⟨*proof*⟩

lemma (**in** *catoid*) *tgt-comp*: $Tgt\ (x \odot y) \subseteq \{\tau\ y\}$
⟨*proof*⟩

lemma (**in** *catoid*) *tgt-comp-cond*: $\Delta\ x\ y \Longrightarrow Tgt\ (x \odot y) = \{\tau\ y\}$
⟨*proof*⟩

lemma (**in** *catoid*) *src-weak-local*: $Src\ (x \odot y) \subseteq Src\ (x \odot \sigma\ y)$
⟨*proof*⟩

lemma (**in** *catoid*) *src-local-cond*:
$\Delta\ x\ y \Longrightarrow Src\ (x \odot y) = Src\ (x \odot \sigma\ y)$
⟨*proof*⟩

lemma (**in** *catoid*) *tgt-weak-local*: $Tgt\ (x \odot y) \subseteq Tgt\ (\tau\ x \odot y)$
⟨*proof*⟩

lemma (**in** *catoid*) *tgt-local-cond*:
$\Delta\ x\ y \Longrightarrow Tgt\ (x \odot y) = Tgt\ (\tau\ x \odot y)$
⟨*proof*⟩

lemma (**in** *catoid*) *src-twisted-aux*:
$u \in x \odot y \Longrightarrow (x \odot \sigma\ y = \sigma\ u \odot x)$
⟨*proof*⟩

lemma (**in** *catoid*) *src-twisted-cond*:
$\Delta\ x\ y \Longrightarrow x \odot \sigma\ y = \bigcup\{\sigma\ u \odot x\ |u.\ u \in x \odot y\}$
⟨*proof*⟩

lemma (**in** *catoid*) *tgt-twisted-aux*:
$u \in x \odot y \Longrightarrow (\tau\ x \odot y = y \odot \tau\ u)$
⟨*proof*⟩

lemma (**in** *catoid*) *tgt-twisted-cond*:
$\Delta\ x\ y \Longrightarrow \tau\ x \odot y = \bigcup\{y \odot \tau\ u\ |u.\ u \in x \odot y\}$
⟨*proof*⟩

lemma (**in** *catoid*) *src-funct*:
$x \in y \odot z \Longrightarrow x' \in y \odot z \Longrightarrow \sigma\ x = \sigma\ x'$
⟨*proof*⟩

lemma (**in** *catoid*) *st-local-iff*:
$(\forall x\ y.\ \Delta\ x\ y = (\tau\ x = \sigma\ y)) = (\forall v\ x\ y\ z.\ v \in x \odot y \longrightarrow \Delta\ y\ z \longrightarrow \Delta\ v\ z)$
⟨*proof*⟩

Again one can lift to properties of modal semirings and quantales.

lemma (**in** *catoid*) *Src-weak-local*: $Src\ (X \star Y) \subseteq Src\ (X \star Src\ Y)$
⟨*proof*⟩

lemma (**in** *catoid*) *Tgt-weak-local*: $Tgt\ (X \star Y) \subseteq Tgt\ (Tgt\ X \star Y)$
⟨*proof*⟩

st-Locality implies locality.

lemma (**in** *catoid*) *st-locality-l-locality*:
assumes $\Delta\ x\ y = (\tau\ x = \sigma\ y)$
shows $Src\ (x \odot y) = Src\ (x \odot \sigma\ y)$
⟨*proof*⟩

lemma (**in** *catoid*) *st-locality-r-locality*:
assumes *lr-locality*: $\Delta\ x\ y = (\tau\ x = \sigma\ y)$
shows $Tgt\ (x \odot y) = Tgt\ (\tau\ x \odot y)$
⟨*proof*⟩

lemma (**in** *catoid*) *st-locality-locality*:
$(Src\ (x \odot y) = Src\ (x \odot \sigma\ y) \wedge Tgt\ (x \odot y) = Tgt\ (\tau\ x \odot y)) = (\Delta\ x\ y = (\tau\ x = \sigma\ y))$
⟨*proof*⟩

2.5 Locality

For st-multimagmas there are different notions of locality. I do not develop this in detail.

class *local-catoid* = *catoid* +
assumes *src-local*: $Src\ (x \odot \sigma\ y) \subseteq Src\ (x \odot y)$
and *tgt-local*: $Tgt\ (\tau\ x \odot y) \subseteq Tgt\ (x \odot y)$

sublocale *local-catoid* ⊆ *sts-msg*: *local-catoid* $\lambda x\ y.\ y \odot x$ *tgt src*
⟨*proof*⟩

lemma (**in** *local-catoid*) *src-local-eq* [*simp*]: $Src\ (x \odot \sigma\ y) = Src\ (x \odot y)$
⟨*proof*⟩

lemma (**in** *local-catoid*) *tgt-local-eq*: $Tgt\ (\tau\ x \odot y) = Tgt\ (x \odot y)$
⟨*proof*⟩

lemma (**in** *local-catoid*) *src-twisted*: $x \odot \sigma\ y = (\bigcup u \in x \odot y.\ \sigma\ u \odot x)$
⟨*proof*⟩

lemma (**in** *local-catoid*) *tgt-twisted*: $\tau\ x \odot y = (\bigcup u \in x \odot y.\ y \odot \tau\ u)$
⟨*proof*⟩

lemma (**in** *local-catoid*) *local-var*: $\Delta\ x\ y \Longrightarrow \Delta\ (\tau\ x)\ (\sigma\ y)$
⟨*proof*⟩

lemma (**in** *local-catoid*) *local-var-eq* [*simp*]: $\Delta\ (\tau\ x)\ (\sigma\ y) = \Delta\ x\ y$
⟨*proof*⟩

I lift locality to powersets.

lemma (**in** *local-catoid*) *Src-local* [*simp*]: $Src\ (X \star Src\ Y) = Src\ (X \star Y)$
⟨*proof*⟩

lemma (**in** *local-catoid*) *Tgt-local* [*simp*]: $Tgt\ (Tgt\ X \star Y) = Tgt\ (X \star Y)$
⟨*proof*⟩

lemma (**in** *local-catoid*) *st-local*: $\Delta\ x\ y = (\tau\ x = \sigma\ y)$
⟨*proof*⟩

2.6 From partial magmas to single-set categories.

class *functional-magma* = *multimagma* +
assumes *functionality*: $x \in y \odot z \Longrightarrow x' \in y \odot z \Longrightarrow x = x'$

begin

Functional magmas could also be called partial magmas. The multioperation corresponds to a partial operation.

lemma *partial-card*: $card\ (x \odot y) \leq 1$
⟨*proof*⟩

lemma *fun-in-sgl*: $(x \in y \odot z) = (\{x\} = y \odot z)$
⟨*proof*⟩

I introduce a partial operation.

definition *pcomp* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\otimes$ *70*) **where**
$x \otimes y = (THE\ z.\ z \in x \odot y)$

lemma *functionality-var*: $\Delta\ x\ y \Longrightarrow (\exists !z.\ z \in x \odot y)$
⟨*proof*⟩

lemma *functionality-lem*: $(\exists !z.\ z \in x \odot y) \vee (x \odot y = \{\})$
⟨*proof*⟩

lemma *functionality-lem-var*: $\Delta\ x\ y = (\exists z.\ \{z\} = x \odot y)$
⟨*proof*⟩

lemma *pcomp-def-var*: $(\Delta\ x\ y \wedge x \otimes y = z) = (z \in x \odot y)$
⟨*proof*⟩

lemma *pcomp-def-var2*: $\Delta\ x\ y \Longrightarrow ((x \otimes y = z) = (z \in x \odot y))$
⟨*proof*⟩

lemma *pcomp-def-var3*: $\Delta\ x\ y \Longrightarrow ((x \otimes y = z) = (\{z\} = x \odot y))$
⟨*proof*⟩

end

class *functional-st-magma* = *functional-magma* + *st-multimagma*

class *functional-semigroup* = *functional-magma* + *multisemigroup*

begin

lemma *pcomp-assoc-defined*: $(\Delta\ u\ v \wedge \Delta\ (u \otimes v)\ w) = (\Delta\ u\ (v \otimes w) \wedge \Delta\ v\ w)$
⟨*proof*⟩

lemma *pcomp-assoc*: $\Delta\ x\ y \wedge \Delta\ (x \otimes y)\ z \Longrightarrow (x \otimes y) \otimes z = x \otimes (y \otimes z)$
⟨*proof*⟩

end

class *functional-catoid* = *functional-semigroup* + *catoid*

Finally, here comes the definition of single-set categories as in Chapter 12 of Mac Lane's book, but with partiality of arrow composition modelled using a multioperation, or a partial operation based on it.

class *single-set-category* = *functional-catoid* + *local-catoid*

begin

lemma *st-assoc*: $\tau\ x = \sigma\ y \Longrightarrow \tau\ y = \sigma\ z \Longrightarrow (x \otimes y) \otimes z = x \otimes (y \otimes z)$
⟨*proof*⟩

end

2.7 Morphisms of multimagmas and lr-multimagmas

In the context of single-set categories, these morphisms are functors. Bounded morphisms are functional bisimulations. They are known as zig-zag morphisms or p-morphism in modal and substructural logics.

definition *mm-morphism* :: $('a{::}multimagma \Rightarrow 'b{::}multimagma) \Rightarrow bool$ **where**
mm-morphism $f = (\forall x\ y.\ image\ f\ (x \odot y) \subseteq f\ x \odot f\ y)$

definition *bounded-mm-morphism* :: $('a{::}multimagma \Rightarrow 'b{::}multimagma) \Rightarrow bool$ **where**
bounded-mm-morphism $f = (mm\text{-}morphism\ f \wedge (\forall x\ u\ v.\ f\ x \in u \odot v \longrightarrow (\exists y\ z.\ u = f\ y \wedge v = f\ z \wedge x \in y \odot z)))$

definition *st-mm-morphism* :: $('a{::}st\text{-}multimagma \Rightarrow 'b{::}st\text{-}multimagma) \Rightarrow bool$ **where**
st-mm-morphism $f = (mm\text{-}morphism\ f \wedge f \circ \sigma = \sigma \circ f \wedge f \circ \tau = \tau \circ f)$

definition *bounded-st-mm-morphism* :: (*'a*::*st-multimagma* ⇒ *'b*::*st-multimagma*) ⇒ *bool* **where**
bounded-st-mm-morphism f = (*bounded-mm-morphism f* ∧ *st-mm-morphism f*)

2.8 Relationship with categories

Next I add a standard definition of a category following Moerdijk and Mac Lane's book and, for good measure, show that categories form single set categories and vice versa.

locale *category* =
fixes *id* :: *'objects* ⇒ *'arrows*
and *dom* :: *'arrows* ⇒ *'objects*
and *cod* :: *'arrows* ⇒ *'objects*
and *comp* :: *'arrows* ⇒ *'arrows* ⇒ *'arrows* (**infixl** · *70*)
assumes *dom-id* [*simp*]: *dom* (*id X*) = *X*
and *cod-id* [*simp*]: *cod* (*id X*) = *X*
and *id-dom* [*simp*]: *id* (*dom f*) · *f* = *f*
and *id-cod* [*simp*]: *f* · *id* (*cod f*) = *f*
and *dom-loc* [*simp*]: *cod f* = *dom g* ⟹ *dom* (*f* · *g*) = *dom f*
and *cod-loc* [*simp*]: *cod f* = *dom g* ⟹ *cod* (*f* · *g*) = *cod g*
and *assoc*: *cod f* = *dom g* ⟹ *cod g* = *dom h* ⟹ (*f* · *g*) · *h* = *f* · (*g* · *h*)

begin

lemma *cod f* = *dom g* ⟹ *dom* (*f* · *g*) = *dom* (*f* · *id* (*dom g*))
⟨*proof*⟩

abbreviation *LL f* ≡ *id* (*dom f*)

abbreviation *RR f* ≡ *id* (*cod f*)

abbreviation *Comp* ≡ λ*f g*. (*if RR f* = *LL g then* {*f* · *g*} *else* {})

end

typedef (**overloaded**) *'a*::*single-set-category st-objects* = {*x*::*'a*::*single-set-category*. σ *x* = *x*}
⟨*proof*⟩

setup-lifting *type-definition-st-objects*

lemma *Sfix-coerce* [*simp*]: *Abs-st-objects* (σ (*Rep-st-objects X*)) = *X*
⟨*proof*⟩

lemma *Rfix-coerce* [*simp*]: *Abs-st-objects* (τ (*Rep-st-objects X*)) = *X*
⟨*proof*⟩

sublocale *single-set-category* ⊆ *sscatcat*: *category Rep-st-objects Abs-st-objects* ∘ σ *Abs-st-objects* ∘ τ (⊗)

⟨*proof*⟩

sublocale *category* ⊆ *catlrm*: *st-multimagma Comp LL RR*
⟨*proof*⟩

sublocale *category* ⊆ *catsscat*: *single-set-category Comp LL RR*
⟨*proof*⟩

2.9 A Mac Lane style variant

Next I present an axiomatisation of single-set categories that follows Mac Lane's axioms in Chapter I of his textbook more closely, but still uses a multioperation for arrow composition.

class *mlss-cat* = *functional-magma* +
fixes *l0* :: $'a \Rightarrow 'a$
fixes *r0* :: $'a \Rightarrow 'a$
assumes *comp0-def*: $(x \odot y \neq \{\}) = (r0\ x = l0\ y)$
assumes *r0l0* [*simp*]: $r0\ (l0\ x) = l0\ x$
assumes *l0r0* [*simp*]: $l0\ (r0\ x) = r0\ x$
assumes *l0-absorb* [*simp*]: $l0\ x \otimes x = x$
assumes *r0-absorb* [*simp*] : $x \otimes r0\ x = x$
assumes *assoc-defined*: $(u \odot v \neq \{\} \wedge (u \otimes v) \odot w \neq \{\}) = (u \odot (v \otimes w) \neq \{\} \wedge v \odot w \neq \{\})$
assumes *comp0-assoc*: $r0\ x = l0\ y \Longrightarrow r0\ y = l0\ z \Longrightarrow x \otimes (y \otimes z) = (x \otimes y) \otimes z$
assumes *locall-var*: $r0\ x = l0\ y \Longrightarrow l0\ (x \otimes y) = l0\ x$
assumes *localr-var*: $r0\ x = l0\ y \Longrightarrow r0\ (x \otimes y) = r0\ y$

begin

lemma *ml-locall* [*simp*]: $l0\ (x \otimes l0\ y) = l0\ (x \otimes y)$
⟨*proof*⟩

lemma *ml-localr* [*simp*]: $r0\ (r0\ x \otimes y) = r0\ (x \otimes y)$
⟨*proof*⟩

lemma *ml-locall-im* [*simp*]: $image\ l0\ (x \odot l0\ y) = image\ l0\ (x \odot y)$
⟨*proof*⟩

lemma *ml-localr-im* [*simp*]: $image\ r0\ (r0\ x \odot y) = image\ r0\ (x \odot y)$
⟨*proof*⟩

end

sublocale *single-set-category* ⊆ *sscatml*: *mlss-cat* $(\odot)\ \sigma\ \tau$
⟨*proof*⟩

sublocale *mlss-cat* ⊆ *mlsscat*: *single-set-category* $(\odot)$ *l0 r0*
⟨*proof*⟩

2.10 Product of catoids

Finally I formalise products of categories as an exercise.

instantiation *prod* :: (*catoid*, *catoid*) *catoid*
begin

definition *src-prod* $x = (\sigma\ (fst\ x), \sigma\ (snd\ x))$
  for $x :: 'a \times 'b$

definition *tgt-prod* $x = (\tau\ (fst\ x), \tau\ (snd\ x))$
  for $x :: 'a \times 'b$

definition *mcomp-prod* $x\ y = \{(u,v)\ |u\ v.\ u \in fst\ x \odot fst\ y \wedge v \in snd\ x \odot snd\ y\}$
  for $x\ y :: 'a \times 'b$

instance
⟨*proof*⟩

end

instantiation *prod* :: (*single-set-category*, *single-set-category*) *single-set-category*
begin

instance
⟨*proof*⟩

end

end

3 Groupoids

theory *Groupoid*
  imports *Catoid*

begin

3.1 st-Multigroupoids

I define multigroupoids, extending the standard definition. I equip catoids with an operation of inversion.

class *inv-op* = **fixes** *inv* :: $'a \Rightarrow 'a$

class *st-multigroupoid* = *catoid* + *inv-op* +
  assumes *invl*: $\sigma\ x \in x \odot inv\ x$
  and *invr*: $\tau\ x \in inv\ x \odot x$

sublocale *st-multigroupoid* ⊆ *st-mgpd*: *st-multigroupoid* $\lambda x\ y.\ y \odot x$ *tgt src inv*
⟨*proof*⟩

Every multigroupoid is local.

lemma (**in** *st-multigroupoid*) *st-mgpd-local*:
assumes $\tau\ x = \sigma\ y$
shows $\Delta\ x\ y$
⟨*proof*⟩

sublocale *st-multigroupoid* ⊆ *stmgpd*: *local-catoid* (⊙) *src tgt*
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *tgt-inv* [*simp*]: τ (*inv* x) $= \sigma\ x$
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *src-inv*: σ (*inv* x) $= \tau\ x$
⟨*proof*⟩

The following lemma is from Theorem 5.2 of Jónsson and Tarski's Boolean Algebras with Operators II article.

lemma (**in** *st-multigroupoid*) *bao3*:
assumes $x \odot y = \{\sigma\ x\}$
shows *inv* $x = y$
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *inv-s* [*simp*]: *inv* $(\sigma\ x) = \sigma\ x$
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *srcfunct-inv*:
$\sigma\ x \in x \odot$ *inv* $x \Longrightarrow \sigma\ y \in x \odot$ *inv* $x \Longrightarrow \sigma\ x = \sigma\ y$
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *tgtfunct-inv*:
$\tau\ x \in$ *inv* $x \odot x \Longrightarrow \tau\ y \in$ *inv* $x \odot x \Longrightarrow \tau\ x = \tau\ y$
⟨*proof*⟩

As for catoids, I prove quantalic properties, lifting to powersets.

abbreviation (**in** *st-multigroupoid*) *Inv* :: *'a set* ⇒ *'a set* **where**
Inv ≡ *image inv*

lemma (**in** *st-multigroupoid*) *Inv-exp*: *Inv* $X = \{$*inv* $x \mid x.\ x \in X\}$
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *Inv-un*: *Inv* $(X \cup Y) =$ *Inv* $X \cup$ *Inv* Y
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *Inv-Un*: *Inv* $(\bigcup \mathcal{X}) = (\bigcup X \in \mathcal{X}.$ *Inv* $X)$
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *Invl*: *Src X* ⊆ *X* ⋆ *Inv X*
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *Invr*: *Tgt X* ⊆ *Inv X* ⋆ *X*
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *Inv-strong-gelfand*: *X* ⊆ *X* ⋆ *Inv X* ⋆ *X*
⟨*proof*⟩

At powerset level, one can define domain and codomain operations explicitly as in relation algebras.

lemma (**in** *st-multigroupoid*) *dom-def*: *Src X* = *sfix* ∩ (*X* ⋆ *Inv X*)
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *cod-def*: *Tgt X* = *sfix* ∩ (*Inv X* ⋆ *X*)
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *dom-def-var*: *Src X* = *sfix* ∩ (*X* ⋆ *UNIV*)
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *cod-def-var*: *Tgt X* = *sfix* ∩ (*UNIV* ⋆ *X*)
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *dom-univ*: *X* ⋆ *UNIV* = *Src X* ⋆ *UNIV*
⟨*proof*⟩

lemma (**in** *st-multigroupoid*) *cod-univ*: *UNIV* ⋆ *X* = *UNIV* ⋆ *Tgt X*
⟨*proof*⟩

3.2 Groupoids

Groupoids are simply functional multigroupoids. I start with a somewhat indirect axiomatisaion.

class *groupoid-var* = *st-multigroupoid* + *functional-catoid*

begin

lemma *invl* [*simp*]: *x* ⊙ *inv x* = {*σ x*}
⟨*proof*⟩

lemma *invr* [*simp*]: *inv x* ⊙ *x* = {*τ x*}
⟨*proof*⟩

end

Next, I provide a more direct axiomatisation.

class *groupoid* = *catoid* + *inv-op* +
assumes *invs* [*simp*]: *x* ⊙ *inv x* = {*σ x*}

and *invt* [*simp*]: $inv\ x \odot x = \{\tau\ x\}$

subclass (**in** *groupoid*) *st-multigroupoid*
⟨*proof*⟩

sublocale *groupoid* ⊆ *lrgpd*: *groupoid* $\lambda x\ y.\ y \odot x$ *tgt src inv*
⟨*proof*⟩

lemma (**in** *groupoid*) *bao4* [*simp*]: $inv\ (inv\ x) = x$
⟨*proof*⟩

lemma (**in** *groupoid*) *rev1*:
$x \in y \odot z \Longrightarrow y \in x \odot inv\ z$
⟨*proof*⟩

lemma (**in** *groupoid*) *rev2*:
$x \in y \odot z \Longrightarrow z \in inv\ y \odot x$
⟨*proof*⟩

lemma (**in** *groupoid*) *rev1-eq*: $(y \in x \odot (inv\ z)) = (x \in y \odot z)$
⟨*proof*⟩

lemma (**in** *groupoid*) *rev2-eq*: $(z \in (inv\ y) \odot x) = (x \in y \odot z)$
⟨*proof*⟩

The following fact show that the axiomatisation above captures indeed groupoids.

lemma (**in** *groupoid*) *lr-mgpd-partial*:
assumes $x \in y \odot z$
and $x' \in y \odot z$
shows $x = x'$
⟨*proof*⟩

subclass (**in** *groupoid*) *single-set-category*
⟨*proof*⟩

Hence st-groupoids are indeed single-set categories in which all arrows are isomorphisms.

lemma (**in** *groupoid*) *src-canc1*:
assumes $\tau\ z = \sigma\ x$
and $\tau\ z = \sigma\ y$
and $z \otimes x = z \otimes y$
shows $x = y$
⟨*proof*⟩

lemma (**in** *groupoid*) *tgt-canc1*:
assumes $\tau\ x = \sigma\ z$
and $\tau\ y = \sigma\ z$
and $x \otimes z = y \otimes z$

shows $x = y$
⟨*proof*⟩

The following lemmas are from Theorem 5.2 of Jónsson and Tarski's BAO II article.

lemma (**in** *groupoid*) *bao1* [*simp*]: $x \otimes (inv\ x \otimes x) = x$
⟨*proof*⟩

lemma (**in** *groupoid*) *bao2* [*simp*]: $(x \otimes inv\ x) \otimes x = x$
⟨*proof*⟩

lemma (**in** *groupoid*) *bao5*:
$\tau\ x = \sigma\ y \Longrightarrow inv\ x \otimes x = y \otimes inv\ y$
⟨*proof*⟩

lemma (**in** *groupoid*) *bao6*: $Inv\ (x \odot y) = inv\ y \odot inv\ x$
⟨*proof*⟩

3.3 Axioms of relation algebra

I formalise a special case of a famous theorem of Jónsson and Tarski, showing that groupoids lift to relation algebras at powerset level. All axioms not related to converse have already been considered previously.

lemma (**in** *groupoid*) *Inv-invol* [*simp*]: $Inv\ (Inv\ X) = X$
⟨*proof*⟩

lemma (**in** *groupoid*) *Inv-contrav*: $Inv\ (X \star Y) = Inv\ Y \star Inv\ X$
⟨*proof*⟩

lemma (**in** *groupoid*) *residuation*: $Inv\ X \star -(X \star Y) \subseteq -Y$
⟨*proof*⟩

lemma (**in** *groupoid*) *modular-law*: $(X \star Y) \cap Z \subseteq (X \cap (Z \star Inv\ Y)) \star Y$
⟨*proof*⟩

lemma (**in** *groupoid*) *dedekind*: $(X \star Y) \cap Z \subseteq (X \cap (Z \star Inv\ Y)) \star (Y \cap (Inv\ X \star Z))$
⟨*proof*⟩

In sum, this shows that the powerset lifting of a groupoid is a relation algebra. I link this formally with relations in an interpretation statement in another component.

Jónsson and Tarski's axioms of relation algebra are slightly different. It is routine to related them formally with those used here. It might also be interested to use their partiality-by-closure approach to defining groupoids in a setting with explicit carrier sets in another Isabelle formalisation.

lemma (**in** *groupoid*) *Inv-compl*: $Inv\ (-X) = -(Inv\ X)$

⟨*proof*⟩

lemma (**in** *groupoid*) *Inv-inter*: *Inv* $(X \cap Y) =$ *Inv* $X \cap$ *Inv* Y
⟨*proof*⟩

lemma (**in** *groupoid*) *Inv-Un*: *Inv* $(\bigcap \mathcal{X}) = (\bigcap X \in \mathcal{X}.$ *Inv* $X)$
⟨*proof*⟩

end

4 Lifting catoids to modal powerset quantales

theory *Catoid-Lifting*
imports *Catoid Quantales-Converse.Modal-Quantale*

begin

instantiation *set* :: (*catoid*) *monoid-mult*

begin

definition *one-set* :: *'a set* **where**
1 = *sfix*

definition *times-set* :: *'a set* ⇒ *'a set* ⇒ *'a set* **where**
$X * Y = X \star Y$

instance
⟨*proof*⟩

end

instantiation *set* :: (*catoid*) *semiring-one-zero*

begin

definition *zero-set* :: *'a set* **where**
zero-set = {}

definition *plus-set* :: *'a set* ⇒ *'a set* ⇒ *'a set* **where**
$X + Y = X \cup Y$

instance
⟨*proof*⟩

end

instantiation *set* :: (*catoid*) *dioid*

begin

instance
⟨*proof*⟩

end

instantiation *set* :: (*local-catoid*) *domain-semiring*

begin

definition *domain-op-set* :: $'a$ *set* $\Rightarrow$ $'a$ *set* **where**
dom X = *Src X*

instance
⟨*proof*⟩

end

instantiation *set* :: (*local-catoid*) *range-semiring*

begin

definition *range-op-set* :: $'a$ *set* $\Rightarrow$ $'a$ *set* **where**
cod X = *Tgt X*

instance
⟨*proof*⟩

end

instantiation *set* :: (*local-catoid*) *dr-modal-semiring*

begin

instance
⟨*proof*⟩

end

instantiation *set* :: (*catoid*) *quantale*

begin

instance
⟨*proof*⟩

end

instantiation *set* :: (*local-catoid*) *domain-quantale*

begin

instance
⟨*proof*⟩

end

instantiation *set* :: (*local-catoid*) *codomain-quantale*

begin

instance
⟨*proof*⟩

end

instantiation *set* :: (*local-catoid*) *dc-modal-quantale*

begin

instance
⟨*proof*⟩

end

end

5 Lifting groupoids to powerset Dedekind quantales and powerset relation algebras

theory *Groupoid-Lifting*
imports *Groupoid Quantales-Converse.Quantale-Converse Catoid-Lifting Relation-Algebra.Relation-Algebra*

begin

instantiation *set* :: (*groupoid*) *dedekind-quantale*
begin

definition *invol-set* :: *'a set* ⇒ *'a set* **where**
invol = *Inv*

instance
⟨*proof*⟩

end

instantiation *set* :: (*groupoid*) *boolean-dedekind-quantale*

begin

instance⟨*proof*⟩

end

instantiation *set* :: (*groupoid*) *relation-algebra*

begin

definition *composition-set* :: $'a$ *set* ⇒ $'a$ *set* ⇒ $'a$ *set* **where**
composition-set $x\ y = x \star y$

definition *converse-set* :: $'a$ *set* ⇒ $'a$ *set* **where**
converse = *Inv*

definition *unit-set* :: $'a$ *set* **where**
unit-set = *sfix*

instance
⟨*proof*⟩

end

end

6 Multimonoids

theory *Multimonoid*
imports *Catoid*

begin

context *multimagma*
begin

6.1 Unital multimagmas

This component presents an alternative approach to catoids, as multisemigroups with many units. This is more akin to the formalisation of single-set categories in Chapter I of Mac Lane's book, but in fact this approach to axiomatising categories goes back to the middle of the twentieth century.

Units can already be defined in multimagmas.

definition *munitl* $e = ((\exists x.\ x \in e \odot x) \wedge (\forall x\ y.\ y \in e \odot x \longrightarrow y = x))$

definition *munitr* $e = ((\exists x.\ x \in x \odot e) \wedge (\forall x\ y.\ y \in x \odot e \longrightarrow y = x))$

abbreviation *munit* $e \equiv$ (*munitl* $e \vee$ *munitr* e)

end

A multimagma is unital if every element has a left and a right unit.

class *unital-multimagma-var* = *multimagma* +
  assumes *munitl-ex*: $\forall x.\exists e.$ *munitl* $e \wedge \Delta\ e\ x$
  assumes *munitr-ex*: $\forall x.\exists e.$ *munitr* $e \wedge \Delta\ x\ e$

begin

lemma *munitl-ex-var*: $\forall x.\exists e.$ *munitl* $e \wedge x \in e \odot x$
  ⟨*proof*⟩

lemma *unitl*: $\bigcup\{e \odot x\ |e.$ *munitl* $e\} = \{x\}$
  ⟨*proof*⟩

lemma *munitr-ex-var*: $\forall x.\exists e.$ *munitr* $e \wedge x \in x \odot e$
  ⟨*proof*⟩

lemma *unitr*: $\bigcup\{x \odot e\ |e.$ *munitr* $e\} = \{x\}$
  ⟨*proof*⟩

end

Here is an alternative definition.

class *unital-multimagma* = *multimagma* +
  fixes E :: *'a set*
  assumes *El*: $\bigcup\{e \odot x\ |e.\ e \in E\} = \{x\}$
  and *Er*: $\bigcup\{x \odot e\ |e.\ e \in E\} = \{x\}$

begin

lemma *E1*: $\forall e \in E.\ (\forall x\ y.\ y \in e \odot x \longrightarrow y = x)$
  ⟨*proof*⟩

lemma *E2*: $\forall e \in E.\ (\forall x\ y.\ y \in x \odot e \longrightarrow y = x)$
  ⟨*proof*⟩

lemma *El11*: $\forall x.\exists e \in E.\ x \in e \odot x$
  ⟨*proof*⟩

lemma *El12*: $\forall x.\exists e \in E.\ e \odot x = \{x\}$
  ⟨*proof*⟩

lemma *Er11*: $\forall x.\exists e \in E.\ x \in x \odot e$
  ⟨*proof*⟩

lemma *Er12*: $\forall x.\exists e \in E.\ x \odot e = \{x\}$
  ⟨*proof*⟩

Units are "orthogonal" idempotents.

lemma *unit-id*: $\forall e \in E.\ e \in e \odot e$
  ⟨*proof*⟩

lemma *unit-id-eq*: $\forall e \in E.\ e \odot e = \{e\}$
  ⟨*proof*⟩

lemma *unit-comp*:
  assumes $e_1 \in E$
and $e_2 \in E$
and $\Delta\ e_1\ e_2$
shows $e_1 = e_2$
⟨*proof*⟩

lemma *unit-comp-iff*: $e_1 \in E \Longrightarrow e_2 \in E \Longrightarrow (\Delta\ e_1\ e_2 = (e_1 = e_2))$
  ⟨*proof*⟩

lemma $\forall e \in E.\exists x.\ x \in e \odot x$
  ⟨*proof*⟩

lemma $\forall e \in E.\exists x.\ x \in x \odot e$
  ⟨*proof*⟩

sublocale *unital-multimagma-var*
  ⟨*proof*⟩

Now it is clear that the two definitions are equivalent.

The next two lemmas show that the set of units is a left and right unit of composition at powerset level.

lemma *conv-unl*: $E \star X = X$
  ⟨*proof*⟩

lemma *conv-unr*: $X \star E = X$
  ⟨*proof*⟩

end

6.2 Multimonoids

A multimonoid is a unital multisemigroup.

class *multimonoid* = *multisemigroup* + *unital-multimagma*

begin

In a multimonoid, left and right units are unique for each element.

lemma *munits-uniquel*: $\forall x.\exists!e.\ e \in E \wedge e \odot x = \{x\}$
⟨*proof*⟩

lemma *munits-uniquer*: $\forall x.\exists!e.\ e \in E \wedge x \odot e = \{x\}$
⟨*proof*⟩

In a monoid, there is of course one single unit, and my definition of many units reduces to this one.

lemma *units-unique*: $(\forall x\ y.\ \Delta\ x\ y) \Longrightarrow \exists!e.\ e \in E$
⟨*proof*⟩

lemma *units-rm2l*: $e_1 \in E \Longrightarrow e_2 \in E \Longrightarrow \Delta\ e_1\ x \Longrightarrow \Delta\ e_2\ x \Longrightarrow e_1 = e_2$
⟨*proof*⟩

lemma *units-rm2r*: $e_1 \in E \Longrightarrow e_2 \in E \Longrightarrow \Delta\ x\ e_1 \Longrightarrow \Delta\ x\ e_2 \Longrightarrow e_1 = e_2$
⟨*proof*⟩

One can therefore express the functional relationship between elements and their units in terms of explict (source and target) maps – as in catoids.

definition *so* :: $'a \Rightarrow 'a$ **where**
$so\ x = (THE\ e.\ e \in E \wedge e \odot x = \{x\})$

definition *ta* :: $'a \Rightarrow 'a$ **where**
$ta\ x = (THE\ e.\ e \in E \wedge x \odot e = \{x\})$

abbreviation *So* :: $'a\ set \Rightarrow 'a\ set$ **where**
$So\ X \equiv image\ so\ X$

abbreviation *Ta* :: $'a\ set \Rightarrow 'a\ set$ **where**
$Ta\ X \equiv image\ ta\ X$

end

6.3 Multimonoids and catoids

It is now easy to show that every catoid is a multimonoid and vice versa.

One cannot have both sublocale statements at the same time.

The converse direction requires some preparation.

lemma (**in** *multimonoid*) *so-unit*: $so\ x \in E$
⟨*proof*⟩

lemma (**in** *multimonoid*) *ta-unit*: $ta\ x \in E$

⟨*proof*⟩

lemma (**in** *multimonoid*) *so-absorbl*: $so\ x \odot x = \{x\}$
⟨*proof*⟩

lemma (**in** *multimonoid*) *ta-absorbr*: $x \odot ta\ x = \{x\}$
⟨*proof*⟩

lemma (**in** *multimonoid*) *semi-locality*: $\Delta\ x\ y \Longrightarrow ta\ x = so\ y$
⟨*proof*⟩

sublocale *multimonoid* ⊆ *monlr*: *catoid* (⊙) *so ta*
⟨*proof*⟩

6.4 From multimonoids to categories

Single-set categories are precisely local partial monoids, that is, object-free categories as in Chapter I of Mac Lane's book.

class *local-multimagma* = *multimagma* +
assumes *locality*: $v \in x \odot y \Longrightarrow \Delta\ y\ z \Longrightarrow \Delta\ v\ z$

class *local-multisemigroup* = *multisemigroup* + *local-multimagma*

In this context, a semicategory is an object-free category without identity arrows

class *of-semicategory* = *local-multisemigroup* + *functional-semigroup*

begin

lemma *part-locality*: $\Delta\ x\ y \Longrightarrow \Delta\ y\ z \Longrightarrow \Delta\ (x \otimes y)\ z$
⟨*proof*⟩

lemma *part-locality-var*: $\Delta\ x\ y \Longrightarrow \Delta\ y\ z \Longrightarrow (x \odot y) \star \{z\} \neq \{\}$
⟨*proof*⟩

lemma *locality-iff*: $(\Delta\ x\ y \wedge \Delta\ y\ z) = (\Delta\ x\ y \wedge \Delta\ (x \otimes y)\ z)$
⟨*proof*⟩

lemma *locality-iff-var*: $(\Delta\ x\ y \wedge \Delta\ y\ z) = (\Delta\ x\ y \wedge (x \odot y) \star \{z\} \neq \{\})$
⟨*proof*⟩

end

class *partial-monoid* = *multimonoid* + *functional-magma*

class *local-multimonoid* = *multimonoid* + *local-multimagma*

begin

lemma *sota-locality*: $ta\ x = so\ y \Longrightarrow \Delta\ x\ y$
⟨*proof*⟩

lemma *So-local*: $So\ (x \odot so\ y) = So\ (x \odot y)$
⟨*proof*⟩

lemma *Ta-local*: $Ta\ (ta\ x \odot y) = Ta\ (x \odot y)$
⟨*proof*⟩

sublocale *locmm*: *local-catoid* $(\odot)$ *so ta*
⟨*proof*⟩

The following statements formalise compatibility properties.

lemma *local-conv*: $v \in x \odot y \Longrightarrow (\Delta\ v\ z = \Delta\ y\ z)$
⟨*proof*⟩

lemma *local-alt*: $e \in E \Longrightarrow x \in x \odot e \Longrightarrow y \in e \odot y \Longrightarrow \Delta\ x\ y$
⟨*proof*⟩

lemma *local-iff*: $\Delta\ x\ y = (\exists\, e \in E.\ \Delta\ x\ e \wedge \Delta\ e\ y)$
⟨*proof*⟩

lemma *local-iff2*: $(ta\ x = so\ y) = \Delta\ x\ y$
⟨*proof*⟩

end

Finally I formalise object-free categories. The axioms are essentially Mac Lane's, but a multioperation is used for arrow composition, to capture partiality.

class *of-category* = *of-semicategory* + *partial-monoid*

The next statements show that single-set categories based on catoids and object-free categories based on multimonoids are the same (we can only have one direction as a sublocale statement). It then follows from results about catoids and single-set categories that object-free categories are indeed categories. These results can be found in the catoid component. I do not present explicit proofs for object-free categories here.

sublocale *of-category* ⊆ *ofss-cat*: *single-set-category* - *so ta*
⟨*proof*⟩

6.5 Multimonoids and relational monoids

Relational monoids are monoids in the category Rel. They have been used previously to construct convolution algebras in another AFP entry. Here I show that relational monoids are isomorphic to multimonoids, but I do not integrate the AFP entry with relational monoids because it uses a historic quantale component, which is different from the quantale component in the AFP. Instead, I simply copy in the definitions leading to relational monoids and leave the consolidation of Isabelle theories to the future.

class *rel-magma* =
  fixes ϱ :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$

class *rel-semigroup* = *rel-magma* +
  assumes *rel-assoc*: $(\exists\, y.\ \varrho\ y\ u\ v \wedge \varrho\ x\ y\ w) = (\exists\, z.\ \varrho\ z\ v\ w \wedge \varrho\ x\ u\ z)$

class *rel-monoid* = *rel-semigroup* +
  fixes ξ :: $'a\ set$
  assumes *unitl-ex*: $\exists\, e \in \xi.\ \varrho\ x\ e\ x$
  and *unitr-ex*: $\exists\, e \in \xi.\ \varrho\ x\ x\ e$
  and *unitl-eq*: $e \in \xi \Longrightarrow \varrho\ x\ e\ y \Longrightarrow x = y$
  and *unitr-eq*: $e \in \xi \Longrightarrow \varrho\ x\ y\ e \Longrightarrow x = y$

Once again, only one of the two sublocale statements compiles.

sublocale *multimonoid* $\subseteq$ *rel-monoid* $\lambda x\ y\ z.\ x \in y \odot z\ E$
  ⟨*proof*⟩

end

References

[1] R. Brown. From groups to groupoids: A brief survey. *Bulletin of the London Mathematical Society*, 19:113–134, 1987.

[2] C. Calk, U. Fahrenberg, C. Johansen, G. Struth, and K. Ziemiański. ℓr-multisemigroups, modal quantales and the origin of locality. In *RAMiCS 2021*, volume 13027 of *LNCS*, pages 90–107. Springer, 2021.

[3] C. Calk, P. Malbos, D. Pous, and G. Struth. Higher catoids, higher quantales and their correspondences. *arXiv*, 2307.09253, 2023.

[4] J. Cranch, S. Doherty, and G. Struth. Relational semigroups and object-free categories. *arXiv*, 2001.11895, 2020.

[5] B. Dongol, V. B. F. Gomes, I. J. Hayes, and G. Struth. Partial semigroups and convolution algebras. *Archive of Formal Proofs*, 2017.

[6] U. Fahrenberg, C. Johansen, G. Struth, and K. Ziemiański. Catoids and modal convolution algebras. *Algebra Universalis*, 84:10, 2023.

[7] B. Jónsson and A. Tarski. Boolean algebras with operators. Part II. *American Journal of Mathematics*, 74(1):127–162, 1952.

[8] S. Mac Lane. *Categories for the Working Mathematician*, volume 5. Springer, second edition, 1998.

[9] E. W. Stark. Category theory with adjunctions and limits. *Archive of Formal Proofs*, 2016.